**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.

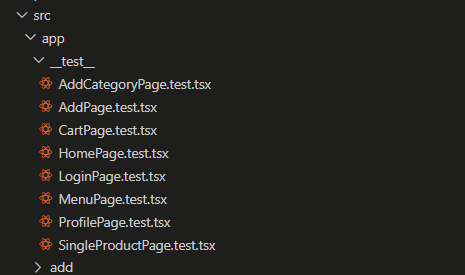
**Test Case Table:**

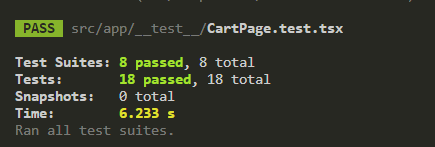
| **Test Case ID** | **Test Case Description** | **Input/Action** | **Expected Output** | **Actual Output** | **Pass/Fail** |
| --- | --- | --- | --- | --- | --- |
| 10.1 | Order button testing | Click on button | View order id, date, price, quantity, and status | View order id, date, price, quantity, 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:**

****

****

**ADD CATEGORY PAGE:**

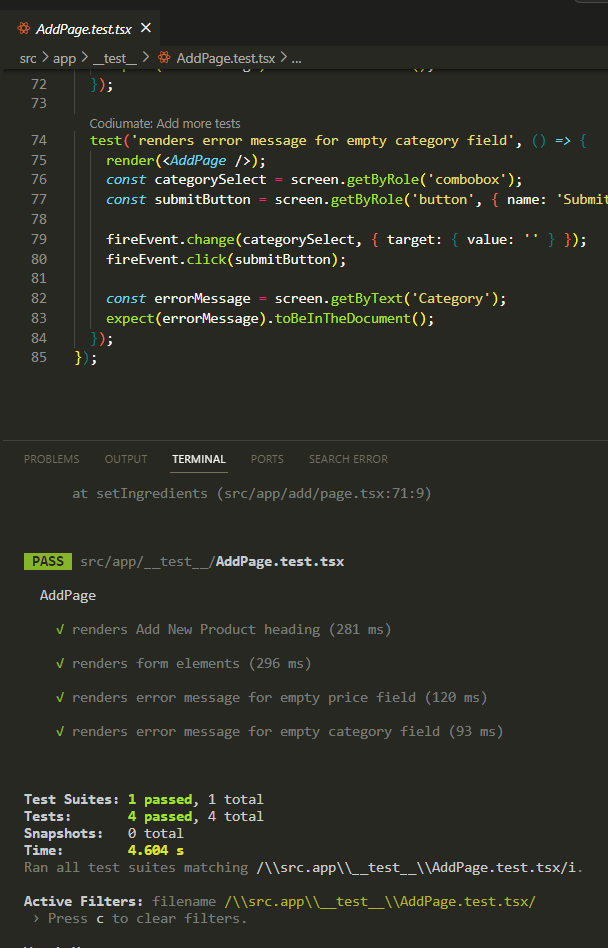
****

****

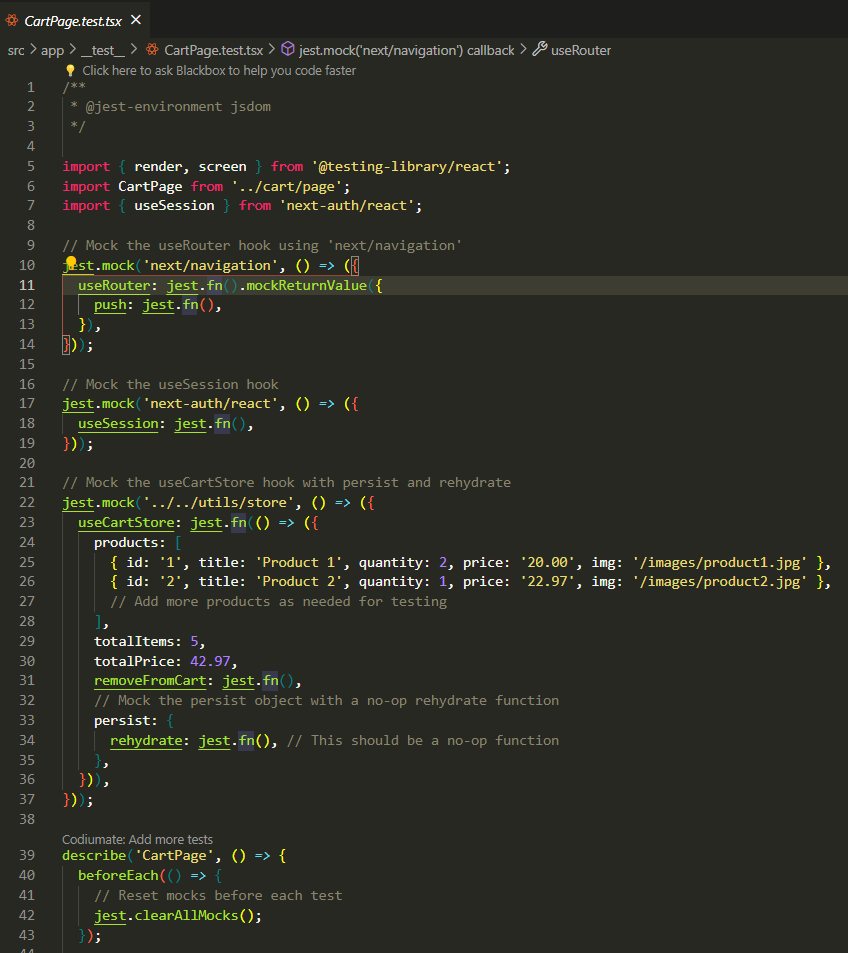
****

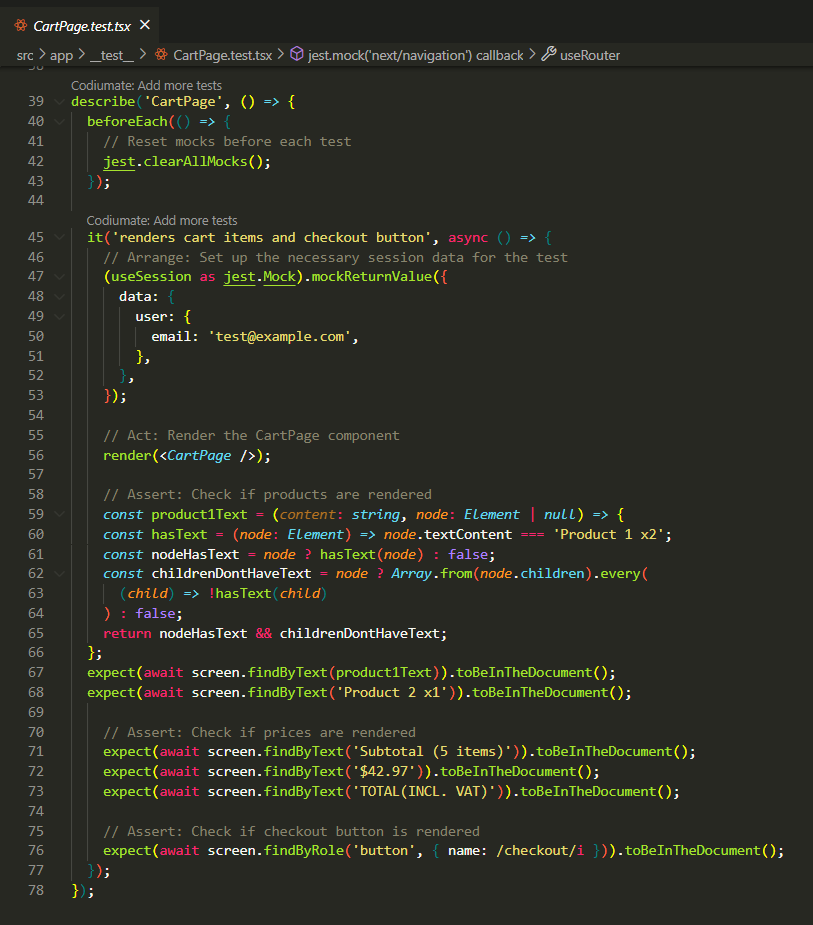
**ADD PRODUCT PAGE:**

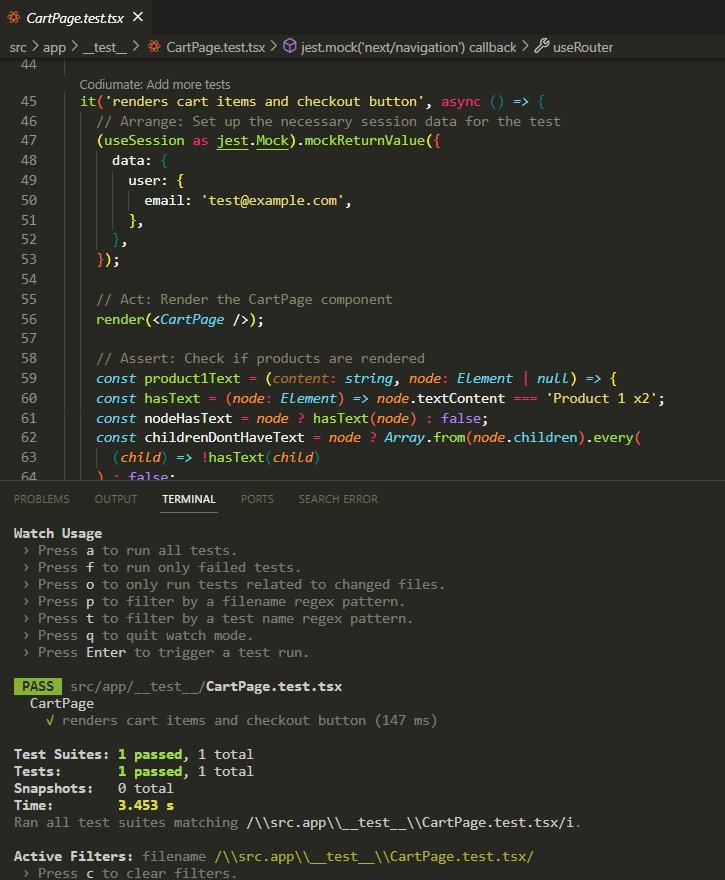
****

****

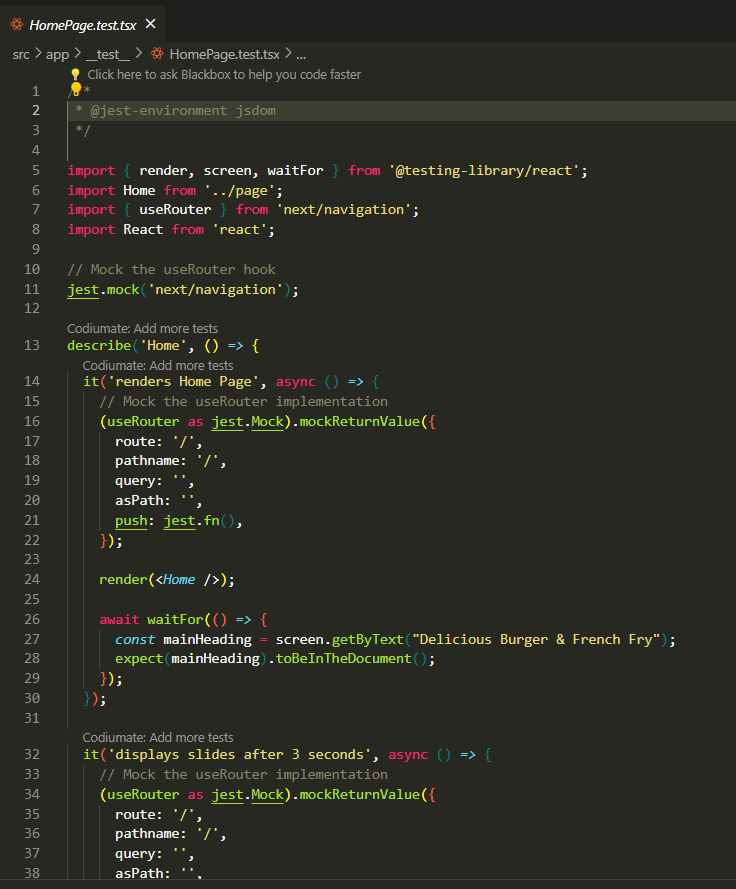
**CART PAGE:**

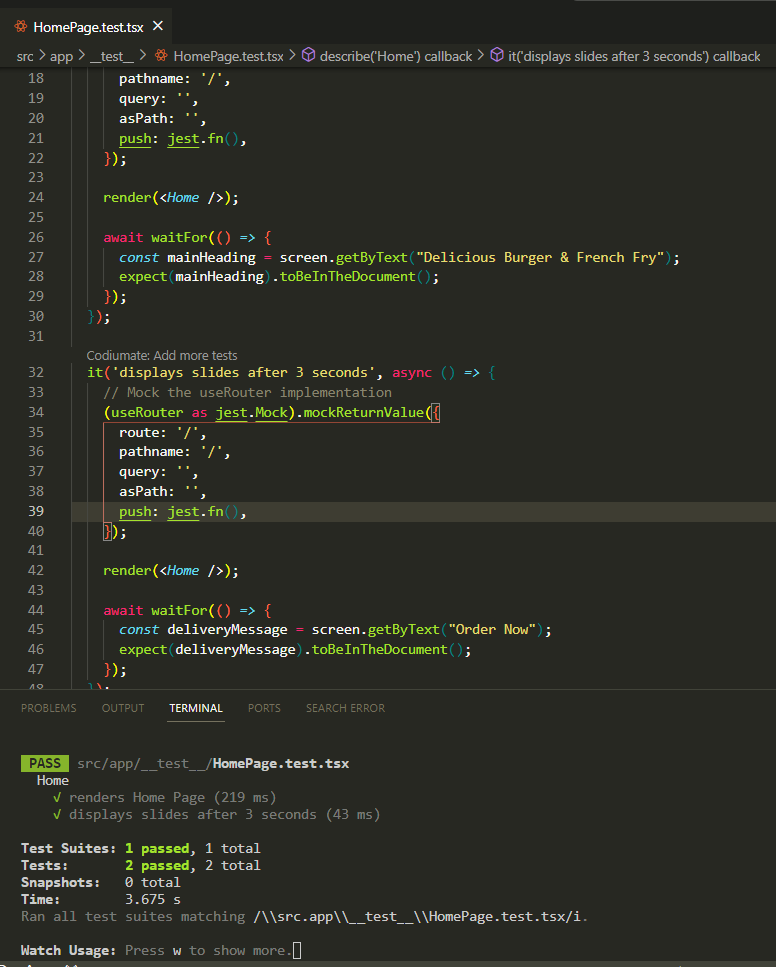
****

****

****

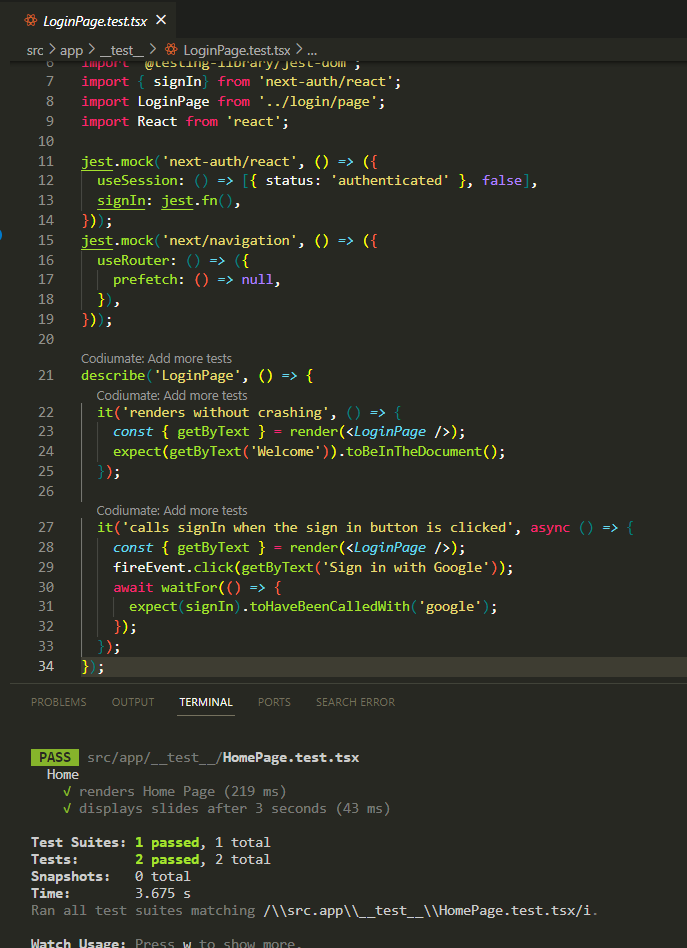
**HOME PAGE:**

****

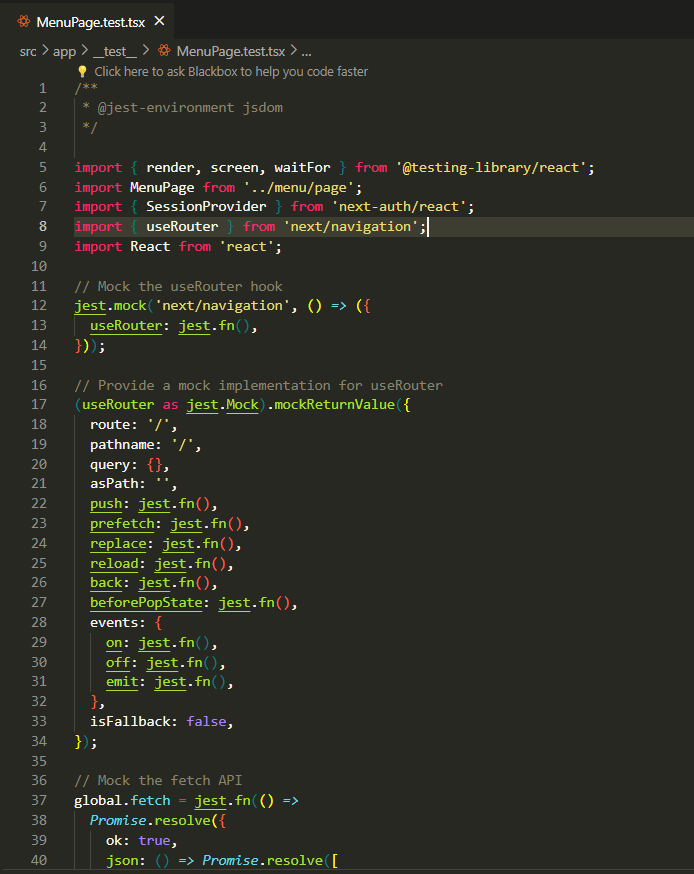
****

**LOGIN PAGE:**

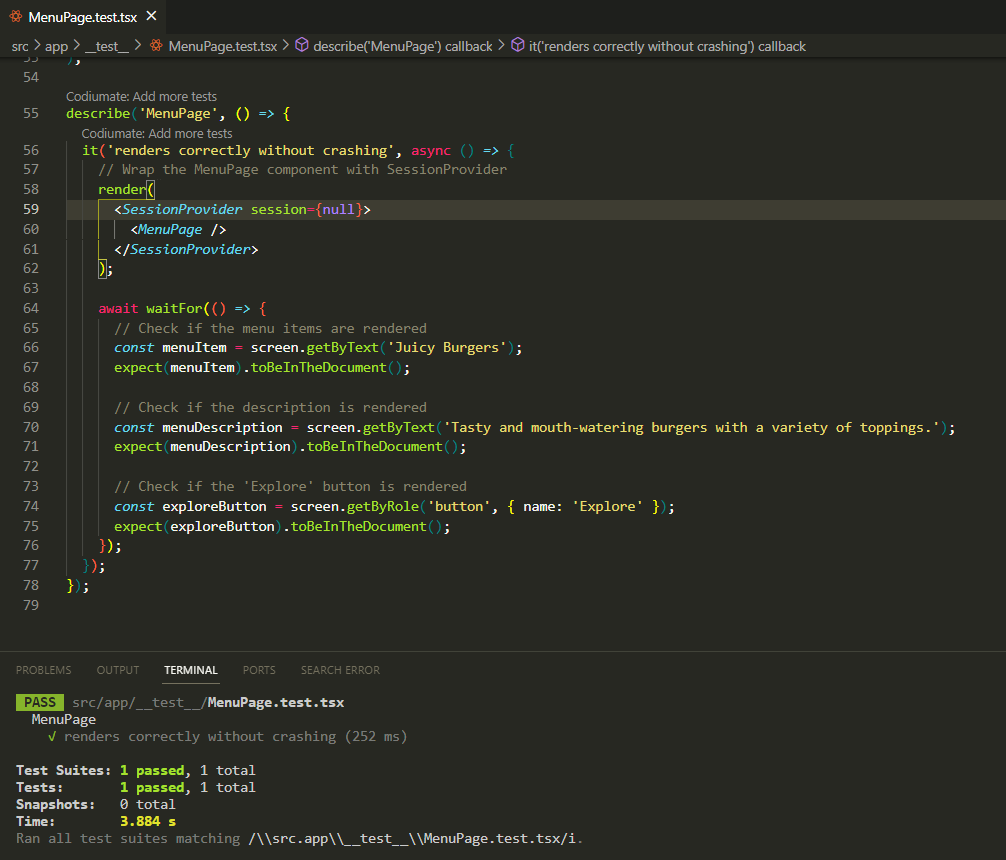
****

****

**MENU PAGE:**

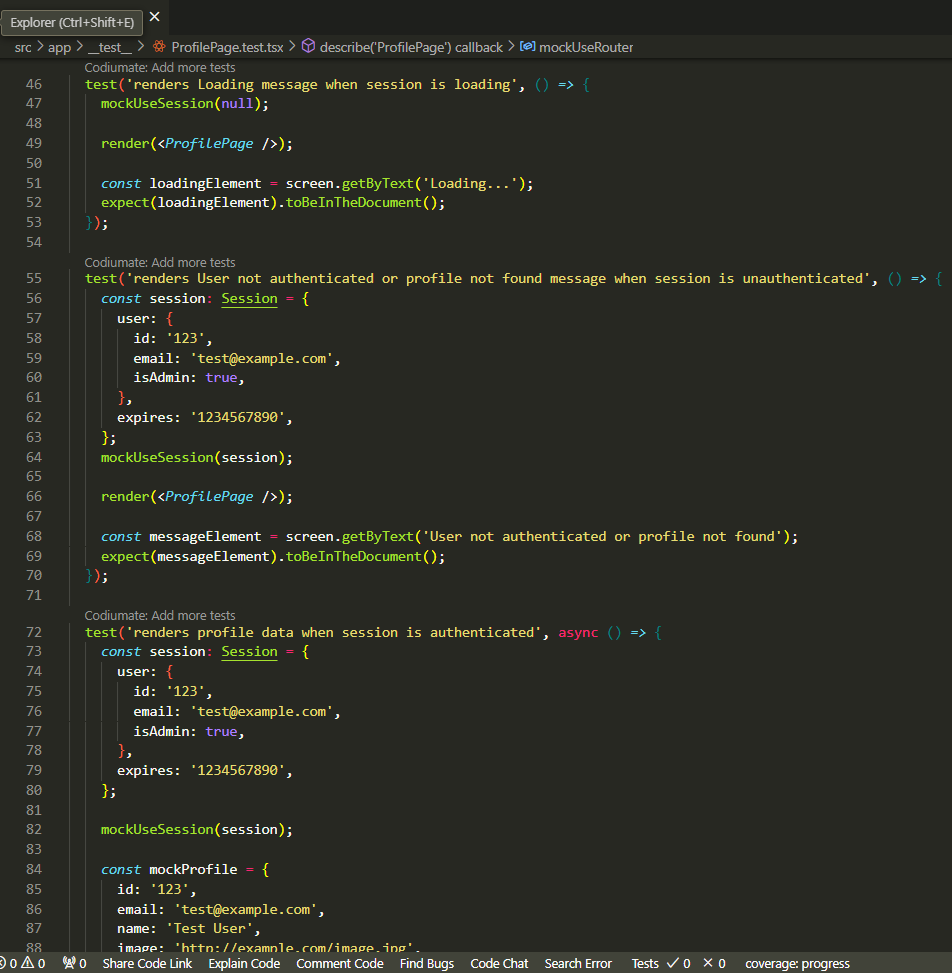
****

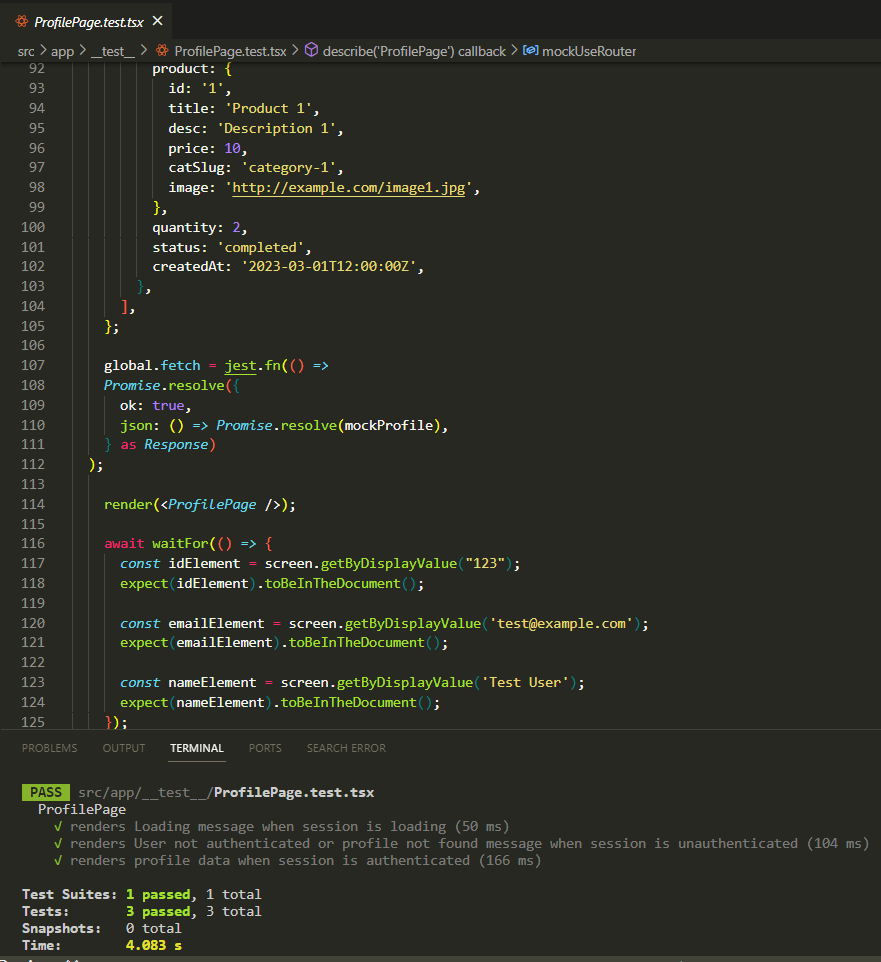
****

****

**PROFILE PAGE:**

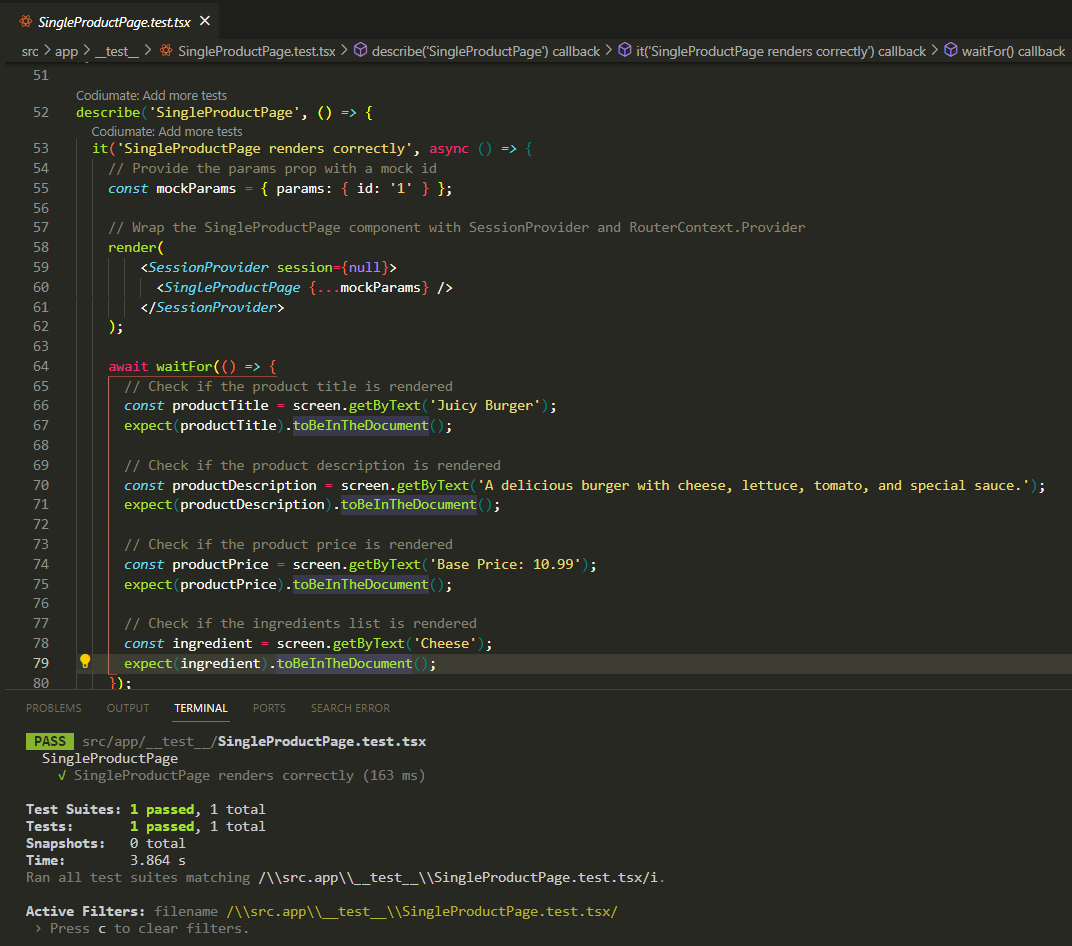
****

****

****

**SINGLE PRODUCT PAGE:**

****

****

**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.

**Add To Cart Page:**

**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:**

**Test Case: Verify Checkout Process**

* **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.