NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES (KARACHI CAMPUS)

FAST School of Computing

Spring 2024

Course: Software Engineering (CS3009)
Course Instructor: Ms. Javeria Farooq
Project Name: Restaurant Food Ordering App
Project Milestone: Manual , Automated and Path Testing

Team Members:

21K-3279	Insha Javed
21K-4503	Muhammad Tahir
21K-3317	Naqeeb Nadir
21K-4606	Sabika Shameel

Manual and Automated Testing
(RESTAURANT FOOD ORDERING APP)

Test Case 1: Google Cloud Login

1. Input: New user registration details (name, email, and password).
2. Action: Log in using the registered credentials.
3. Expected Output: Successful login or error message if login fails.

Test Case Table:

Test Case ID	Test Case Description	Input	Expected Output	Actual Output	Pass/Fail
1.2	User Login Success	Email: john@example.com, Password: Test123	Login Successful	Successfully logged in and redirected to homepage	Pass
1.3	User Login Failure (Invalid Credentials)	Email: john@example.com, Password: WrongPassword	Error: Invalid email or password	Incorrect email/password	Pass

Test Case 2: Homepage buttons

1. Input: click on orders, logout or profile
2. Action: Click on the buttons
3. Expected Output: Pages successfully viewed and rendered.

Test Case Table:

Test Case ID	Test Case Description	Input	Expected Output	Actual Output	Pass/Fail
2.1	Order button testing	Click on order button	Order id, date, price, products quantity viewed.	Order id, date, price, products quantity viewed.	Pass
2.2	Contact us button testing	Click on contact us button	Contact details and address viewed	Contact details and address viewed	Pass
2.3	Profile button testing	Click on profile button	Customer ID, email, name, and total orders viewed	Customer ID, email, name, and total orders viewed	Pass
2.4	Logout button testing	Click on logout button	Home page viewed, with an option to login	Home page viewed, with an option to login	Pass

Test Case 3: Menu Browsing and Selection

1. Input: Navigate to the menu section.
2. Action: Select a food item from the menu.
3. Expected Output: Item added to the cart.

Test Case Table:

Test Case ID	Test Case Description	Input/Action	Expected Output	Actual Output	Pass/Fail
3.1	Menu Display	Navigate to menu section	Menu items displayed	Menu items displayed	Pass
3.2	Item Selection	Select 'Pizza' from the menu	View more pizza options	View more pizza options	Pass
3.3	Item Customization	Select 'Burger' and choose 'Extra Cheese'	'Burger' with extra cheese added	Burger display having extra cheeses	Pass
3.4	Item Customization	Select ‘Pizza’ with plate	no menu having ingredient plate viewed	no menu having ingredient plate viewed	Pass

Test Case 4: Payment Processing

1. Input: Select a payment method.
2. Expected Output: Payment method confirmed.
3. Action: Enter payment details.

Test Case Table:

Test Case ID	Test Case Description	Input/Action	Expected Output	Actual Output	Pass/Fail
4.1	Select Payment Method	Choose 'Credit Card' as payment method	Show credentials page	Show credentials page	Pass
4.2	Select Payment Method	Choose cash on delivery	Oder confirmed	Order confirmed	Pass

Test Case 5: Order Placement

1. Input: click on Order Placement./Add multiple items to the cart.
2. Expected Output: Items displayed in the cart.
3. Action: Proceed to checkout and place the order.

Test Case Table:

Test Case ID	Test Case Description	Input/Action	Expected Output	Actual Output	Pass/Fail
5.1	Add Items to Cart	Add 'Pizza', 'Burger', and 'Coke' to cart	Items displayed in the cart	Items are displaying	Pass
5.2	Proceed to Checkout	Click on 'Checkout' button	Proceed to payment page	View credentials page	Pass
5.3	Place Order	Complete payment process	Order confirmation message and redirected to orders page	Order confirmed message and redirected to orders page	Pass
5.4	Checkout without login	Click on 'Checkout' button	Take to sign in page on google	If signed in successfully, view credentials page	Pass

Test Case 6: Cart Transactions

1. Action: Add/remove items to/from the cart.
2. Expected Output: Items displayed in the cart or items removed from the cart

Test Case Table:

Test Case ID	Test Case Description	Input/Action	Expected Output	Actual Output	Pass/Fail
6.1	Add Items to Cart	Add 'Pizza', 'Burger', and 'Coke' to cart	Items displayed in the cart	Items displayed in the cart	Pass
6.2	Remove Item from Cart	Remove 'Coke' from the cart	'Coke' removed from the cart	Coke removes, while other items remained	Pass
6.3	Add items costing less than $50	Add 'Pizza'	Item cost, and delivery cost $5 viewing	Item cost, and delivery cost $5 viewing	Pass
6.4	Add items costing more than $50	Add pizza, burger etc	Item cost viewed and no delivery charges	Item cost viewed and no delivery charges	Pass

Test Cases 7: Customizable Options of Sicilian Pizza:

Test Case 7.1: Size Selection

1. Input: Select "Medium" size for Sicilian pizza.
2. Action: Choose Medium size from the size options.
3. Expected Output: Medium size selected for the pizza.

Test Case Table:

Test Case ID	Test Case Description	Input/Action	Expected Output	Actual Output	Pass/Fail
7.1.1	Select Small Size	Choose Small size from options	Small size selected, ingredients, price viewed	Small, price and ingredient list viewed	Pass
7.1.2	Select Medium Size	Choose Medium size from options	Small size selected, ingredients, price viewed	medium, price and ingredient list viewed	Pass
7.1.3	Select Large Size	Choose Large size from options	Small size selected, ingredients, price viewed	large, price and ingredient list viewed	Pass
7.1.4	Increase/decrease quantity	Click on arrows	Increment by one every time when increased and decrease until 1 left	Successfully increased or decreased quantity	Pass

Test Case 7.2: Crust Type Selection:

1. Input: Select "Thick Crust" for Sicilian pizza.
2. Action: Choose Thick Crust from the crust type options. Expected Output: Thick Crust selected for the pizza.

Test Case Table:

Test Case ID	Test Case Description	Input/Action	Expected Output	Actual Output	Pass/Fail
7.2.1	Select Thin Crust	Choose Thin Crust from options	Thin Crust selected	Thin Crust selected	Pass
7.2.2	Select Thick Crust	Choose Thick Crust from options	Thick Crust selected	Thick Crust selected	Pass
7.2.3	Select Stuffed Crust	Choose Stuffed Crust from options	Stuffed Crust selected	Stuffed Crust selected	Pass

Test Case 7.3: Remove Toppings:

1. Input: Remove "Olives" from the toppings.
2. Action: Deselect Olives from the extra toppings list.
3. Expected Output: Olives removed from the toppings.

Test Case Table:

Test Case ID	Test Case Description	Input/Action	Expected Output	Actual Output	Pass/Fail
7.3.1	Remove Olives Topping	Deselect Olives from extra toppings list	Olives removed from extra toppings	View items without olives	Pass
7.3.2	Remove Onions Topping	Deselect Onions from extra toppings list	Onions removed from extra toppings	View items without onions	Pass
7.3.3	Remove Jalapenos Topping	Deselect Jalapenos from extra toppings list	Jalapenos removed from extra toppings	View items without jalapenos	Pass

Test Case 7.4: Special Instructions:

1. Input: Enter special instruction as "Extra crispy crust".
2. Action: Input "Extra crispy crust" in the special instructions text field.
3. Expected Output: Special instruction "Extra crispy crust" added.

Test Case Table:

Test Case ID	Test Case Description	Input/Action	Expected Output	Actual Output	Pass/Fail
7.4.1	Add Special Instruction	Enter "Extra crispy crust" in text field	"Extra crispy crust" added	Instructions sent to admin	Pass
7.4.2	Add Gluten-Free Instruction	Enter "Gluten-free crust" in text field	"Gluten-free crust" added	Instructions sent to admin	Pass
7.4.3	Add Well-Done Crust Instruction	Enter "Well-done crust" in text field	"Well-done crust" added	Instructions sent to admin	Pass

Test Cases 8: Admin Dashboard testing (Add product):

1. Input: Click on Add product button.

2. Action: click on a field and enter.
3. Expected Output: form must be submitted.

Test Case Table:

Test Case ID	Test Case Description	Input/Action	Expected Output	Actual Output	Pass/Fail
8.1	Product title/description/options/ingredient	Insert a character	accepted	accepted	Pass
8.2	Product title/description/options/ingredient	Enter a numeric value	Invalid input	Invalid input	Pass
8.3	Upload image	click	Take to pc files	Take to pc files	Pass
8.4	Leave any field empty	Leave price empty and submit	Please fill in all field	Please fill in all field	Pass

Test Cases 9: Admin Dashboard testing (Add Category)

1. Input: Click on Add Category button.
2. Action: click on an add category button.
3. Expected Output: form must be submitted.

Test Case Table:

Test Case ID	Test Case Description	Input/Action	Expected Output	Actual Output	Pass/Fail
9.1	Title, description and slug	Insert a character	accepted	accepted	Pass
9.2	Title, description and slug	Enter a numeric value	Invalid input	Invalid input	Pass
9.3	Upload image	click	Take to pc files	Take to pc files	Pass
9.4	Leave any field empty	Leave price empty and submit	Please fill in all field	Please fill in all field	Pass

Test Case 10: Admin Dashboard testing (View orders):

1. Input: Click on View Orders button.

2. Action: Click on orders button.
3. Expected Output: record viewed.

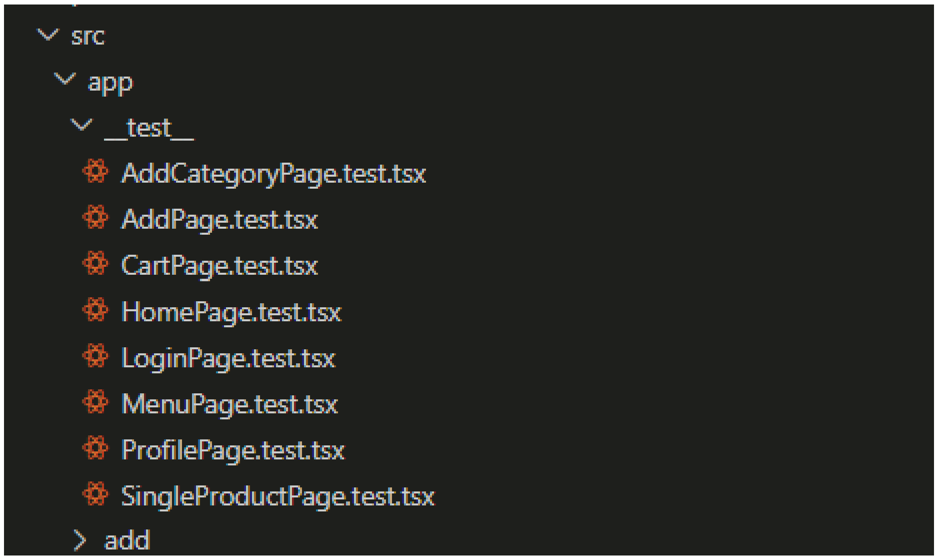
Test Case Table:

Test Case ID	**Test Case Description**	**Input/Action**	**Expected Output**	**Actual Output**	**Pass/Fail**
10.1	Order button testing	Click on button	View order id, date, price, quantity, and status	View order id, date, price, quantity, and status	Pass
10.2	Status updating	Enter a numeric value	Invalid input	Invalid input	Pass
10.3	Status updating	Enter a character value	Order status updated	Order status updated	Pass

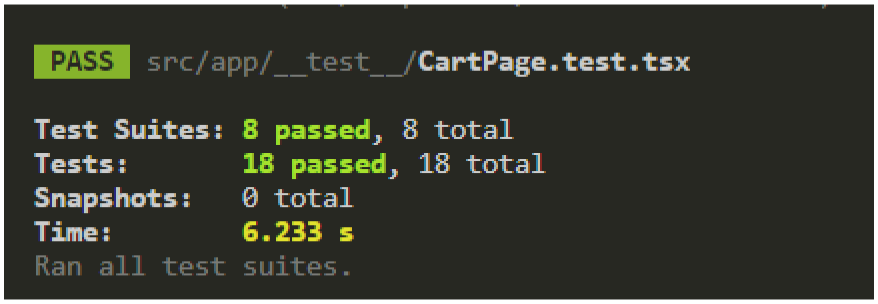
ADMIN AND USER SIDE AUTOMATED TEST CODES

Tools: Jest, Babel and testing-library/react

TOTAL AUTOMATED TESTS:

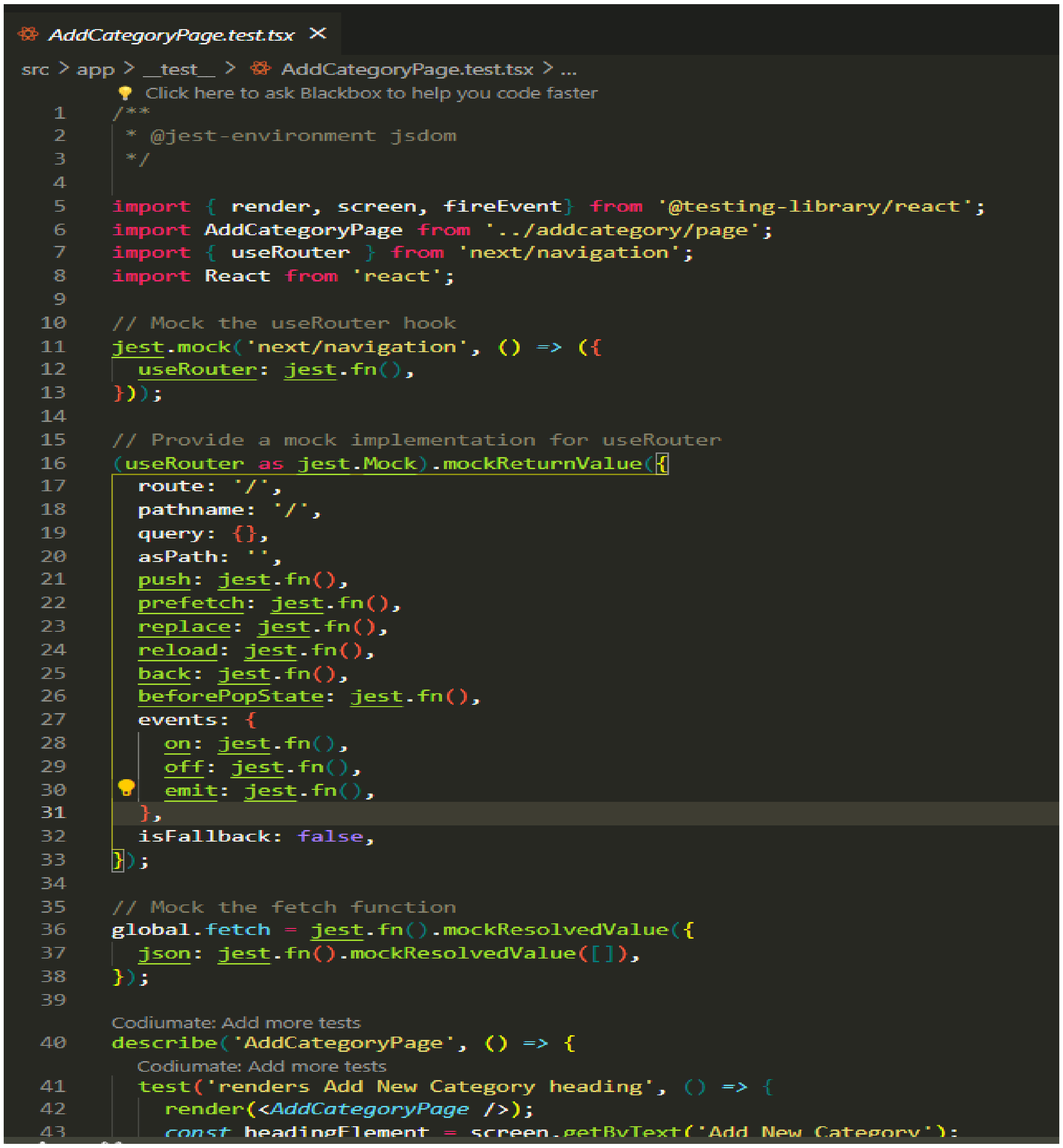


```
src
  app
    __test__
      AddCategoryPage.test.tsx
      AddPage.test.tsx
      CartPage.test.tsx
      HomePage.test.tsx
      LoginPage.test.tsx
      MenuPage.test.tsx
      ProfilePage.test.tsx
      SingleProductPage.test.tsx
    add
```

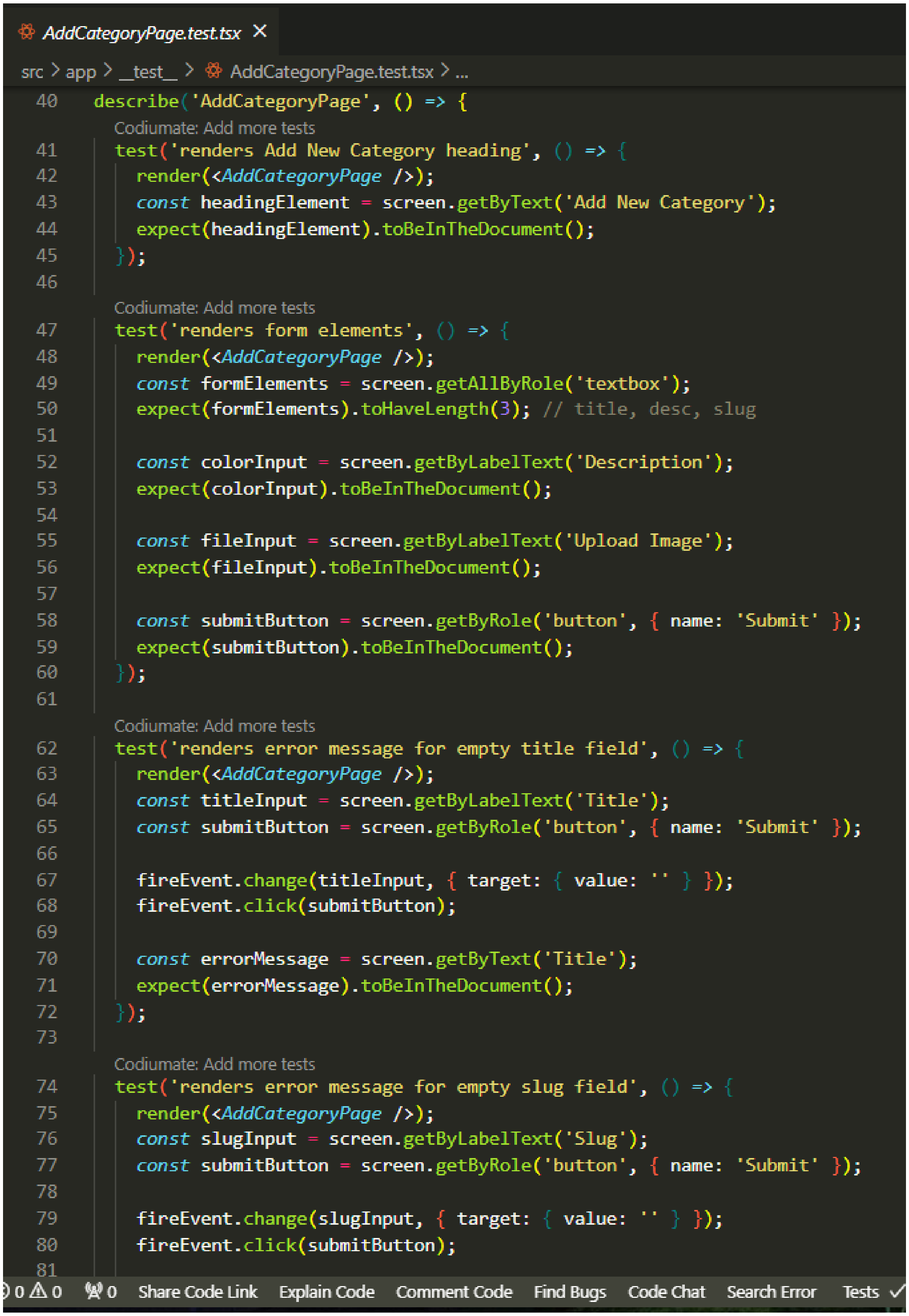


```
PASS  src/app/__test__/CartPage.test.tsx

Test Suites: 8 passed, 8 total
Tests:       18 passed, 18 total
Snapshots:   0 total
Time:        6.233 s
Ran all test suites.
```

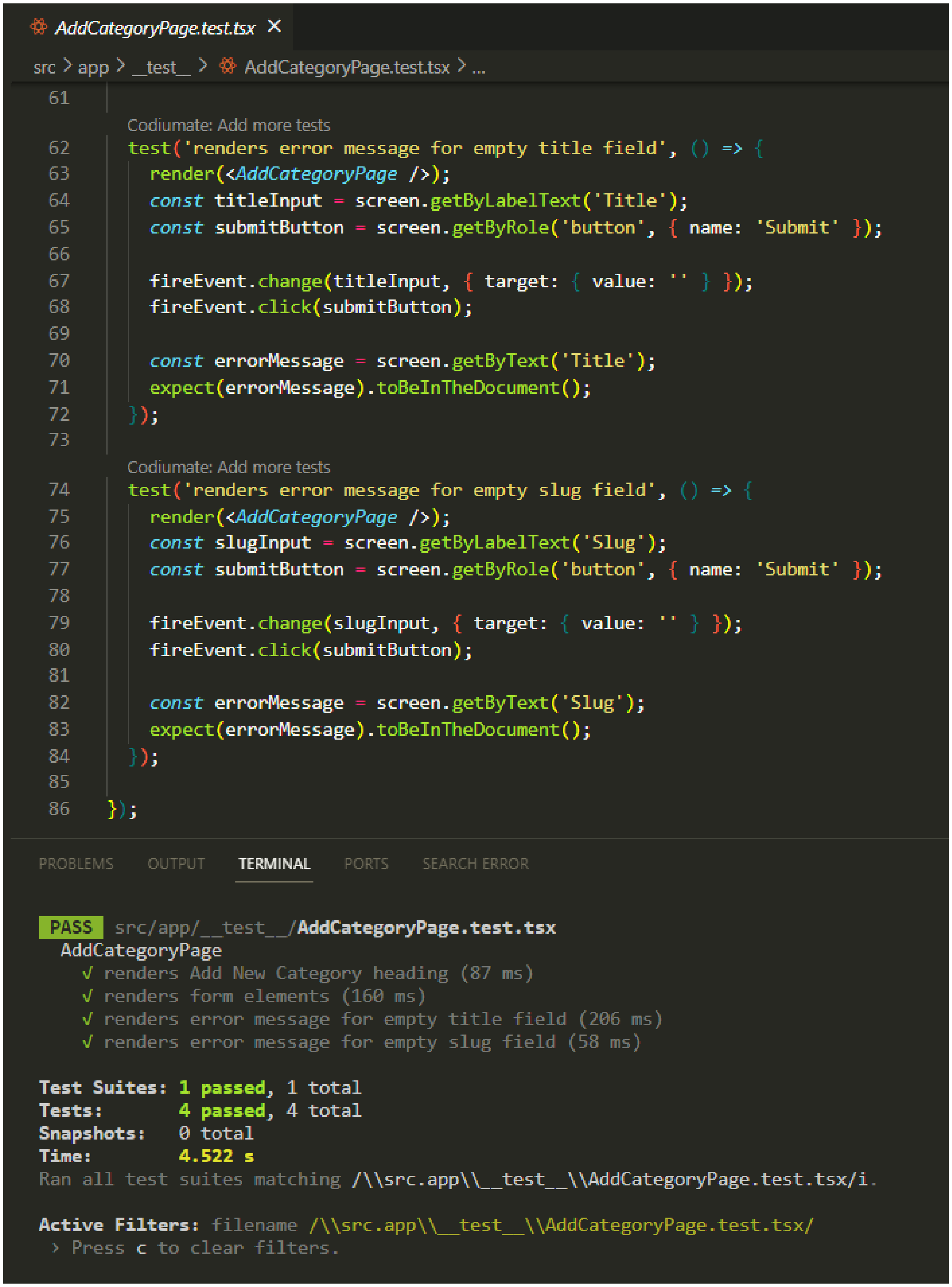
ADD CATEGORY PAGE:


AddCategoryPage.test.tsx
src > app > __test__ > AddCategoryPage.test.tsx > ...
Click here to ask Blackbox to help you code faster
Codiumate: Add more tests

```
/**
 * @jest-environment jsdom
 */

import { render, screen, fireEvent} from '@testing-library/react';
import AddCategoryPage from '../addcategory/page';
import { useRouter } from 'next/navigation';
import React from 'react';

// Mock the useRouter hook
jest.mock('next/navigation', () => ({
  useRouter: jest.fn(),
}));

// Provide a mock implementation for useRouter
(useRouter as jest.Mock).mockReturnValue({
  route: '/',
  pathname: '/',
  query: {},
  asPath: '',
  push: jest.fn(),
  prefetch: jest.fn(),
  replace: jest.fn(),
  reload: jest.fn(),
  back: jest.fn(),
  beforePopState: jest.fn(),
  events: {
    on: jest.fn(),
    off: jest.fn(),
    emit: jest.fn(),
  },
  isFallback: false,
});

// Mock the fetch function
global.fetch = jest.fn().mockResolvedValue({
  json: jest.fn().mockResolvedValue([]),
});

describe('AddCategoryPage', () => {
  test('renders Add New Category heading', () => {
    render(<AddCategoryPage />);
    const headingElement = screen.getByText('Add New Category');
```



```tsx
describe('AddCategoryPage', () => {
  Codiumate: Add more tests
  test('renders Add New Category heading', () => {
    render(<AddCategoryPage />);
    const headingElement = screen.getByText('Add New Category');
    expect(headingElement).toBeInTheDocument();
  });

  Codiumate: Add more tests
  test('renders form elements', () => {
    render(<AddCategoryPage />);
    const formElements = screen.getAllByRole('textbox');
    expect(formElements).toHaveLength(3); // title, desc, slug

    const colorInput = screen.getByLabelText('Description');
    expect(colorInput).toBeInTheDocument();

    const fileInput = screen.getByLabelText('Upload Image');
    expect(fileInput).toBeInTheDocument();

    const submitButton = screen.getByRole('button', { name: 'Submit' });
    expect(submitButton).toBeInTheDocument();
  });

  Codiumate: Add more tests
  test('renders error message for empty title field', () => {
    render(<AddCategoryPage />);
    const titleInput = screen.getByLabelText('Title');
    const submitButton = screen.getByRole('button', { name: 'Submit' });

    fireEvent.change(titleInput, { target: { value: '' } });
    fireEvent.click(submitButton);

    const errorMessage = screen.getByText('Title');
    expect(errorMessage).toBeInTheDocument();
  });

  Codiumate: Add more tests
  test('renders error message for empty slug field', () => {
    render(<AddCategoryPage />);
    const slugInput = screen.getByLabelText('Slug');
    const submitButton = screen.getByRole('button', { name: 'Submit' });

    fireEvent.change(slugInput, { target: { value: '' } });
    fireEvent.click(submitButton);

```


AddCategoryPage.test.tsx ×

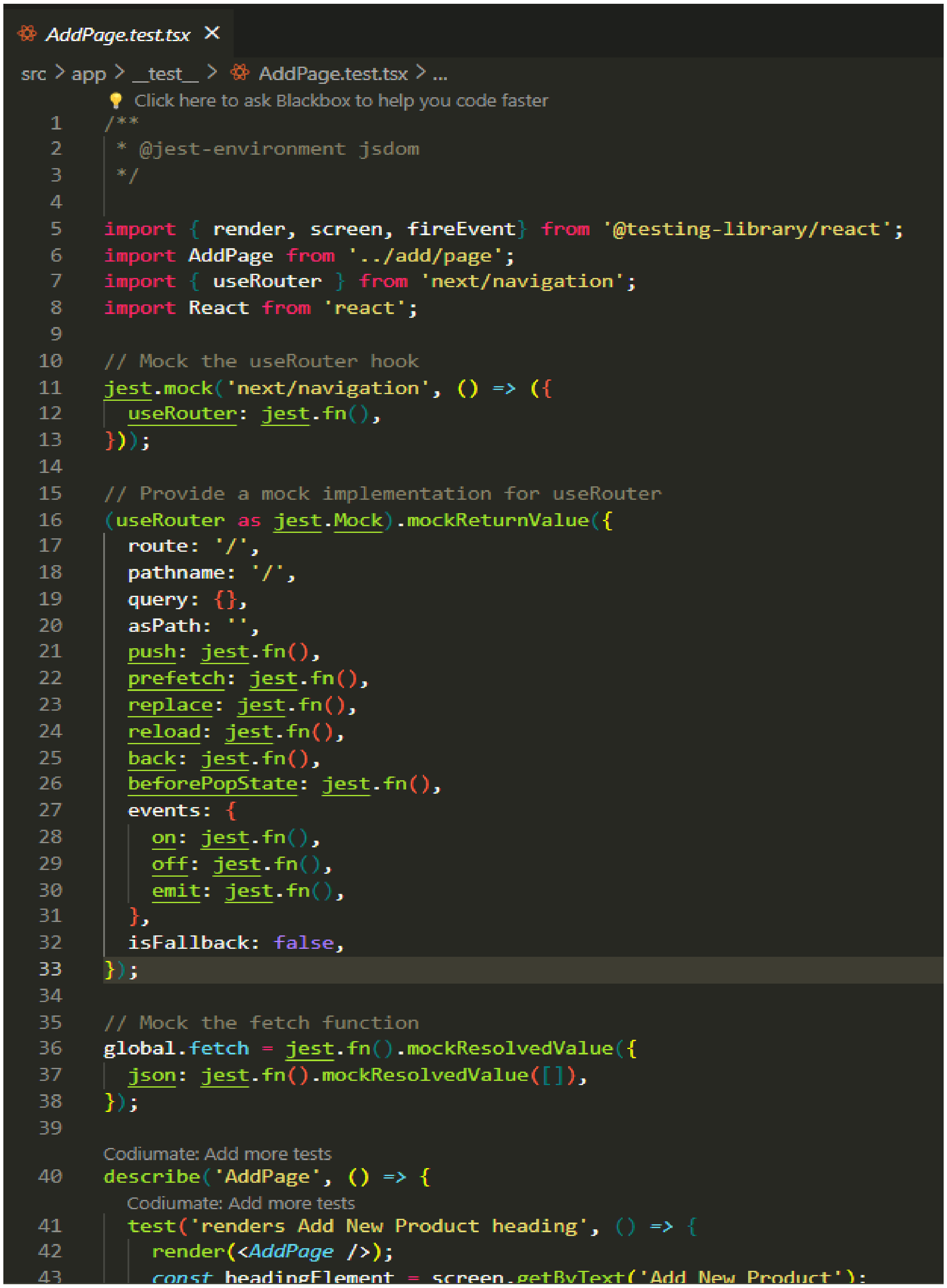
src > app > __test__ > AddCategoryPage.test.tsx > ...

```
Codiumate: Add more tests
test('renders error message for empty title field', () => {
  render(<AddCategoryPage />);
  const titleInput = screen.getByLabelText('Title');
  const submitButton = screen.getByRole('button', { name: 'Submit' });

  fireEvent.change(titleInput, { target: { value: '' } });
  fireEvent.click(submitButton);

  const errorMessage = screen.getByText('Title');
  expect(errorMessage).toBeInTheDocument();
});

Codiumate: Add more tests
test('renders error message for empty slug field', () => {
  render(<AddCategoryPage />);
  const slugInput = screen.getByLabelText('Slug');
  const submitButton = screen.getByRole('button', { name: 'Submit' });

  fireEvent.change(slugInput, { target: { value: '' } });
  fireEvent.click(submitButton);

  const errorMessage = screen.getByText('Slug');
  expect(errorMessage).toBeInTheDocument();
});

});
```

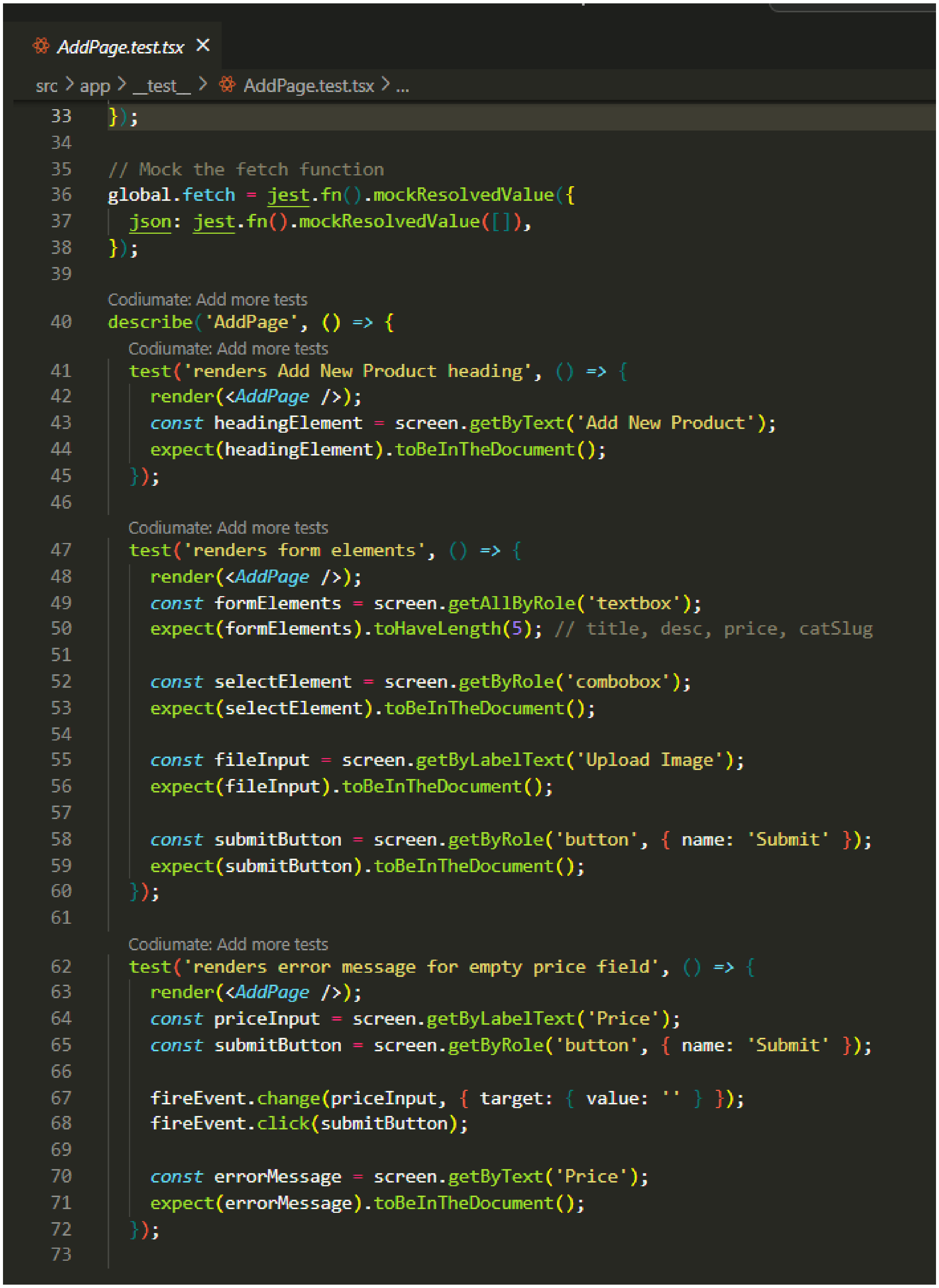
PROBLEMS OUTPUT TERMINAL PORTS SEARCH ERROR

```
PASS  src/app/__test__/AddCategoryPage.test.tsx
  AddCategoryPage
    √ renders Add New Category heading (87 ms)
    √ renders form elements (160 ms)
    √ renders error message for empty title field (206 ms)
    √ renders error message for empty slug field (58 ms)

Test Suites: 1 passed, 1 total
Tests:       4 passed, 4 total
Snapshots:   0 total
Time:        4.522 s
Ran all test suites matching /\\src.app\\__test__\\AddCategoryPage.test.tsx/i.

Active Filters: filename /\\src.app\\__test__\\AddCategoryPage.test.tsx/
 › Press c to clear filters.
```

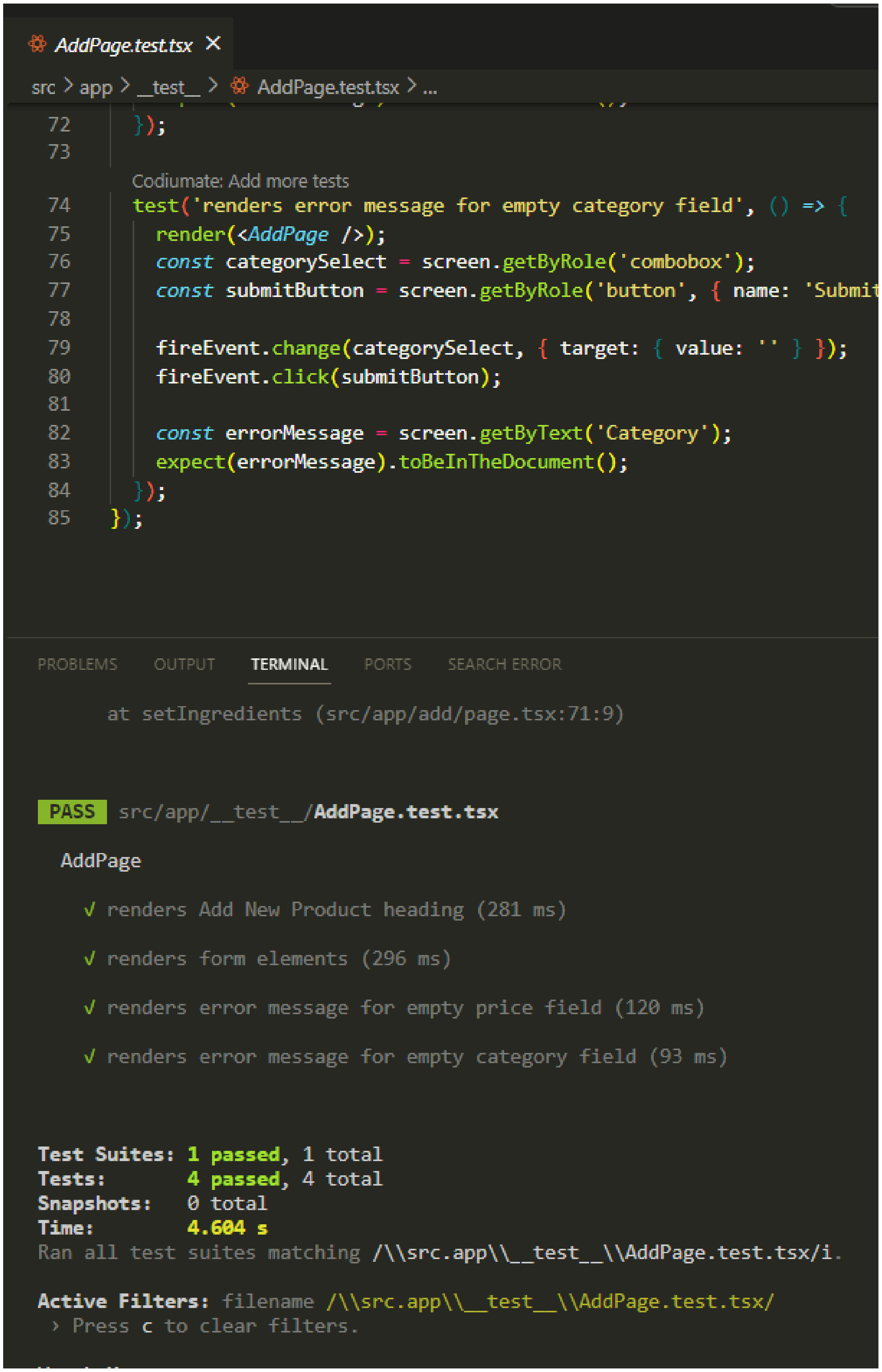
ADD PRODUCT PAGE:


AddPage.test.tsx ×
src > app > _test_ > AddPage.test.tsx > ...
Click here to ask Blackbox to help you code faster

```tsx
/**
 * @jest-environment jsdom
 */

import { render, screen, fireEvent} from '@testing-library/react';
import AddPage from '../add/page';
import { useRouter } from 'next/navigation';
import React from 'react';

// Mock the useRouter hook
jest.mock('next/navigation', () => ({
  useRouter: jest.fn(),
}));

// Provide a mock implementation for useRouter
(useRouter as jest.Mock).mockReturnValue({
  route: '/',
  pathname: '/',
  query: {},
  asPath: '',
  push: jest.fn(),
  prefetch: jest.fn(),
  replace: jest.fn(),
  reload: jest.fn(),
  back: jest.fn(),
  beforePopState: jest.fn(),
  events: {
    on: jest.fn(),
    off: jest.fn(),
    emit: jest.fn(),
  },
  isFallback: false,
});

// Mock the fetch function
global.fetch = jest.fn().mockResolvedValue({
  json: jest.fn().mockResolvedValue([]),
});

Codiumate: Add more tests
describe('AddPage', () => {
  Codiumate: Add more tests
  test('renders Add New Product heading', () => {
    render(<AddPage />);
    const headingElement = screen.getByText('Add New Product');
```


AddPage.test.tsx
src > app > __test__ > AddPage.test.tsx > ...
Codiumate: Add more tests

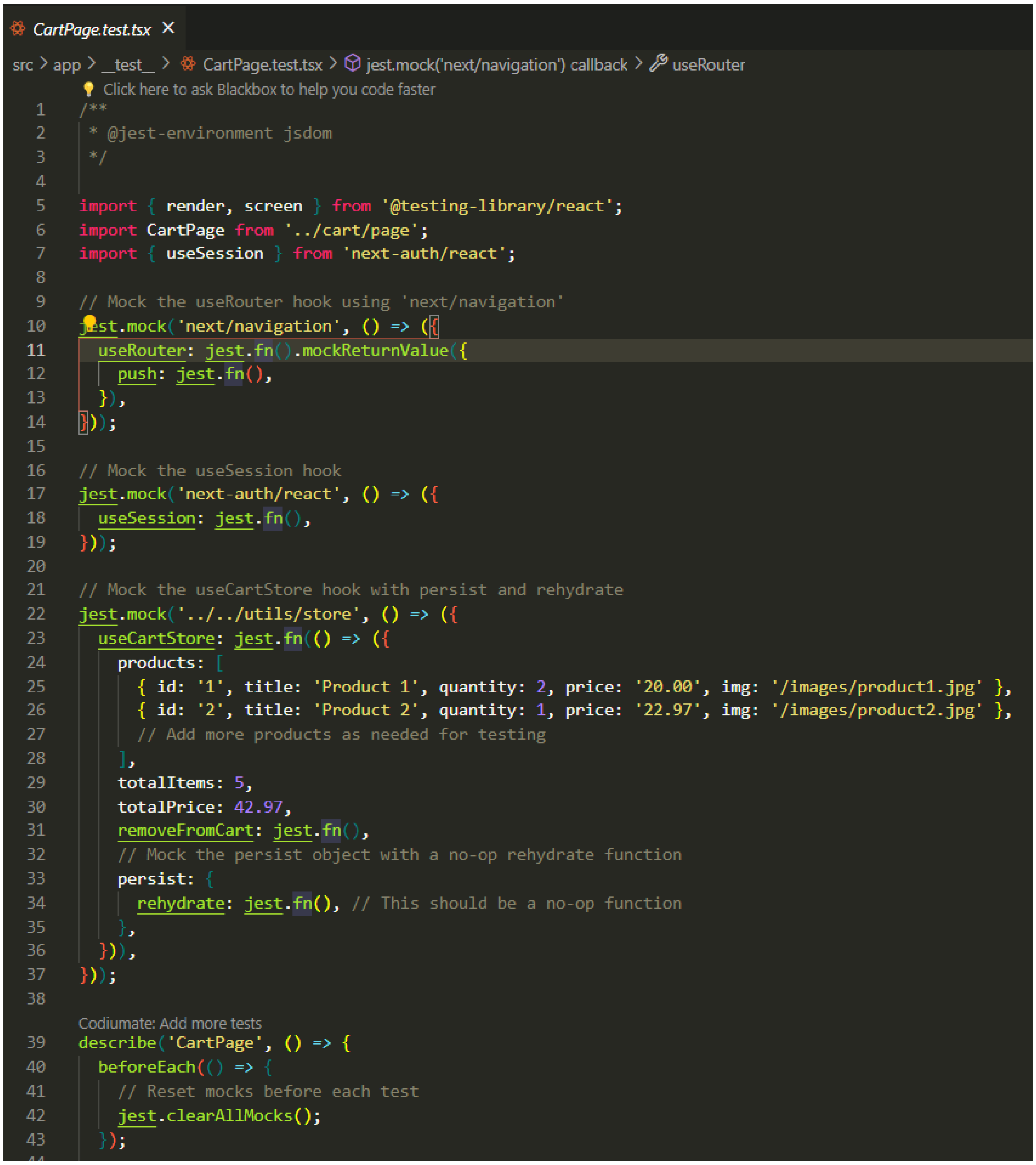
```tsx
});

// Mock the fetch function
global.fetch = jest.fn().mockResolvedValue({
  json: jest.fn().mockResolvedValue([]),
});

describe('AddPage', () => {
  test('renders Add New Product heading', () => {
    render(<AddPage />);
    const headingElement = screen.getByText('Add New Product');
    expect(headingElement).toBeInTheDocument();
  });

  test('renders form elements', () => {
    render(<AddPage />);
    const formElements = screen.getAllByRole('textbox');
    expect(formElements).toHaveLength(5); // title, desc, price, catSlug

    const selectElement = screen.getByRole('combobox');
    expect(selectElement).toBeInTheDocument();

    const fileInput = screen.getByLabelText('Upload Image');
    expect(fileInput).toBeInTheDocument();

    const submitButton = screen.getByRole('button', { name: 'Submit' });
    expect(submitButton).toBeInTheDocument();
  });

  test('renders error message for empty price field', () => {
    render(<AddPage />);
    const priceInput = screen.getByLabelText('Price');
    const submitButton = screen.getByRole('button', { name: 'Submit' });

    fireEvent.change(priceInput, { target: { value: '' } });
    fireEvent.click(submitButton);

    const errorMessage = screen.getByText('Price');
    expect(errorMessage).toBeInTheDocument();
  });

```


AddPage.test.tsx
src > app > __test__ > AddPage.test.tsx > ...
Codiumate: Add more tests
PROBLEMS
OUTPUT
TERMINAL
PORTS
SEARCH ERROR

```tsx
  });

  test('renders error message for empty category field', () => {
    render(<AddPage />);
    const categorySelect = screen.getByRole('combobox');
    const submitButton = screen.getByRole('button', { name: 'Submit

    fireEvent.change(categorySelect, { target: { value: '' } });
    fireEvent.click(submitButton);

    const errorMessage = screen.getByText('Category');
    expect(errorMessage).toBeInTheDocument();
  });
});
```

```
      at setIngredients (src/app/add/page.tsx:71:9)


PASS  src/app/__test__/AddPage.test.tsx
  AddPage
    √ renders Add New Product heading (281 ms)
    √ renders form elements (296 ms)
    √ renders error message for empty price field (120 ms)
    √ renders error message for empty category field (93 ms)


Test Suites: 1 passed, 1 total
Tests:       4 passed, 4 total
Snapshots:   0 total
Time:        4.604 s
Ran all test suites matching /\\src.app\\__test__\\AddPage.test.tsx/i.

Active Filters: filename /\\src.app\\__test__\\AddPage.test.tsx/
 › Press c to clear filters.
```

CART PAGE:

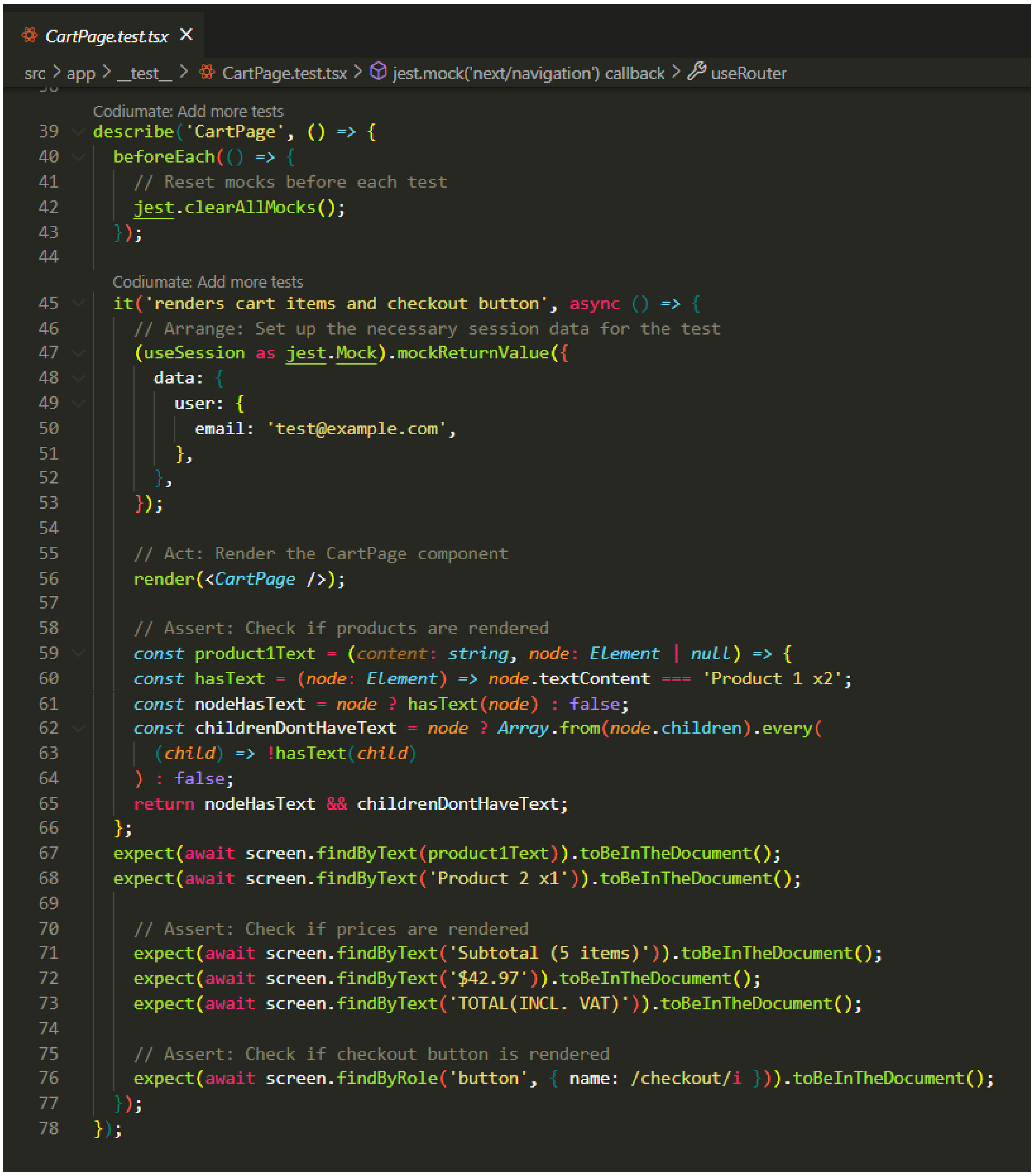


CartPage.test.tsx

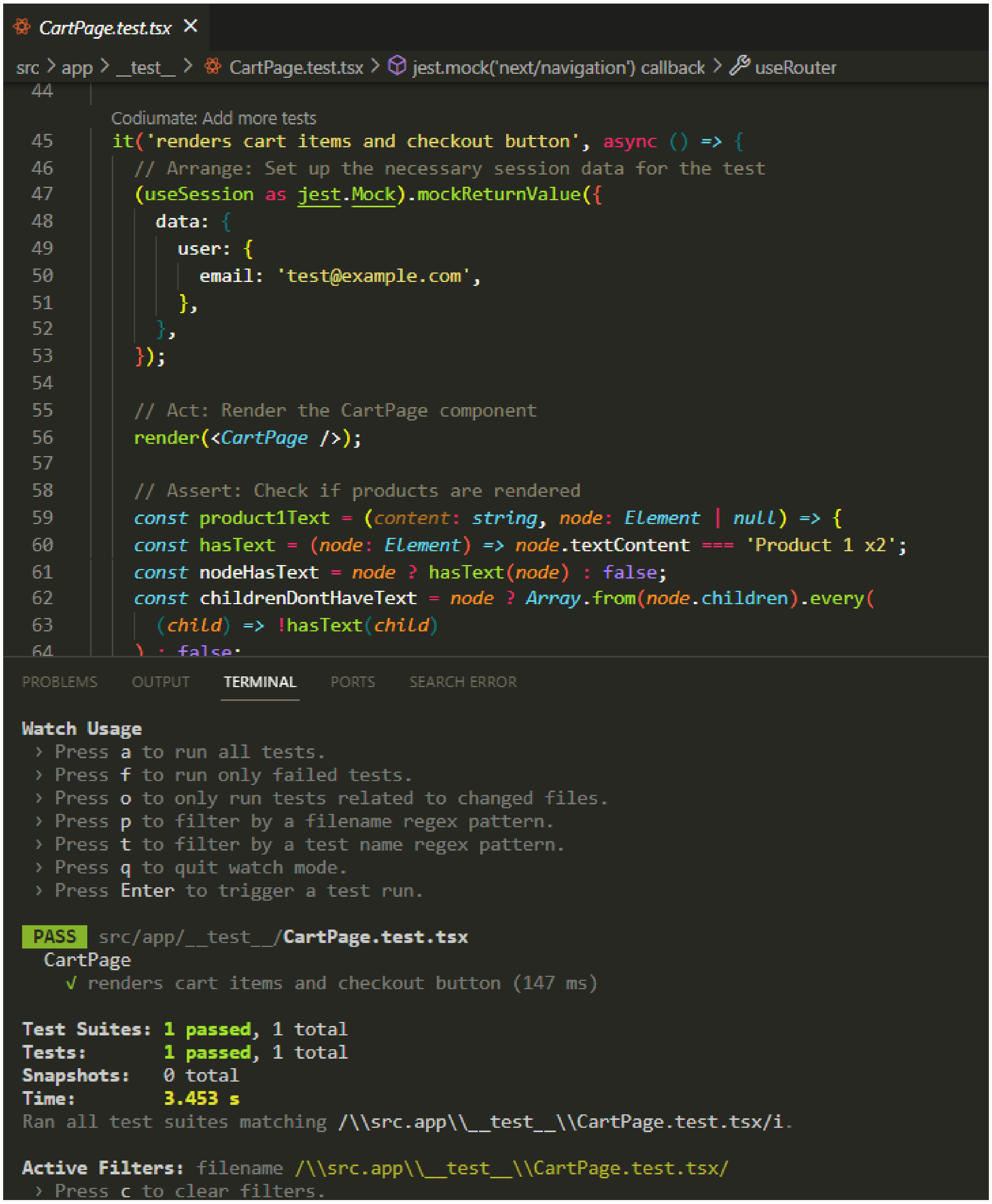
src > app > __test__ > CartPage.test.tsx > jest.mock('next/navigation') callback > useRouter

Click here to ask Blackbox to help you code faster

```
/**
 * @jest-environment jsdom
 */

import { render, screen } from '@testing-library/react';
import CartPage from '../cart/page';
import { useSession } from 'next-auth/react';

// Mock the useRouter hook using 'next/navigation'
jest.mock('next/navigation', () => ({
  useRouter: jest.fn().mockReturnValue({
    push: jest.fn(),
  }),
}));

// Mock the useSession hook
jest.mock('next-auth/react', () => ({
  useSession: jest.fn(),
}));

// Mock the useCartStore hook with persist and rehydrate
jest.mock('../../utils/store', () => ({
  useCartStore: jest.fn(() => ({
    products: [
      { id: '1', title: 'Product 1', quantity: 2, price: '20.00', img: '/images/product1.jpg' },
      { id: '2', title: 'Product 2', quantity: 1, price: '22.97', img: '/images/product2.jpg' },
      // Add more products as needed for testing
    ],
    totalItems: 5,
    totalPrice: 42.97,
    removeFromCart: jest.fn(),
    // Mock the persist object with a no-op rehydrate function
    persist: {
      rehydrate: jest.fn(), // This should be a no-op function
    },
  })),
}));

Codiumate: Add more tests
describe('CartPage', () => {
  beforeEach(() => {
    // Reset mocks before each test
    jest.clearAllMocks();
  });
```


CartPage.test.tsx
src > app > __test__ > CartPage.test.tsx > jest.mock('next/navigation') callback > useRouter
Codiumate: Add more tests

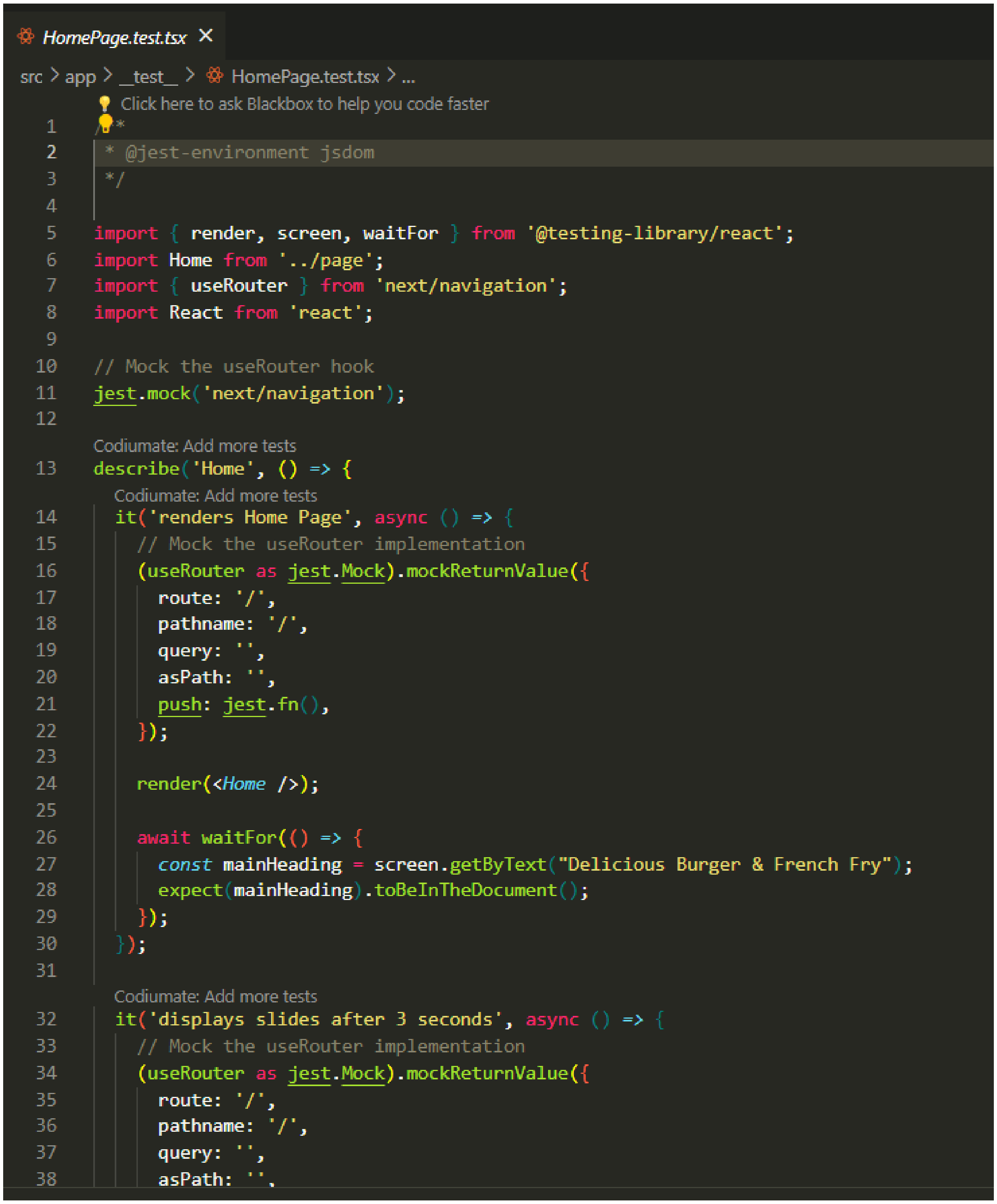
```tsx
describe('CartPage', () => {
  beforeEach(() => {
    // Reset mocks before each test
    jest.clearAllMocks();
  });

  it('renders cart items and checkout button', async () => {
    // Arrange: Set up the necessary session data for the test
    (useSession as jest.Mock).mockReturnValue({
      data: {
        user: {
          email: 'test@example.com',
        },
      },
    });

    // Act: Render the CartPage component
    render(<CartPage />);

    // Assert: Check if products are rendered
    const product1Text = (content: string, node: Element | null) => {
    const hasText = (node: Element) => node.textContent === 'Product 1 x2';
    const nodeHasText = node ? hasText(node) : false;
    const childrenDontHaveText = node ? Array.from(node.children).every(
      (child) => !hasText(child)
    ) : false;
    return nodeHasText && childrenDontHaveText;
  };
  expect(await screen.findByText(product1Text)).toBeInTheDocument();
  expect(await screen.findByText('Product 2 x1')).toBeInTheDocument();

    // Assert: Check if prices are rendered
    expect(await screen.findByText('Subtotal (5 items)')).toBeInTheDocument();
    expect(await screen.findByText('$42.97')).toBeInTheDocument();
    expect(await screen.findByText('TOTAL(INCL. VAT)')).toBeInTheDocument();

    // Assert: Check if checkout button is rendered
    expect(await screen.findByRole('button', { name: /checkout/i })).toBeInTheDocument();
  });
});
```


CartPage.test.tsx
src > app > __test__ > CartPage.test.tsx > jest.mock('next/navigation') callback > useRouter
Codiumate: Add more tests
PROBLEMS OUTPUT TERMINAL PORTS SEARCH ERROR

```
  it('renders cart items and checkout button', async () => {
    // Arrange: Set up the necessary session data for the test
    (useSession as jest.Mock).mockReturnValue({
      data: {
        user: {
          email: 'test@example.com',
        },
      },
    });

    // Act: Render the CartPage component
    render(<CartPage />);

    // Assert: Check if products are rendered
    const product1Text = (content: string, node: Element | null) => {
    const hasText = (node: Element) => node.textContent === 'Product 1 x2';
    const nodeHasText = node ? hasText(node) : false;
    const childrenDontHaveText = node ? Array.from(node.children).every(
      (child) => !hasText(child)
    ) : false;
```

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

 PASS  src/app/__test__/CartPage.test.tsx
  CartPage
    √ renders cart items and checkout button (147 ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        3.453 s
Ran all test suites matching /\\src.app\\__test__\\CartPage.test.tsx/i.

Active Filters: filename /\\src.app\\__test__\\CartPage.test.tsx/
 › Press c to clear filters.
```

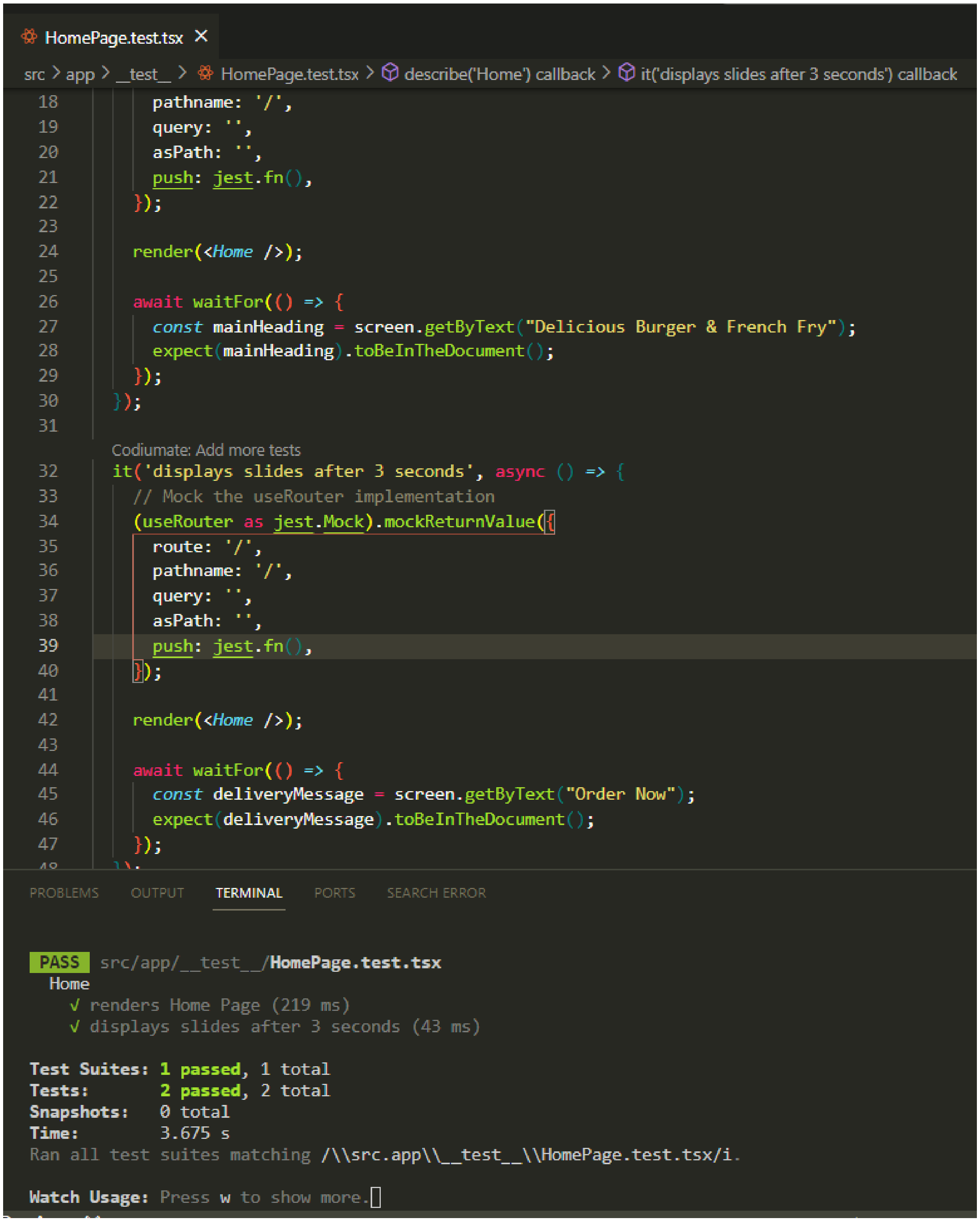
HOME PAGE:


HomePage.test.tsx ×

src > app > _test_ > HomePage.test.tsx > ...

Click here to ask Blackbox to help you code faster

```
/**
 * @jest-environment jsdom
 */

import { render, screen, waitFor } from '@testing-library/react';
import Home from '../page';
import { useRouter } from 'next/navigation';
import React from 'react';

// Mock the useRouter hook
jest.mock('next/navigation');

Codiumate: Add more tests
describe('Home', () => {
  Codiumate: Add more tests
  it('renders Home Page', async () => {
    // Mock the useRouter implementation
    (useRouter as jest.Mock).mockReturnValue({
      route: '/',
      pathname: '/',
      query: '',
      asPath: '',
      push: jest.fn(),
    });

    render(<Home />);

    await waitFor(() => {
      const mainHeading = screen.getByText("Delicious Burger & French Fry");
      expect(mainHeading).toBeInTheDocument();
    });
  });

  Codiumate: Add more tests
  it('displays slides after 3 seconds', async () => {
    // Mock the useRouter implementation
    (useRouter as jest.Mock).mockReturnValue({
      route: '/',
      pathname: '/',
      query: '',
      asPath: '',
```



HomePage.test.tsx ×

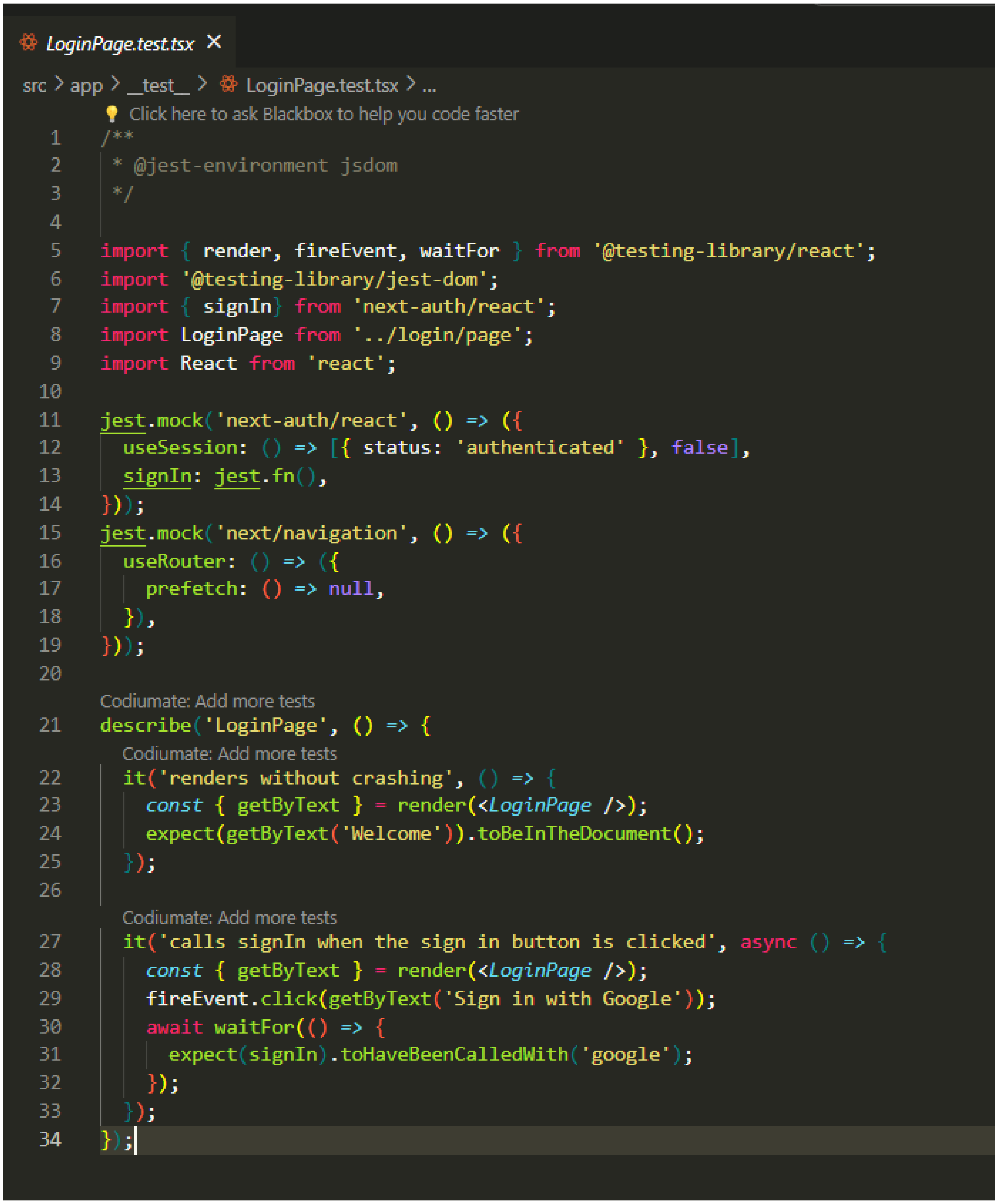
src > app > __test__ > HomePage.test.tsx > describe('Home') callback > it('displays slides after 3 seconds') callback

```
      pathname: '/',
      query: '',
      asPath: '',
      push: jest.fn(),
    });

    render(<Home />);

    await waitFor(() => {
      const mainHeading = screen.getByText("Delicious Burger & French Fry");
      expect(mainHeading).toBeInTheDocument();
    });
  });

  Codiumate: Add more tests
  it('displays slides after 3 seconds', async () => {
    // Mock the useRouter implementation
    (useRouter as jest.Mock).mockReturnValue({
      route: '/',
      pathname: '/',
      query: '',
      asPath: '',
      push: jest.fn(),
    });

    render(<Home />);

    await waitFor(() => {
      const deliveryMessage = screen.getByText("Order Now");
      expect(deliveryMessage).toBeInTheDocument();
    });
```

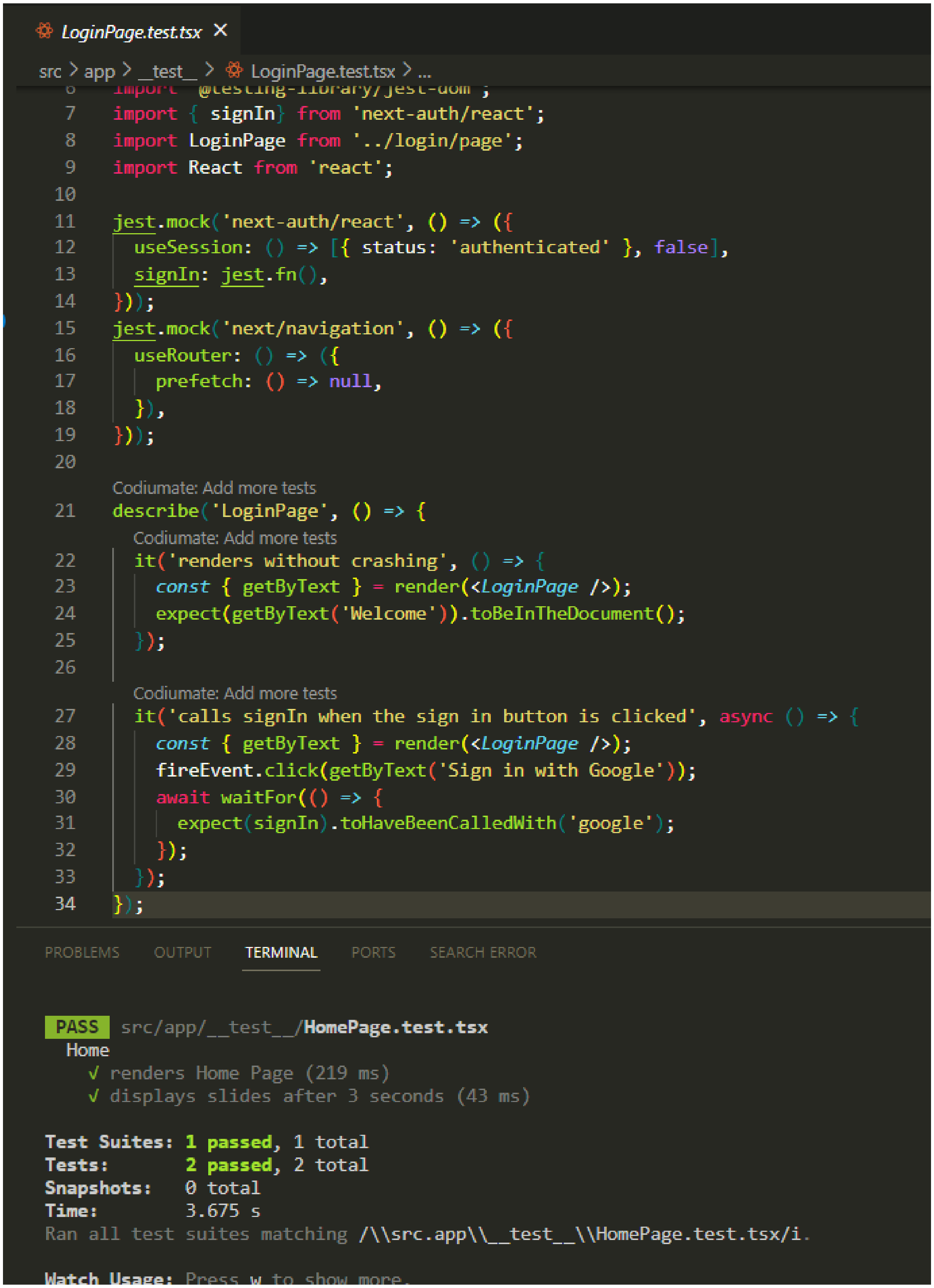
PROBLEMS   OUTPUT   TERMINAL   PORTS   SEARCH ERROR

```
PASS  src/app/__test__/HomePage.test.tsx
  Home
    √ renders Home Page (219 ms)
    √ displays slides after 3 seconds (43 ms)

Test Suites: 1 passed, 1 total
Tests:       2 passed, 2 total
Snapshots:   0 total
Time:        3.675 s
Ran all test suites matching /\\src.app\\__test__\\HomePage.test.tsx/i.

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

LOGIN PAGE:


LoginPage.test.tsx
src > app > __test__ > LoginPage.test.tsx > ...
Click here to ask Blackbox to help you code faster
Codiumate: Add more tests

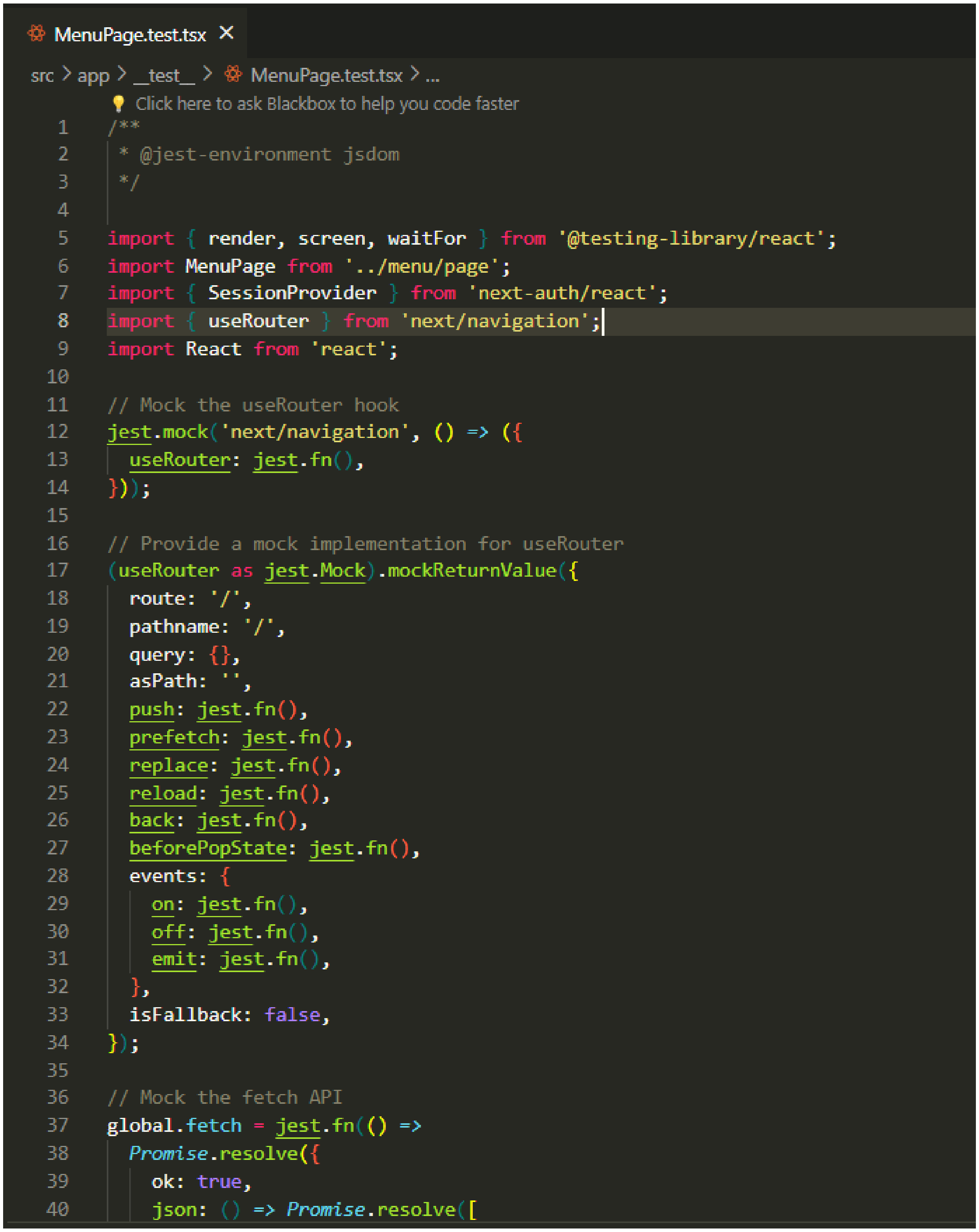
```tsx
/**
 * @jest-environment jsdom
 */

import { render, fireEvent, waitFor } from '@testing-library/react';
import '@testing-library/jest-dom';
import { signIn} from 'next-auth/react';
import LoginPage from '../login/page';
import React from 'react';

jest.mock('next-auth/react', () => ({
  useSession: () => [{ status: 'authenticated' }, false],
  signIn: jest.fn(),
}));
jest.mock('next/navigation', () => ({
  useRouter: () => ({
    prefetch: () => null,
  }),
}));

describe('LoginPage', () => {
  it('renders without crashing', () => {
    const { getByText } = render(<LoginPage />);
    expect(getByText('Welcome')).toBeInTheDocument();
  });

  it('calls signIn when the sign in button is clicked', async () => {
    const { getByText } = render(<LoginPage />);
    fireEvent.click(getByText('Sign in with Google'));
    await waitFor(() => {
      expect(signIn).toHaveBeenCalledWith('google');
    });
  });
});
```


LoginPage.test.tsx
src > app > __test__ > LoginPage.test.tsx > ...
Codiumate: Add more tests
PROBLEMS OUTPUT TERMINAL PORTS SEARCH ERROR

```tsx
import '@testing-library/jest-dom';
import { signIn } from 'next-auth/react';
import LoginPage from '../login/page';
import React from 'react';

jest.mock('next-auth/react', () => ({
  useSession: () => [{ status: 'authenticated' }, false],
  signIn: jest.fn(),
}));
jest.mock('next/navigation', () => ({
  useRouter: () => ({
    prefetch: () => null,
  }),
}));

describe('LoginPage', () => {
  it('renders without crashing', () => {
    const { getByText } = render(<LoginPage />);
    expect(getByText('Welcome')).toBeInTheDocument();
  });

  it('calls signIn when the sign in button is clicked', async () => {
    const { getByText } = render(<LoginPage />);
    fireEvent.click(getByText('Sign in with Google'));
    await waitFor(() => {
      expect(signIn).toHaveBeenCalledWith('google');
    });
  });
});
```

```
PASS  src/app/__test__/HomePage.test.tsx
  Home
    √ renders Home Page (219 ms)
    √ displays slides after 3 seconds (43 ms)

Test Suites: 1 passed, 1 total
Tests:       2 passed, 2 total
Snapshots:   0 total
Time:        3.675 s
Ran all test suites matching /\\src.app\\__test__\\HomePage.test.tsx/i.

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

MENU PAGE:

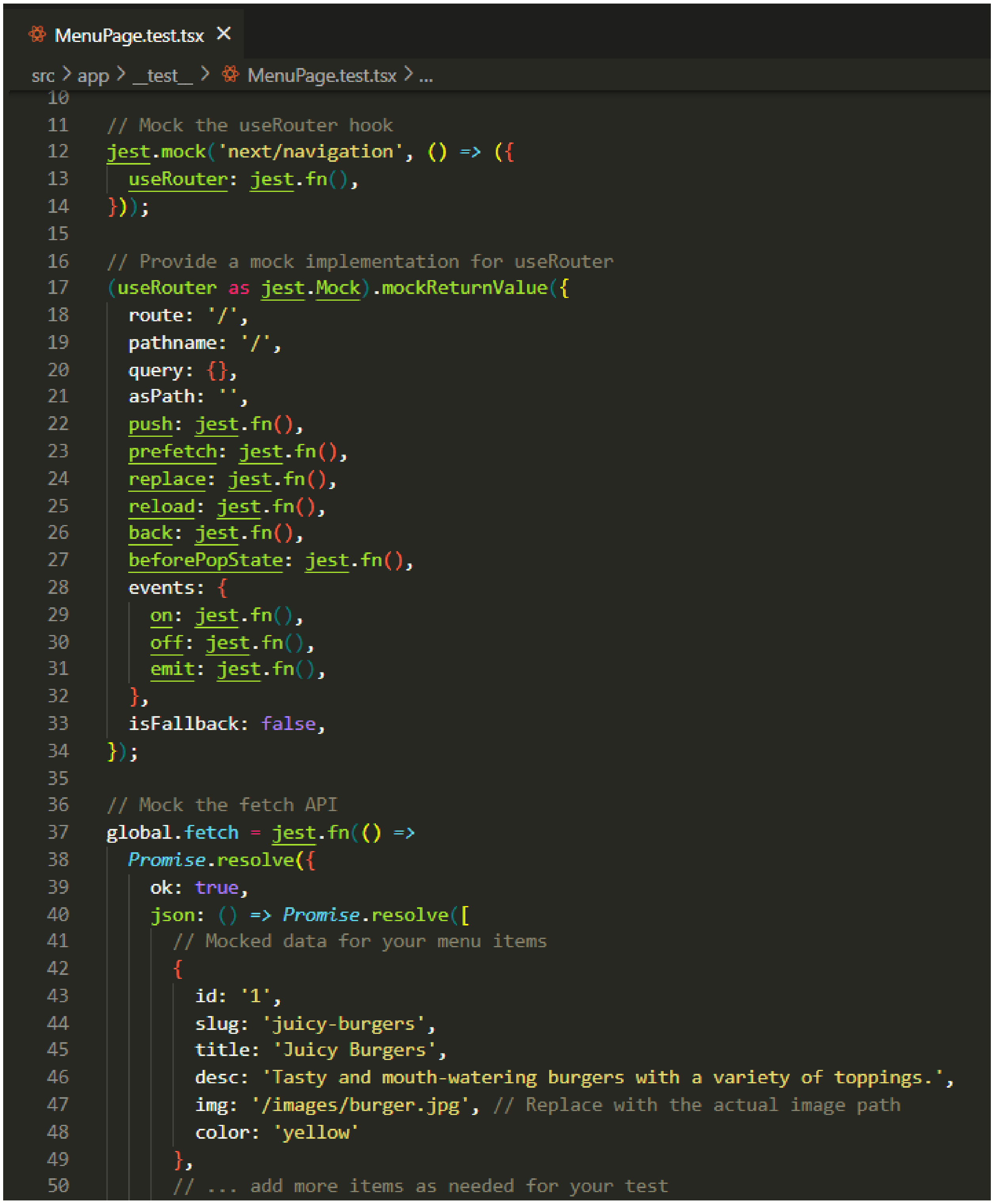


MenuPage.test.tsx ×

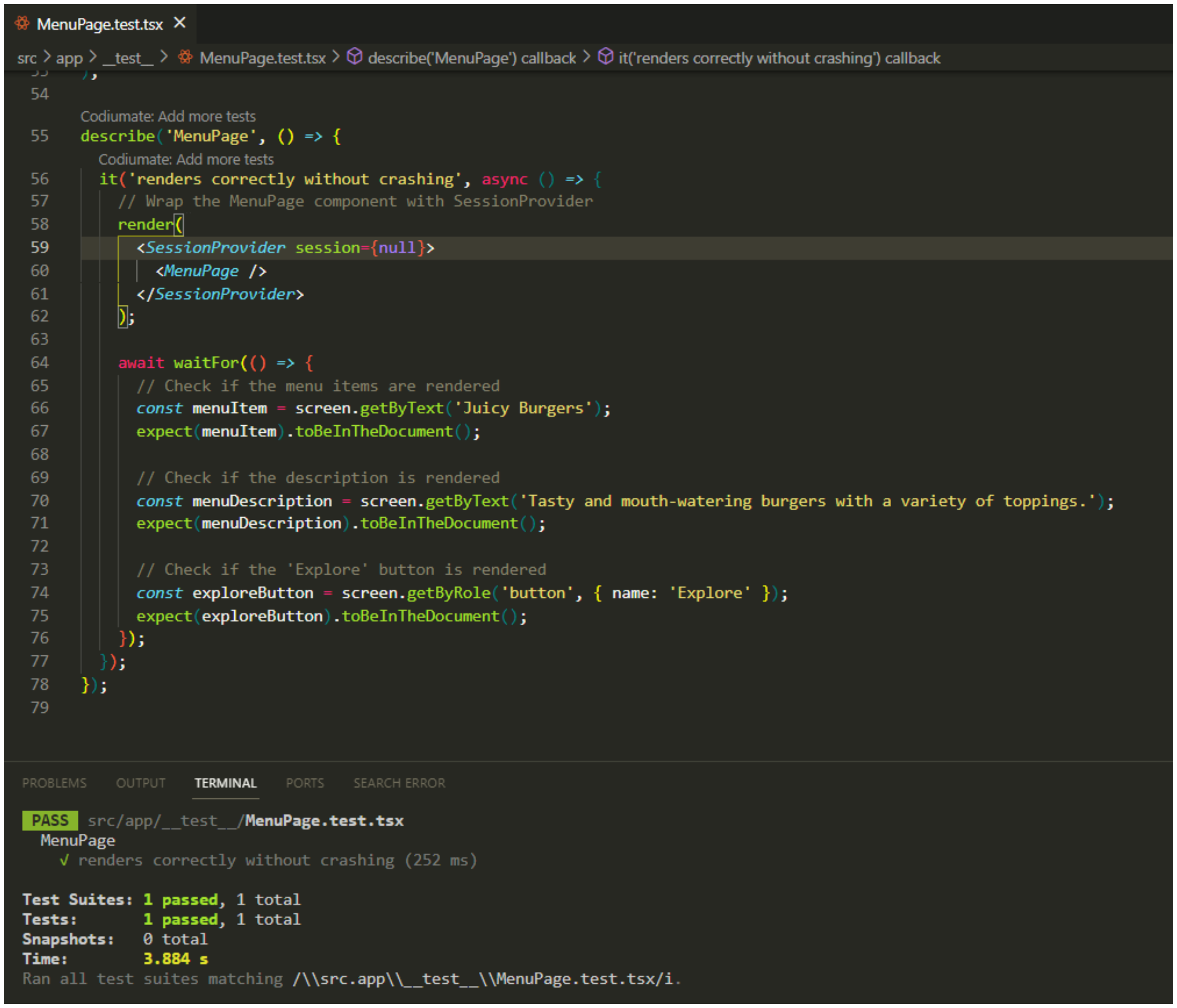
src > app > _test_ > MenuPage.test.tsx > ...

Click here to ask Blackbox to help you code faster

```
/**
 * @jest-environment jsdom
 */

import { render, screen, waitFor } from '@testing-library/react';
import MenuPage from '../menu/page';
import { SessionProvider } from 'next-auth/react';
import { useRouter } from 'next/navigation';
import React from 'react';

// Mock the useRouter hook
jest.mock('next/navigation', () => ({
  useRouter: jest.fn(),
}));

// Provide a mock implementation for useRouter
(useRouter as jest.Mock).mockReturnValue({
  route: '/',
  pathname: '/',
  query: {},
  asPath: '',
  push: jest.fn(),
  prefetch: jest.fn(),
  replace: jest.fn(),
  reload: jest.fn(),
  back: jest.fn(),
  beforePopState: jest.fn(),
  events: {
    on: jest.fn(),
    off: jest.fn(),
    emit: jest.fn(),
  },
  isFallback: false,
});

// Mock the fetch API
global.fetch = jest.fn(() =>
  Promise.resolve({
    ok: true,
    json: () => Promise.resolve([
```



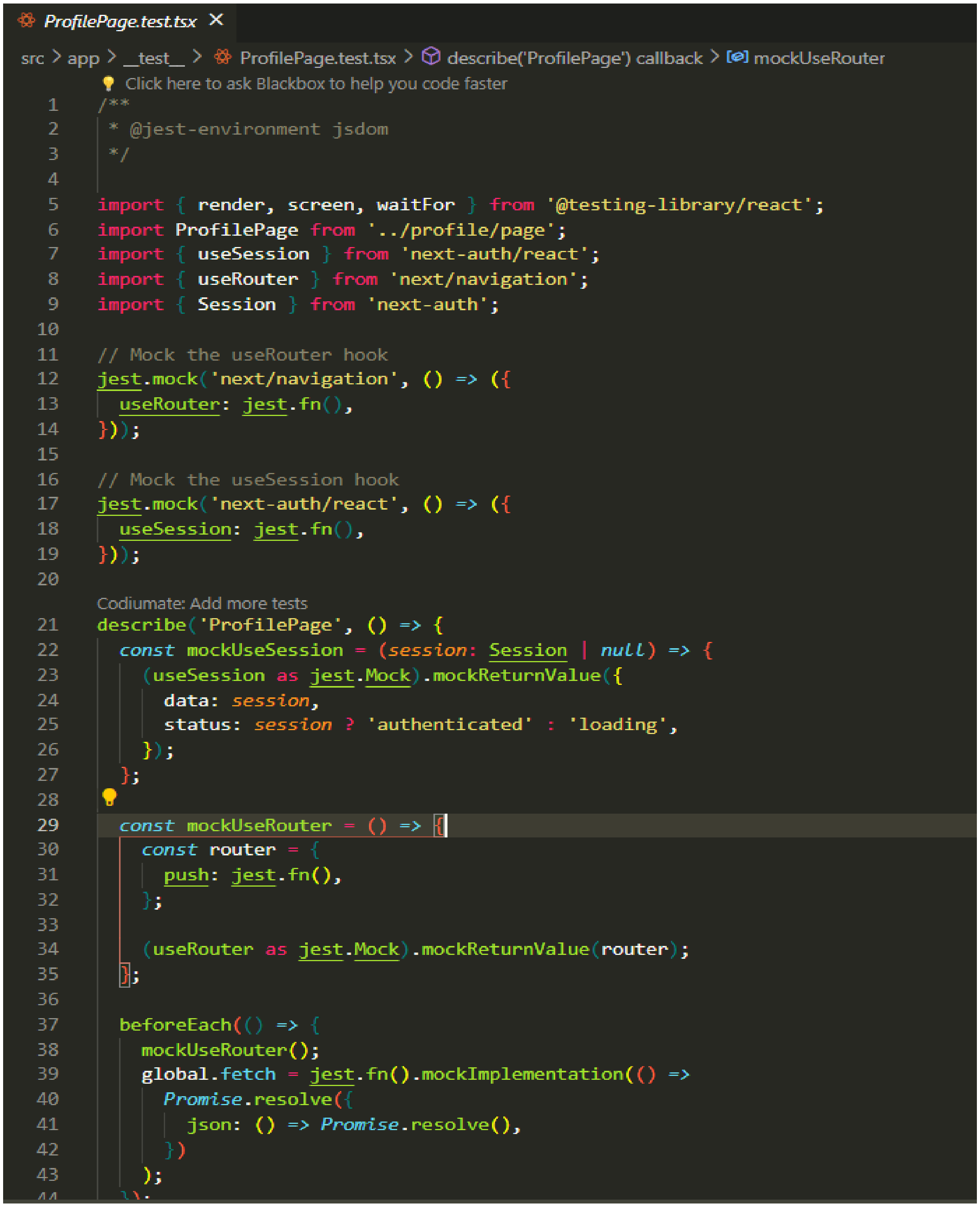
```tsx

// Mock the useRouter hook
jest.mock('next/navigation', () => ({
  useRouter: jest.fn(),
}));

// Provide a mock implementation for useRouter
(useRouter as jest.Mock).mockReturnValue({
  route: '/',
  pathname: '/',
  query: {},
  asPath: '',
  push: jest.fn(),
  prefetch: jest.fn(),
  replace: jest.fn(),
  reload: jest.fn(),
  back: jest.fn(),
  beforePopState: jest.fn(),
  events: {
    on: jest.fn(),
    off: jest.fn(),
    emit: jest.fn(),
  },
  isFallback: false,
});

// Mock the fetch API
global.fetch = jest.fn(() =>
  Promise.resolve({
    ok: true,
    json: () => Promise.resolve([
      // Mocked data for your menu items
      {
        id: '1',
        slug: 'juicy-burgers',
        title: 'Juicy Burgers',
        desc: 'Tasty and mouth-watering burgers with a variety of toppings.',
        img: '/images/burger.jpg', // Replace with the actual image path
        color: 'yellow'
      },
      // ... add more items as needed for your test
```


MenuPage.test.tsx
src > app > _test_ > MenuPage.test.tsx > describe('MenuPage') callback > it('renders correctly without crashing') callback
Codiumate: Add more tests
PROBLEMS OUTPUT TERMINAL PORTS SEARCH ERROR

```tsx
describe('MenuPage', () => {
  it('renders correctly without crashing', async () => {
    // Wrap the MenuPage component with SessionProvider
    render(
      <SessionProvider session={null}>
        <MenuPage />
      </SessionProvider>
    );

    await waitFor(() => {
      // Check if the menu items are rendered
      const menuItem = screen.getByText('Juicy Burgers');
      expect(menuItem).toBeInTheDocument();

      // Check if the description is rendered
      const menuDescription = screen.getByText('Tasty and mouth-watering burgers with a variety of toppings.');
      expect(menuDescription).toBeInTheDocument();

      // Check if the 'Explore' button is rendered
      const exploreButton = screen.getByRole('button', { name: 'Explore' });
      expect(exploreButton).toBeInTheDocument();
    });
  });
});
```

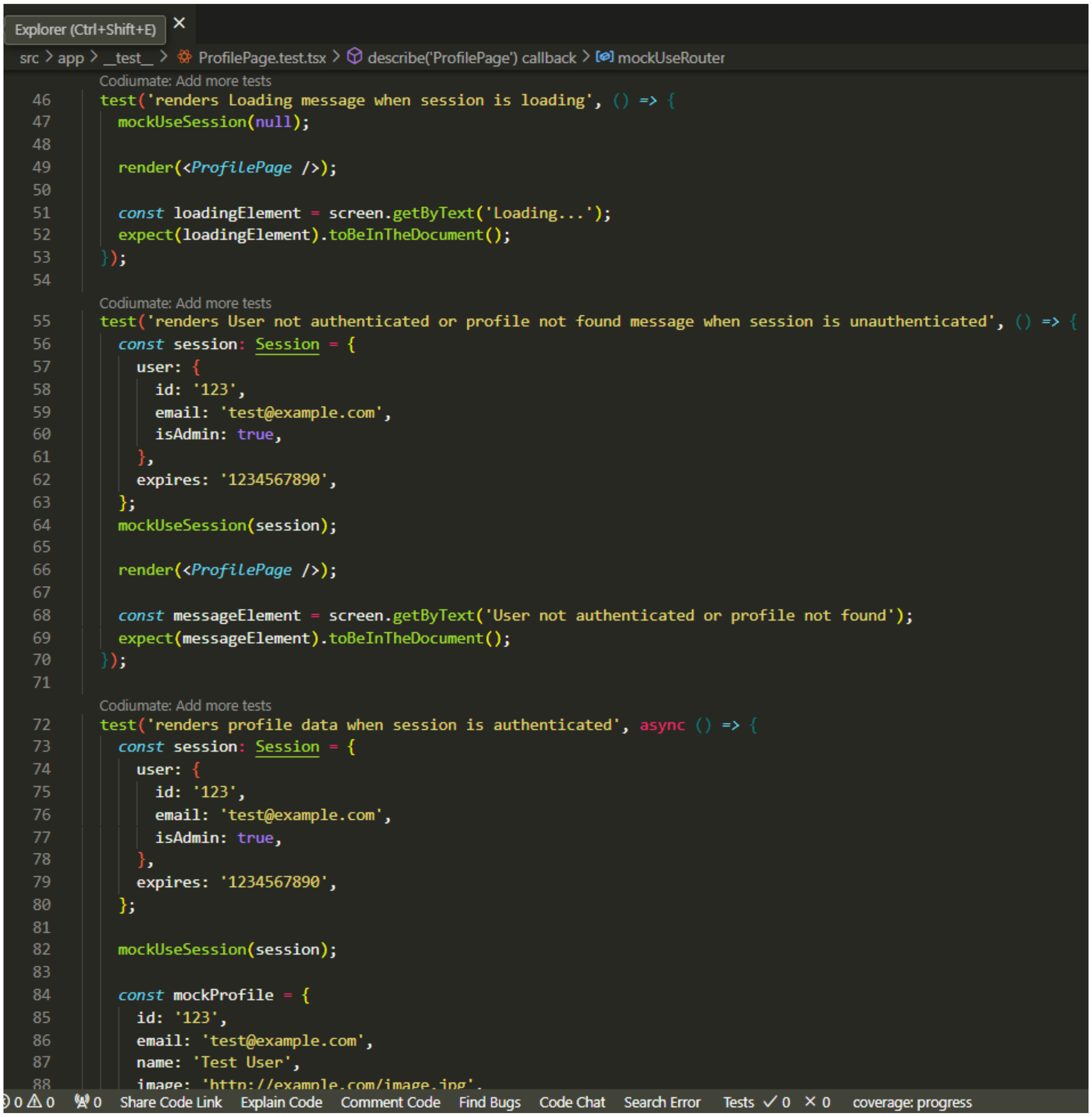
```
PASS  src/app/__test__/MenuPage.test.tsx
  MenuPage
    √ renders correctly without crashing (252 ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        3.884 s
Ran all test suites matching /\\src.app\\__test__\\MenuPage.test.tsx/i.
```

PROFILE PAGE:



ProfilePage.test.tsx ×
src > app > __test__ > ProfilePage.test.tsx > describe('ProfilePage') callback > mockUseRouter
Click here to ask Blackbox to help you code faster
Codiumate: Add more tests

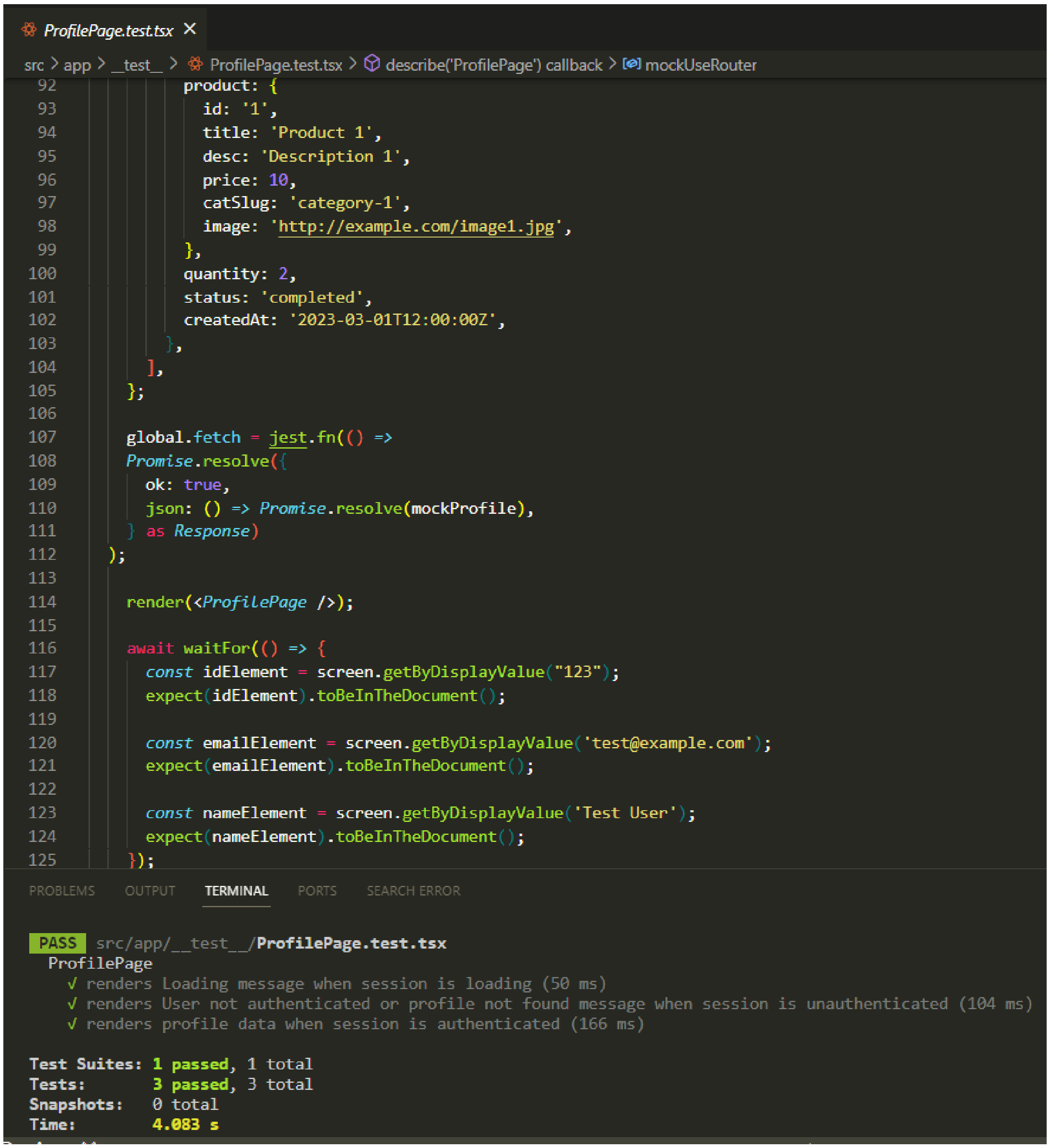
```
/**
 * @jest-environment jsdom
 */

import { render, screen, waitFor } from '@testing-library/react';
import ProfilePage from '../profile/page';
import { useSession } from 'next-auth/react';
import { useRouter } from 'next/navigation';
import { Session } from 'next-auth';

// Mock the useRouter hook
jest.mock('next/navigation', () => ({
  useRouter: jest.fn(),
}));

// Mock the useSession hook
jest.mock('next-auth/react', () => ({
  useSession: jest.fn(),
}));

describe('ProfilePage', () => {
  const mockUseSession = (session: Session | null) => {
    (useSession as jest.Mock).mockReturnValue({
      data: session,
      status: session ? 'authenticated' : 'loading',
    });
  };

  const mockUseRouter = () => {
    const router = {
      push: jest.fn(),
    };

    (useRouter as jest.Mock).mockReturnValue(router);
  };

  beforeEach(() => {
    mockUseRouter();
    global.fetch = jest.fn().mockImplementation(() =>
      Promise.resolve({
        json: () => Promise.resolve(),
      })
    );
```



Explorer (Ctrl+Shift+E)
src > app > _test_ > ProfilePage.test.tsx > describe('ProfilePage') callback > mockUseRouter

```
Codiumate: Add more tests
test('renders Loading message when session is loading', () => {
  mockUseSession(null);

  render(<ProfilePage />);

  const loadingElement = screen.getByText('Loading...');
  expect(loadingElement).toBeInTheDocument();
});

Codiumate: Add more tests
test('renders User not authenticated or profile not found message when session is unauthenticated', () => {
  const session: Session = {
    user: {
      id: '123',
      email: 'test@example.com',
      isAdmin: true,
    },
    expires: '1234567890',
  };
  mockUseSession(session);

  render(<ProfilePage />);

  const messageElement = screen.getByText('User not authenticated or profile not found');
  expect(messageElement).toBeInTheDocument();
});

Codiumate: Add more tests
test('renders profile data when session is authenticated', async () => {
  const session: Session = {
    user: {
      id: '123',
      email: 'test@example.com',
      isAdmin: true,
    },
    expires: '1234567890',
  };

  mockUseSession(session);

  const mockProfile = {
    id: '123',
    email: 'test@example.com',
    name: 'Test User',
    image: 'http://example.com/image.jpg',
```

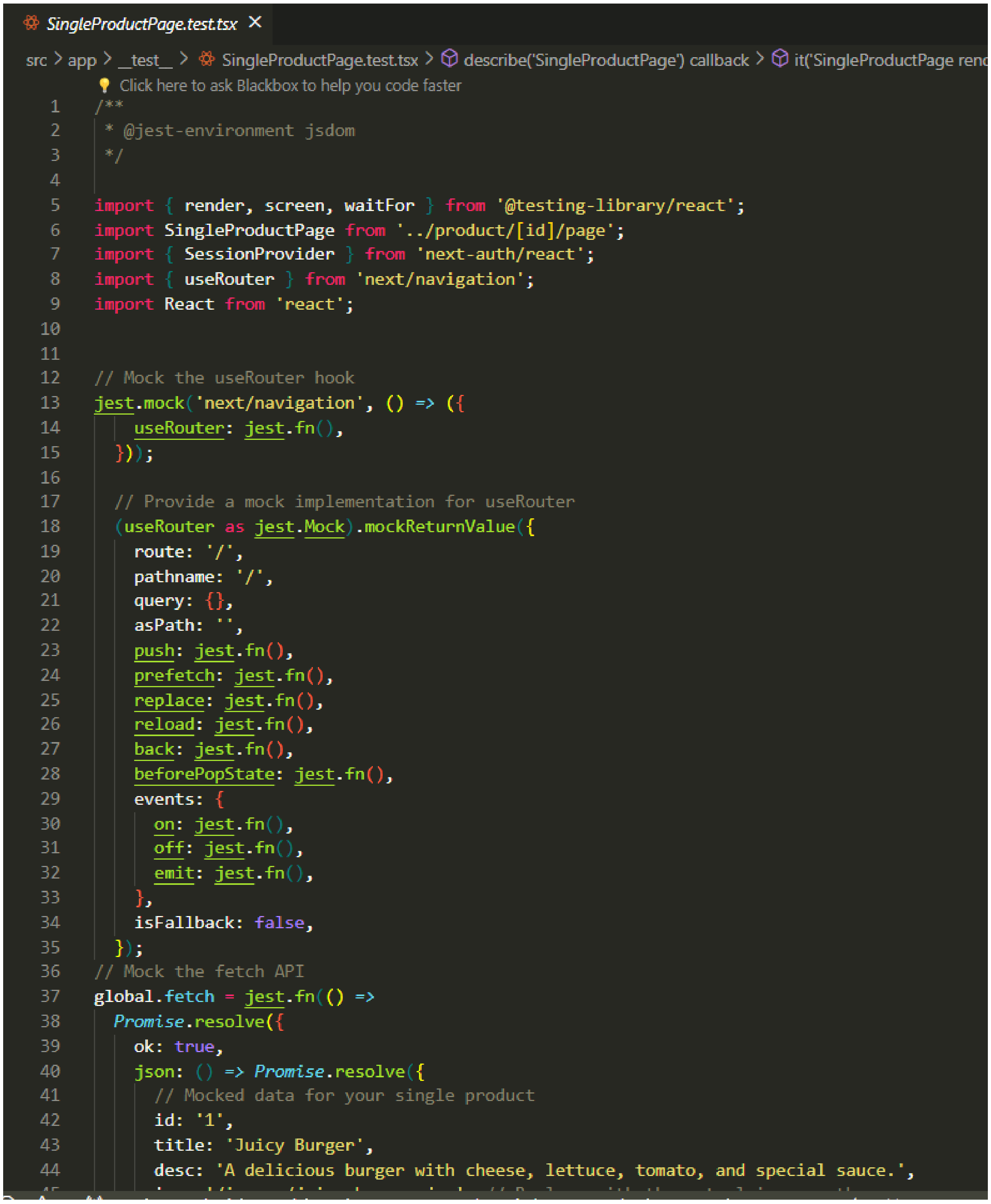
0 ⚠ 0 0 Share Code Link Explain Code Comment Code Find Bugs Code Chat Search Error Tests ✓ 0 ✕ 0 coverage: progress


ProfilePage.test.tsx
src > app > _test_ > ProfilePage.test.tsx > describe('ProfilePage') callback > mockUseRouter
PROBLEMS OUTPUT TERMINAL PORTS SEARCH ERROR

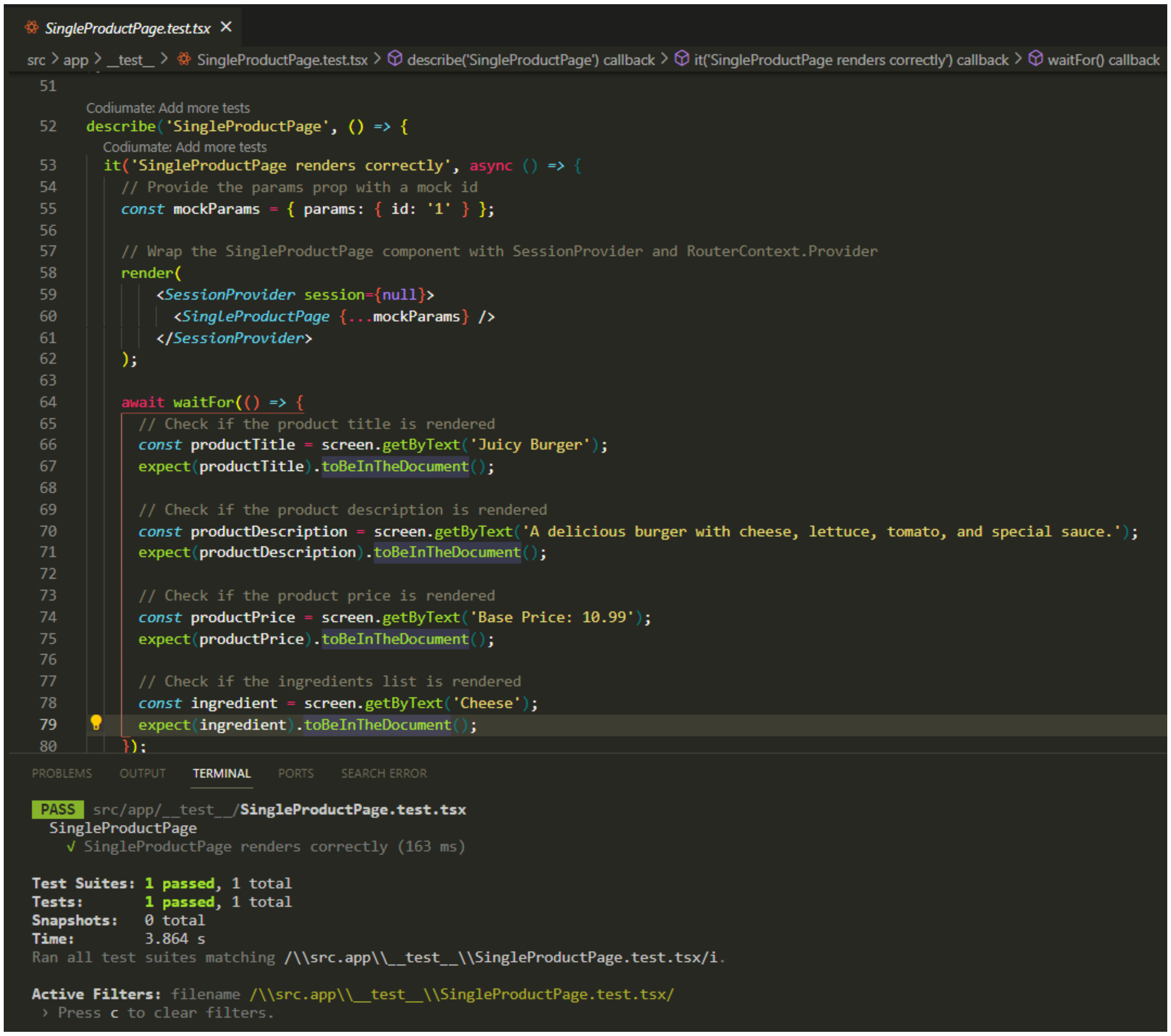
```tsx
          product: {
            id: '1',
            title: 'Product 1',
            desc: 'Description 1',
            price: 10,
            catSlug: 'category-1',
            image: 'http://example.com/image1.jpg',
          },
          quantity: 2,
          status: 'completed',
          createdAt: '2023-03-01T12:00:00Z',
        },
      ],
    };

    global.fetch = jest.fn(() =>
    Promise.resolve({
      ok: true,
      json: () => Promise.resolve(mockProfile),
    } as Response)
  );

    render(<ProfilePage />);

    await waitFor(() => {
      const idElement = screen.getByDisplayValue("123");
      expect(idElement).toBeInTheDocument();

      const emailElement = screen.getByDisplayValue('test@example.com');
      expect(emailElement).toBeInTheDocument();

      const nameElement = screen.getByDisplayValue('Test User');
      expect(nameElement).toBeInTheDocument();
    });
```

```
 PASS  src/app/__test__/ProfilePage.test.tsx
  ProfilePage
    √ renders Loading message when session is loading (50 ms)
    √ renders User not authenticated or profile not found message when session is unauthenticated (104 ms)
    √ renders profile data when session is authenticated (166 ms)

Test Suites: 1 passed, 1 total
Tests:       3 passed, 3 total
Snapshots:   0 total
Time:        4.083 s
```

SINGLE PRODUCT PAGE:


SingleProductPage.test.tsx
src > app > _test_ > SingleProductPage.test.tsx > describe('SingleProductPage') callback > it('SingleProductPage rend
Click here to ask Blackbox to help you code faster

```
/**
 * @jest-environment jsdom
 */

import { render, screen, waitFor } from '@testing-library/react';
import SingleProductPage from '../product/[id]/page';
import { SessionProvider } from 'next-auth/react';
import { useRouter } from 'next/navigation';
import React from 'react';


// Mock the useRouter hook
jest.mock('next/navigation', () => ({
    useRouter: jest.fn(),
  }));

  // Provide a mock implementation for useRouter
  (useRouter as jest.Mock).mockReturnValue({
    route: '/',
    pathname: '/',
    query: {},
    asPath: '',
    push: jest.fn(),
    prefetch: jest.fn(),
    replace: jest.fn(),
    reload: jest.fn(),
    back: jest.fn(),
    beforePopState: jest.fn(),
    events: {
      on: jest.fn(),
      off: jest.fn(),
      emit: jest.fn(),
    },
    isFallback: false,
  });
// Mock the fetch API
global.fetch = jest.fn(() =>
  Promise.resolve({
    ok: true,
    json: () => Promise.resolve({
      // Mocked data for your single product
      id: '1',
      title: 'Juicy Burger',
      desc: 'A delicious burger with cheese, lettuce, tomato, and special sauce.',
```



SingleProductPage.test.tsx

src > app > _test_ > SingleProductPage.test.tsx > describe('SingleProductPage') callback > it('SingleProductPage renders correctly') callback > waitFor() callback

```
Codiumate: Add more tests
describe('SingleProductPage', () => {
  Codiumate: Add more tests
  it('SingleProductPage renders correctly', async () => {
    // Provide the params prop with a mock id
    const mockParams = { params: { id: '1' } };

    // Wrap the SingleProductPage component with SessionProvider and RouterContext.Provider
    render(
      <SessionProvider session={null}>
        <SingleProductPage {...mockParams} />
      </SessionProvider>
    );

    await waitFor(() => {
      // Check if the product title is rendered
      const productTitle = screen.getByText('Juicy Burger');
      expect(productTitle).toBeInTheDocument();

      // Check if the product description is rendered
      const productDescription = screen.getByText('A delicious burger with cheese, lettuce, tomato, and special sauce.');
      expect(productDescription).toBeInTheDocument();

      // Check if the product price is rendered
      const productPrice = screen.getByText('Base Price: 10.99');
      expect(productPrice).toBeInTheDocument();

      // Check if the ingredients list is rendered
      const ingredient = screen.getByText('Cheese');
      expect(ingredient).toBeInTheDocument();
    });
```

PROBLEMS OUTPUT TERMINAL PORTS SEARCH ERROR

```
 PASS  src/app/__test__/SingleProductPage.test.tsx
  SingleProductPage
    √ SingleProductPage renders correctly (163 ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        3.864 s
Ran all test suites matching /\\src.app\\__test__\\SingleProductPage.test.tsx/i.

Active Filters: filename /\\src.app\\__test__\\SingleProductPage.test.tsx/
 › Press c to clear filters.
```

Path Testing
(Restaurant Food Ordering App)

SOME POSSIBLE PATHS FOR A USER COULD BE:

1. Login with Google Cloud -> View Homepage
2. View Homepage -> Click on "View Contacts" -> View Contacts Page
3. View Homepage -> Click on "View Profile" -> View Profile Page
4. View Homepage -> Click on "View Categories" -> Select Category -> View Products in Category Page
5. View Products in Category Page -> Click on Product -> View Product Details Page
6. View Product Details Page -> Click on "Checkout" -> Enter Shipping Information -> Complete Purchase

SOME POSSIBLE PATHS FOR AN ADMIN COULD BE:

1. Admin Logs In -> View Homepage-> Click on "Add Category" -> Add Category Page
2. Admin Logs In -> View Homepage-> Click on "Add Product" -> Add Product Page
3. Admin Logs In -> View Homepage -> Click on "Delete Category" -> Confirm Deletion -> Category Deleted
4. Admin Logs In -> View Homepage -> Click on "Delete Product" -> Confirm Deletion -> Product Deleted
5. Admin Logs In -> View Homepage -> Click on "Manage Orders" -> Manage Orders Page.

PATH TESTING OF INDIVIDUAL PAGES

View Category Page:

Step 1: Draw a control graph

```
                 Start
                   |
                   v
        [getData]---[setMenu]
              |         |
              |         |
              v         v
        [menu.map]     [/]
              |         |
              |         |
              v         v
            [div]        [/]
              |         |
              |         |
              v         v
           [Link]     [DeleteButton]
              |         |
              |         |
              v         v
      [/menu/{slug}]     [/]
```

Step 2: Cyclomatic Complexity:

The cyclomatic complexity of the View Category Page component is 2.

Step 3: Basis Set of Paths:

Based on the cyclomatic complexity, we can identify two independent paths in the code:

1. getData function is called and returns successfully, and setMenu is called with the returned data.
2. getData function is called and throws an error, and the error is caught and handled.

Step 4: Generate test cases to exercise each path

Test Case 1: Verify successful response from getData()

- Input: A valid API endpoint that returns a JSON response with an array of Menu Type objects.
- Expected Output: The Menu Page component should render the list of categories returned from the API.

Test Case 2: Verify error handling for invalid API endpoint

- Input: An invalid API endpoint that returns an error response.
- Expected Output: The Category Page component should handle the error and not break the application.

View Product Page:

Step 1: Draw a control graph

```
                      Start
                        |
                        v
            [getData]---[setSingleProduct]
                        |
                        |
                        v
                    [Loading]
                        |
                        v
        [IMAGE CONTAINER]---[TEXT CONTAINER]
                  |              |
                  |              v
                  v              v
               [Image]      [Title, DeleteButton]
                  |              |
                  |              v
                  v              v
                  |      [Description, Base Price, Ingredients, Price]
                  |              |
                  |              v
                  v              /
              [session.isAdmin check]
```

Step 2: Cyclomatic Complexity:

The cyclomatic complexity of the Product Page component is 1 as there are no decision points in the code.

Step 3: Basis Set of Paths:

Since there's only one path, there's no need to identify independent paths

Step 4: Generate test cases

Test Case: Verify Component Rendering

- **Input:** A valid product data is provided.
- **Expected Output:** The Product Page component should render with the product details displayed correctly.

Delete Category Page:

Step 1: Draw a control graph:

```
                      Start
                        |
                        v
          [useSession]---[useRouter]
               |              |
               |              v
               v              v
           [Loading]          |
               |              |
               |              v
               v              v
     [Unauthenticated]   [isAdmin check]
               |              |
               |              v
               v              v
       [handleDelete]
                        |
                        v
                 [Button Render]
```

Step 2: Cyclomatic Complexity:

The cyclomatic complexity of the Delete Category component is 1.

Step 3: Basis Set of Paths:

Since there's only one path, there's no need to identify independent paths.

Step 4: Generate test cases

Test Case: Verify Button Functionality

- **Input:** A valid category ID.
- **Expected Output:** Clicking the delete button should trigger the deletion process. Upon successful deletion, a success message should be displayed.

Delete Product Page:

Step 1: Draw a control graph:

```
                 Start
                   |
                   v
        [useSession]---[useRouter]
             |              |
             |              |
             v              |
         [Loading]          |
             |              |
             |              |
             v              v
    [Unauthenticated]  [isAdmin check]
             |              |
             |              |
             v              v
       [handleDelete]    [null]
                   |
                   v
             [Button Render]
```

Step 2: Cyclomatic Complexity:

The cyclomatic complexity of the Delete Product component is 1,

Step 3: Basis Set of Paths:

Since there's only one path, there's no need to identify independent paths

Step 4: Generate test cases:

Test Case: Verify Button Functionality

- **Input:** A valid product ID.
- **Expected Output:** Clicking the delete button should trigger the deletion process. Upon successful deletion, a success message should be displayed and the user should be redirected to the menu page. If there's an error, an error message should be displayed.

Add Product Page:

Step 1: Draw a control graph:

```
                         Start
                           |
                           v
          [fetchCategories]---[fetchIngredients]
                  |                  |
                  |                  |
                  v                  v
        [handleChange]     [handleCategoryChange]
                  |                  |
                  |                  |
                  v                  v
         [handleChangeImg]     [changeOption]
                  |                  |
                  |                  |
                  v                  v
       [handleIngredientChange]   [addOption]
                  |                  |
                  |                  |
                  v                  v
       [handleAddNewIngredient] [removeOption]
                  |                  |
                  |                  |
                  v                  v
           [handleSubmit]      [handleSearch]
                           |
                           v
                    [Button Render]
```

Step 2: Cyclomatic Complexity:

The cyclomatic complexity of the Add Product Page component is 1.

Step 3: Basis Set of Paths:

Since there's only one path, there's no need to identify independent paths.

Step 4: Generate test cases:

Test Case: Verify Form Submission

- **Input:** Valid product data is provided.
- **Expected Output:** Clicking the submit button should trigger the form submission process. Upon successful submission, the user should be redirected to the product page.

Add Category Page:

Step 1: Draw a control graph:

```
                     Start
                       |
                       v
      [handleChange]---[handleChangeImg]
             |               |
             |               |
             v               v
       [handleSubmit]     [null]
                       |
                       v
                 [Button Render]
```

Step 2: Cyclomatic Complexity:

The cyclomatic complexity of the Add Category Page component is 1.

Step 3: Basis Set of Paths:

Since there's only one path, there's no need to identify independent paths.

Step 4: Generate test cases:

Test Case: Verify Form Submission

- **Input:** Valid category data is provided.
- **Expected Output:** Clicking the submit button should trigger the form submission process. Upon successful submission, the user should be redirected to the home page.

<u>Add To Cart Page:</u>

Step 1: Draw a control graph:

```
Start
  |
  v
[useEffect1]
  |
  v
[useEffect2]
  |
  v
[handleCheckout]
  |
  v
[Button Render]
```

Step 2: Cyclomatic Complexity:

The cyclomatic complexity of the Add to Cart Page component is 1.

Step 3: Basis Set of Paths:

Since there's only one path, there's no need to identify independent paths.

Step 4: Generate test cases:

<u>Test Case: Verify Checkout Process</u>

- **Input:** Clicking the checkout button.
- **Expected Output:** If the user is not logged in, they should be redirected to the login page. If the user is logged in, the checkout process should proceed, and the user should be redirected to the payment page.