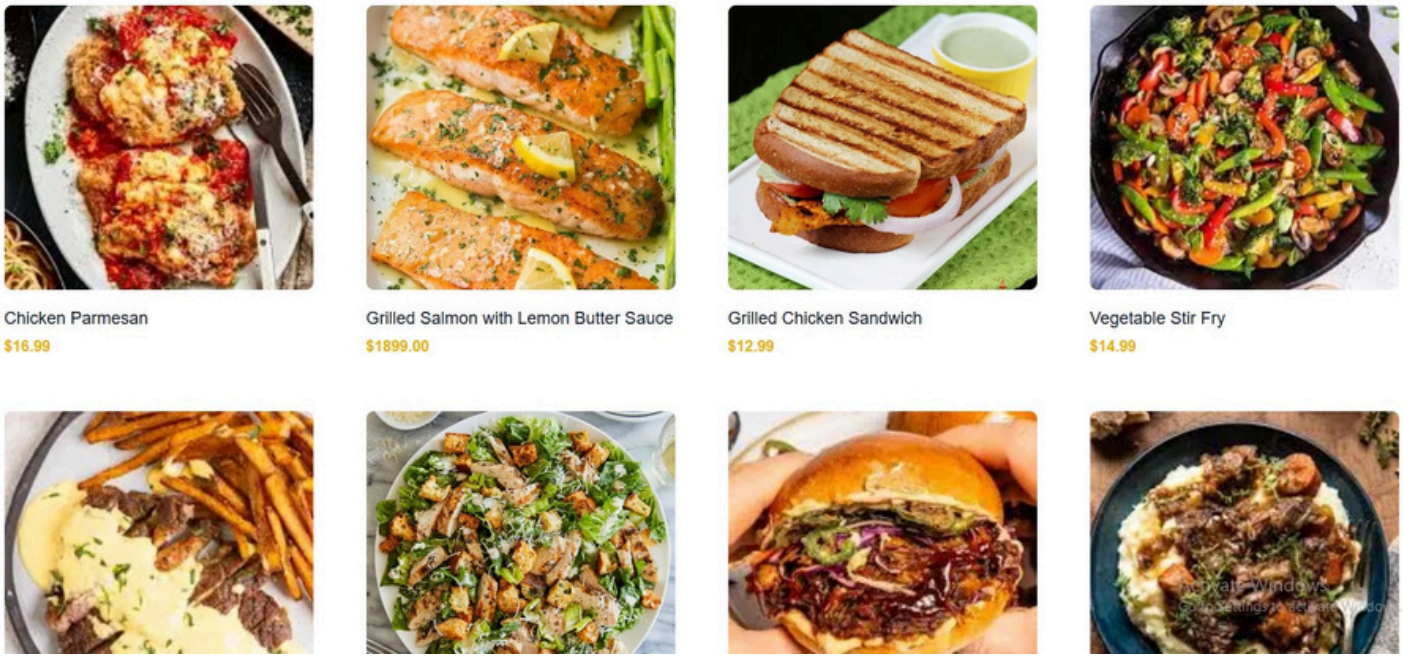
"Day 5: Testing Error Handling & Backend Integration Refinement (Q-Commerce)."

Test Case ID: TC001 - A unique identifier for the test. **Test Case Description: Validates the functionality of the Product Listing Page** - Explains the purpose of the test.
Test Steps: Lists the actions taken during the test, in this case, navigating to the Products Page and verifying the correct display of products.
Expected Result: Describes the anticipated outcome if the functionality is working correctly.

Actual Result: States what actually happened during the test. In this case, all products were displayed as expected.
Severity Level: Low - Indicates the impact of a potential failure, if one occurred. **Assigned To: (Not Assigned)** - Shows that no one is currently assigned responsibility for this test .
Remarks: Provides additional notes about the test; here, it confirms the successful completion and lack of issues.



Test Case ID: TC002 - The second test case in the suite.

Test Case Description: Validates API Error Handling for Data Fetching of Products - This test focuses on how the system handles errors when retrieving product data.

Test Steps:

1. **Simulate a disconnection in the GROQ query.** - This likely means intentionally disrupting the database query used to fetch products. GROQ is a query language used with Sanity.io, a content platform.
2. **Refresh the page to trigger the error handling mechanism.** - Refreshing the page after simulating the disconnection should activate the system's error handling.

Expected Result:

- **An appropriate error message should be displayed on the screen, clearly informing the user of the issue.** - The test specifies that a user-friendly error message is expected.

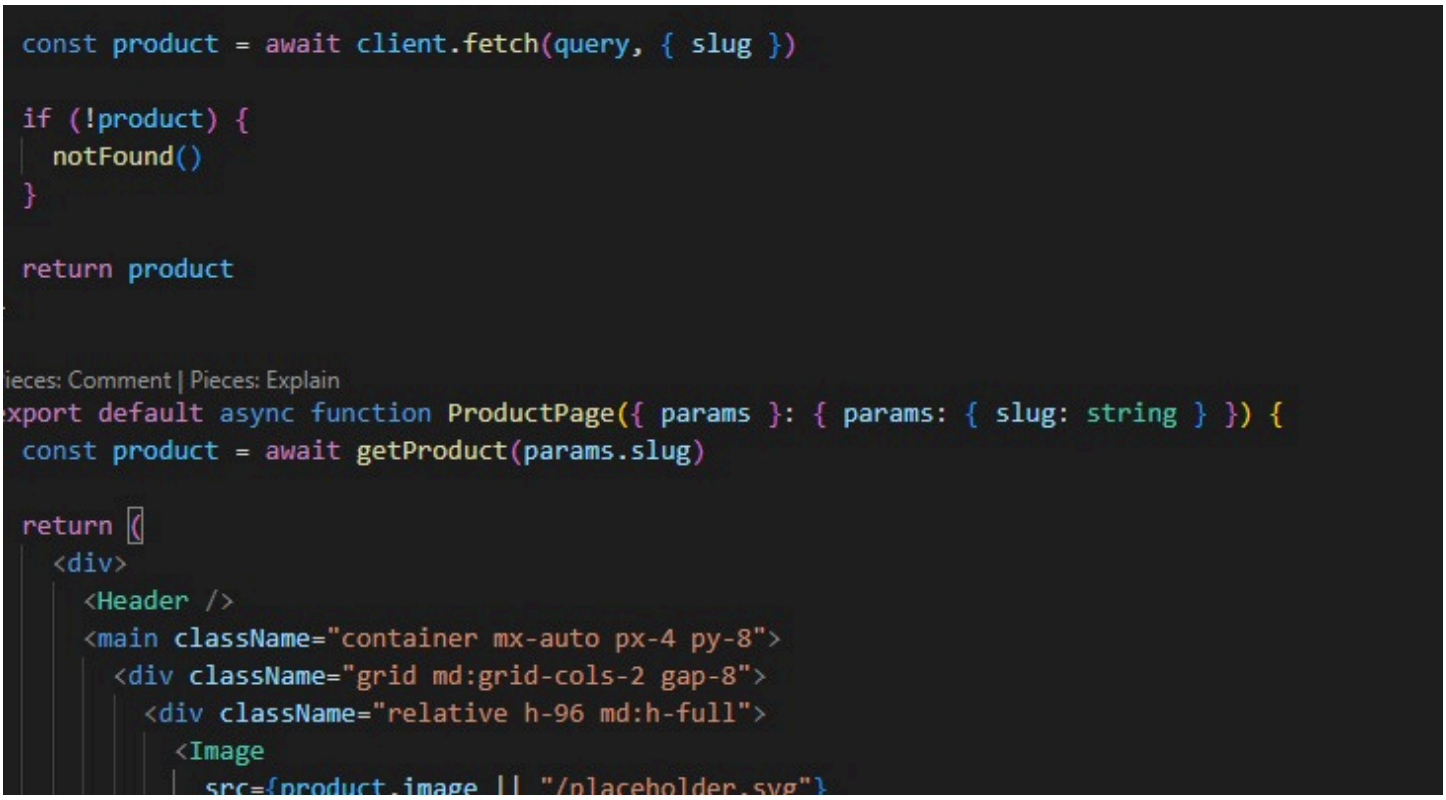
Actual Result:

- **The error message was displayed on the screen as expected, providing clear feedback to the user.** - The system behaved as expected.

Severity Level: Low - Indicates that this error, while important to handle, is not critical in terms of system stability.
Assigned To: (Not Assigned) - No one is assigned to this test instance.

Remarks:

- **Error handling works as expected. Consider refining the user experience to ensure graceful handling of such scenarios.** - The error handling functions, but there's room for improvement in how the error is presented to the user.



```
const product = await client.fetch(query, { slug })

if (!product) {
  notFound()
}

return product


ieces: Comment | Pieces: Explain
xport default async function ProductPage({ params }: { params: { slug: string } }) {
 const product = await getProduct(params.slug)

 return (
   <div>
     <Header />
     <main className="container mx-auto px-4 py-8">
       <div className="grid md:grid-cols-2 gap-8">
         <div className="relative h-96 md:h-full">
           <Image
             src={product.image || "/placeholder.svg"}
```

3. Test Case ID: TC003 - The third test case.

Test Case Description: Validate Cart Functionality - Focuses on verifying the core functions of the shopping cart.
Test Steps:

1. **Add a product to the cart.** - A product is added to the shopping cart.
2. **Navigate to the cart page.** - The tester then navigates to the cart page to view the added items.

Expected Result:

- **The selected product(s) should be displayed in the cart with accurate details such as name, quantity, price, and any applicable discounts.** - This outlines the expected information to be displayed for the added product in the cart.

Actual Result:

- **The selected product(s) are displayed in the cart as expected, with all details accurately shown.** - The system performed as expected, displaying all the necessary product information.

Severity Level: Medium - A medium severity suggests that while not critical, issues with the cart functionality could significantly impact the user experience and sales.
Assigned To: (Not Assigned)

Remarks:

- **Cart functionality is working as expected.** - A concise summary of the test result.

4. Test Case ID: TC004 - The fourth test case.

Test Case Description: Validate Mobile Responsiveness - This test checks if the website adapts correctly to di erent mobile screen sizes.
Test Steps:

1. **Open the website in a browser.** - Access the website using a web browser.
2. **Enable the device toolbar in the browser's developer tools.** - This step activates the browser's built-in tools to simulate di erent mobile devices.
3. **Verify the content display and layout.** - The tester checks if the website's content and layout are displayed correctly on the simulated mobile devices.

Expected Result:

- **The content should adjust and display properly according to the selected mobile device's screen size, ensuring a responsive layout and seamless user experience.** - This defines the desired outcome: the website should adapt seamlessly to various mobile screens.

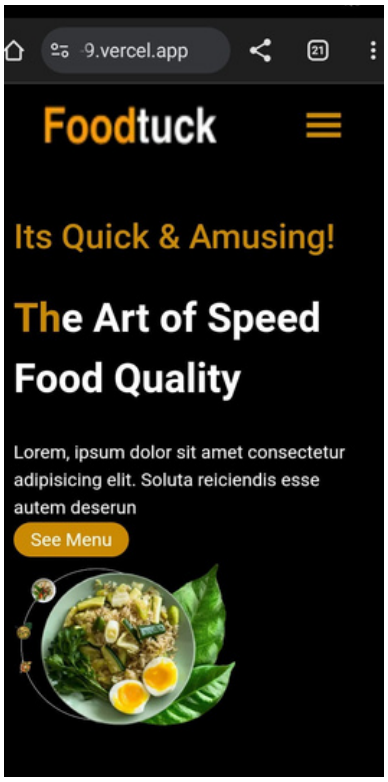
Actual Result:

- **The content adjusts and displays correctly based on the selected mobile device, maintaining responsiveness.** - The website performed as expected, adapting to the simulated mobile devices.

Severity Level: Medium - A medium severity highlights the importance of mobile responsiveness for a good user experience.
Assigned To: (Not Assigned) - No one is assigned to this test instance.

Remarks:

- **The application is responsive on mobile devices.**



5. Test Case ID: TC005 - The fifth test case.

Test Case Description: Validate Dynamic Product Detail Pages - This test verifies that clicking on a product card opens the correct product detail page.
Test Steps:

1. **Navigate to the product page.** - Go to the main listing of products.
2. **Click on any product card to view its details.** - Select a specific product to see its detailed information.

Expected Result:

- **The corresponding dynamic product detail page should open.** - The correct product details should be displayed upon clicking a product card.

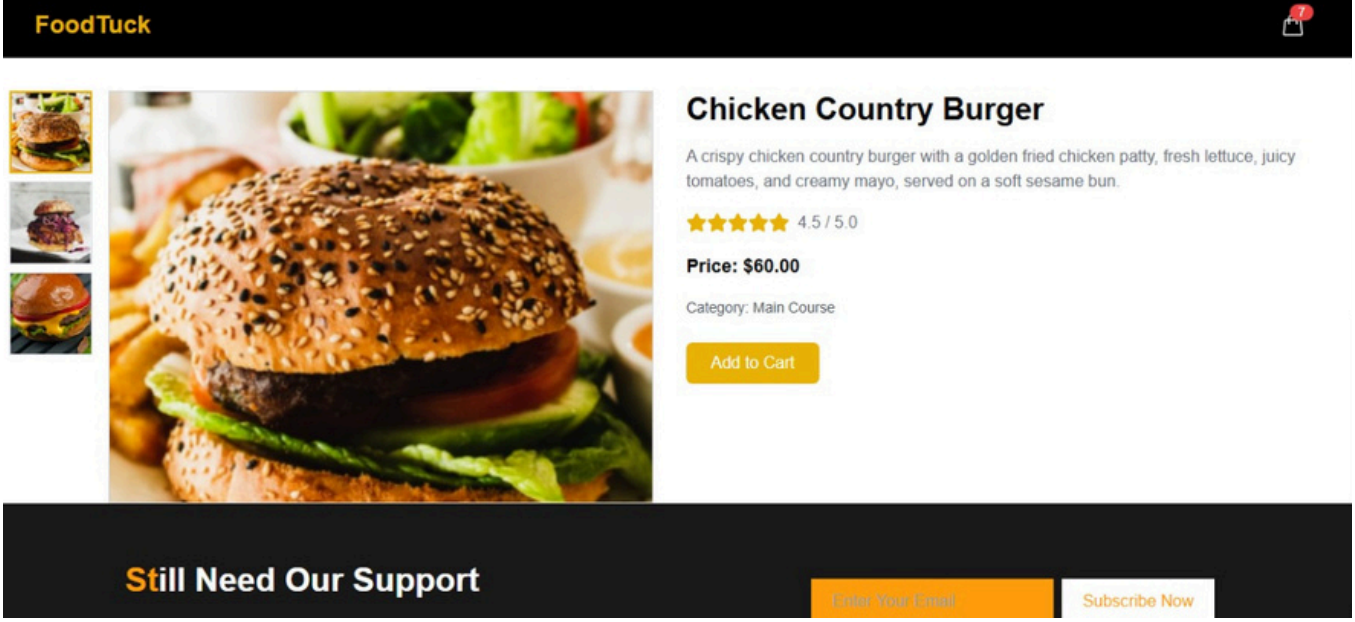
Actual Result:

- **The dynamic product detail page successfully opened.** - The system behaved as expected.

Severity Level: Low - A low severity suggests that while important, this functionality isn't critical to the core user flow.
Assigned To: - - No one is assigned to this test instance.

Remarks:

- **Test was successful, no issues were encountered.** - A concise summary of the test result.



6. Test Case ID: TC006 - The sixth test case.

Test Case Description: Cross-Browser Compatibility Testing on Opera - This test verifies that the website functions correctly in the Opera browser.
Test Steps:

1. **Open our website using the Opera browser.** - Access the website using the Opera web browser.
2. **Verify the website functions as expected.** - The tester checks all functionalities of the website to ensure they work correctly in Opera.

Expected Result:

- **The website should render and function correctly on the Opera browser.** - The website should display properly and all features should work as intended in Opera.

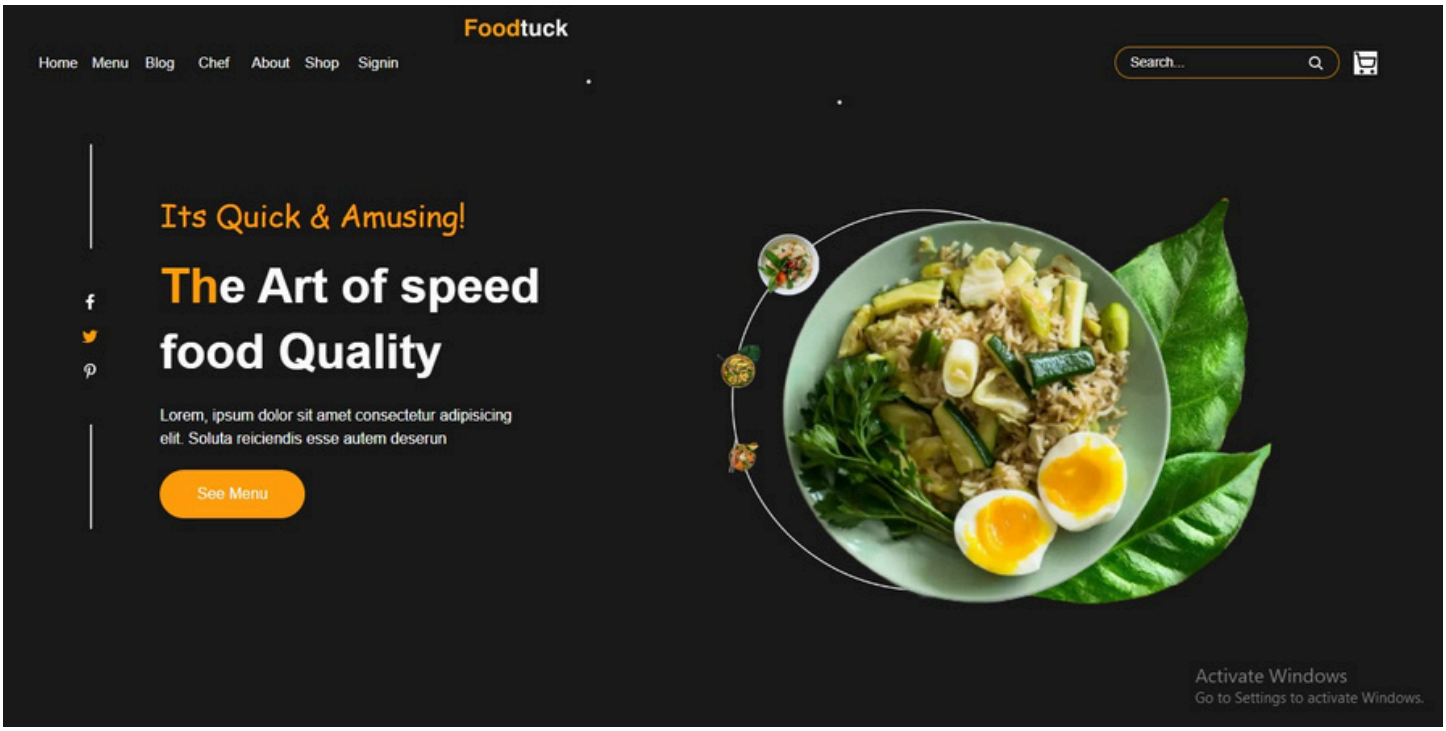
Actual Result:

- **The website is working correctly on the Opera browser.** - The system performed as expected in the Opera browser.

Severity Level: Low - A low severity suggests that while important, browser compatibility issues are not critical to the core user flow.
Assigned To: - - No one is assigned to this test instance.

Remarks:

- **Test completed successfully on Opera without any issues.**


Foodtuck
Home Menu Blog Chef About Shop Signin
Search...
Its Quick & Amusing!
The Art of speed food Quality
Lorem, ipsum dolor sit amet consectetur adipisicing elit. Soluta reiciendis esse autem deserun
See Menu
Activate Windows
Go to Settings to activate Windows.