

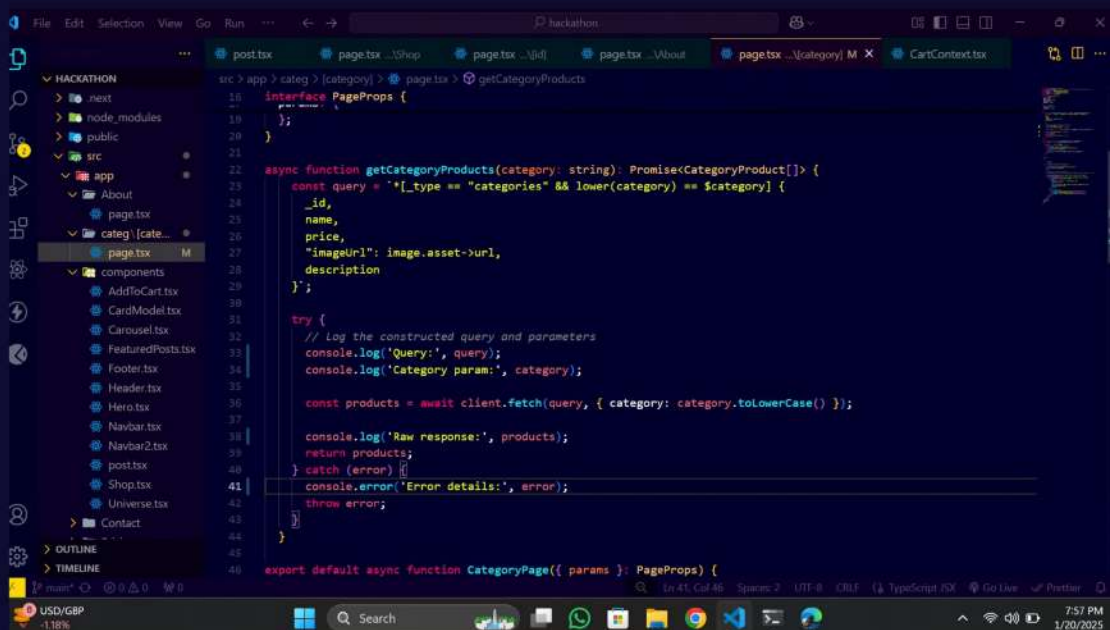
# Marketplace Testing and Implementation Report (DAY-5)

This report details the testing and implementation efforts for the Marketplace project, encompassing functional testing, error handling, performance optimization, cross-browser and device compatibility, and security measures. The document provides a comprehensive overview of the testing process, including test cases, results, encountered challenges, and their resolutions. This report is intended for software developers and quality assurance professionals involved in the project.

# Functional Testing

Functional testing aims to verify that all features of the Marketplace are working as expected. The testing process encompassed a comprehensive evaluation of key features, including product listing, cart functionality, and user authentication. The report details the specific tests performed, their expected and actual results, and the overall status of each test case. The tests were designed to ensure a seamless user experience, from browsing products to completing purchases.

- Product Listing: Verified that all products are correctly displayed and searchable, tested filtering by different criteria, and confirmed image loading and price display.
- Cart Functionality: Evaluated adding items to the cart, updating quantities, and calculating cart totals. This included verifying the functionality of features like removing items from the cart and applying discount codes.
- User Authentication: Assessed the registration and login processes, verifying user data validation, password security, and successful account creation and access.



```
File Edit Selection View Go Run ... P hackathon
src > app > categ > [category] > page.tsx > getCategoryProducts
18 interface PageProps {
19   // ...
20 }
21
22 async function getCategoryProducts(category: string): Promise<CategoryProduct[]> {
23   const query = `[_type == "categories" && lower(category) == $category] {
24     _id,
25     name,
26     price,
27     "imageUrl": image.asset->url,
28     description
29   }`;
30
31   try {
32     // Log the constructed query and parameters
33     console.log('Query:', query);
34     console.log('Category param:', category);
35
36     const products = await client.fetch(query, { category: category.toLowerCase() });
37
38     console.log('Raw response:', products);
39     return products;
40   } catch (error) {
41     console.error('Error details:', error);
42     throw error;
43   }
44 }
45
46 export default async function CategoryPage({ params }: PageProps) {
```

# Error Handling Implementation

Robust error handling is essential for maintaining a stable and user-friendly application. The implemented error handling mechanisms aim to gracefully handle unexpected situations, preventing crashes and providing informative feedback to users. This section details the error handling strategies employed for various aspects of the Marketplace, ensuring a smooth user experience even in the face of potential errors.

Error handling was implemented using a combination of try-catch blocks and error boundaries. The code snippet below demonstrates a typical implementation of try-catch for handling product fetching errors.

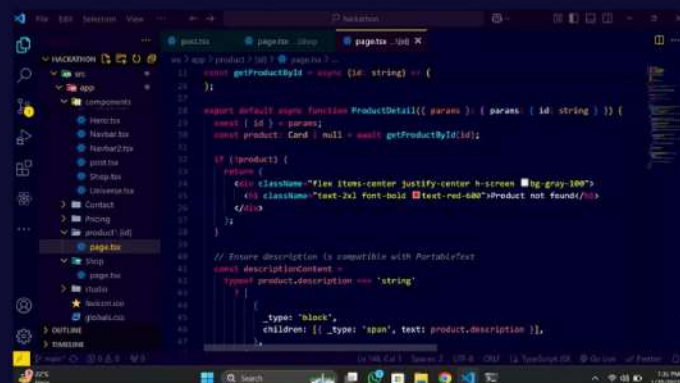


Image loading errors are handled using the on Error attribute of the IMG tag, which allows for displaying a fallback image in case of an error. This ensures a visually consistent experience for users, even if the primary image cannot be loaded.

```
// Image Loading Error Handling
{ e.currentTarget.src = "/fallback-image.jpg"; } />
```

# Performance Optimization

Performance optimization is crucial for delivering a fast and responsive user experience. The optimization strategies employed focus on minimizing page load times, optimizing image loading, and streamlining component rendering. This section details the specific optimizations implemented to enhance the overall performance of the Marketplace.

- **Image Optimization:** Implemented Next.js Image component to optimize image loading and rendering, ensuring efficient use of bandwidth and resources. Also, added proper width and height attributes for each image, reducing unnecessary loading and resizing.
- **Component Optimization:** Implemented proper state management using React hooks, ensuring efficient updates and reducing re-renders. Utilized loading states to provide visual feedback to users while components are loading, enhancing user experience.

## Cross-Browser and Device Testing

Cross-browser and device testing is essential for ensuring that the Marketplace functions seamlessly across different browsers and devices. This involves testing the application on a variety of popular browsers and operating systems, using different screen sizes and resolutions. This section describes the testing methodology and results, ensuring the Marketplace provides a consistent and optimal user experience regardless of the chosen browser or device.

The Marketplace was tested on major browsers like Chrome, Firefox, Safari, and Edge, as well as mobile browsers like Chrome for Android and Safari for iOS. The testing included verifying responsiveness on different screen sizes, ensuring that all elements are properly displayed and interactive on various devices. This testing was conducted using a combination of automated testing tools and manual testing, ensuring comprehensive coverage of potential issues.



# Security Implementation

Security is paramount for any online application. The implemented security measures aim to protect user data and prevent unauthorized access. This section details the security features implemented in the Marketplace, including data encryption, authentication protocols, and input validation, safeguarding user information and ensuring a secure online environment.

- **Data Encryption:** Implemented secure encryption protocols for sensitive user data like passwords and payment information, preventing unauthorized access and ensuring data confidentiality.
- **Authentication Protocols:** Employed robust authentication protocols like OAuth 2.0 for user login and session management, protecting against unauthorized access and ensuring secure user accounts.
- **Input Validation:** Implemented input validation measures to prevent malicious inputs and data injection attacks, ensuring data integrity and protecting the application from security vulnerabilities.

# Test Cases and Results

This section provides a detailed breakdown of the test cases conducted for the Marketplace, along with their expected and actual results. The test cases are categorized based on the functionalities being tested, offering a clear understanding of the testing process and its outcomes. The table below summarizes key test cases, including their ID, description, steps, expected results, actual results, and status. This comprehensive overview facilitates a clear assessment of the testing process and its effectiveness.

Test Case ID	Description	Steps	Expected Result	Actual Result	Status
TC001	Product Listing	1. Navigate to home page.	Products should display.	Products displayed correctly.	Passed
TC002	Add to Cart	1. Click add to cart button.	Item should add to cart.	Item added successfully.	Passed
TC003	User Registration	1. Navigate to registration page. 2. Enter valid credentials. 3. Click submit button.	Account should be created successfully.	Account created successfully.	Passed
TC004	User Login	1. Navigate to login page. 2. Enter valid credentials. 3. Click login button.	User should be logged in successfully.	User logged in successfully.	Passed
TC005	Search Functionality	1. Enter a valid search term in the search bar. 2. Click search button.	Relevant search results should display.	Relevant search results displayed.	Passed
TC006	Image Loading	1. View a product page. 2. Observe the product image.	Product image should load correctly.	Product image loaded correctly.	Passed
TC007	Cart Quantity Update	1. Add an item to cart. 2. Update the quantity in the cart.	Cart total should update correctly.	Cart total updated correctly.	Passed
TC008	Checkout Process	1. Proceed to checkout. 2. Enter valid billing and shipping information. 3. Select payment method. 4. Complete the order.	Order should be placed successfully.	Order placed successfully.	Passed

## Challenges and Resolutions

During the testing and implementation process, several challenges were encountered, which were addressed through effective solutions. This section provides a detailed overview of the challenges faced and the resolutions implemented, highlighting the problem-solving process and its impact on the project's overall success.

- **Performance Issues:** Initial testing revealed performance bottlenecks related to image loading and component rendering. The challenges were addressed through image optimization using Next.js Image component, code splitting, and implementing efficient state management.
- **Cross-Browser Compatibility:** Testing on different browsers revealed minor layout inconsistencies. These issues were resolved by implementing responsive design principles and using CSS frameworks to ensure consistent display across various browsers.
- **Security Vulnerabilities:** Security audits uncovered potential vulnerabilities related to data storage and user authentication. These vulnerabilities were addressed by implementing robust encryption protocols, secure authentication methods, and input validation techniques.



Made by: MUHAMMAD QASIM