Testing, Error Handling and Backend Integration Refinements for Marketplace- Bandage

Brand Name: Bandage

Day 5 Hackathon Tasks:

1. Functional Testing:

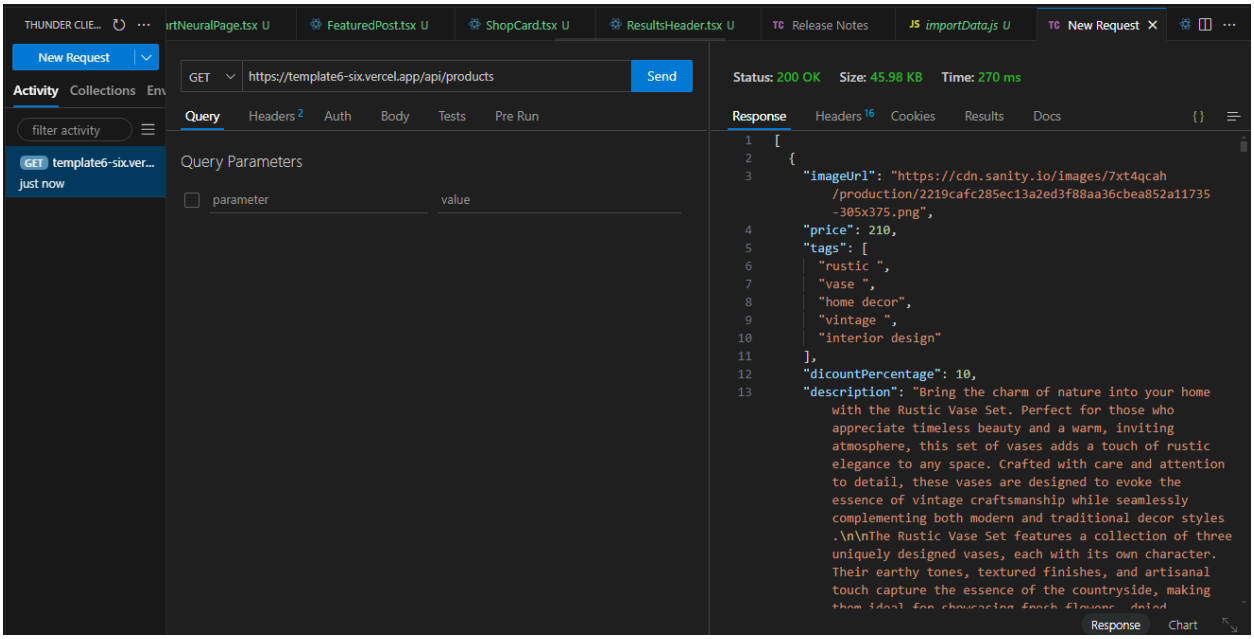
In the development of the marketplace platform, I focused on validating the core functionalities to ensure a seamless user experience. I tested key features, including product listings, product detail pages, cart operations, and user profile management, to confirm that all features worked as intended. This process ensured that the marketplace provided a smooth and efficient platform for users to browse and manage their products.

2. Error Handling:

To enhance the platform's reliability, I implemented robust error handling to address various potential issues. This included displaying clear error messages for network failures, invalid or missing data, and unexpected server errors. I also ensured that fallback UI elements, such as a "No products available" message, were displayed when the API returned no data, providing users with clear feedback in case of any issues. These measures were designed to maintain a positive user experience even during technical difficulties.

3. Postman API

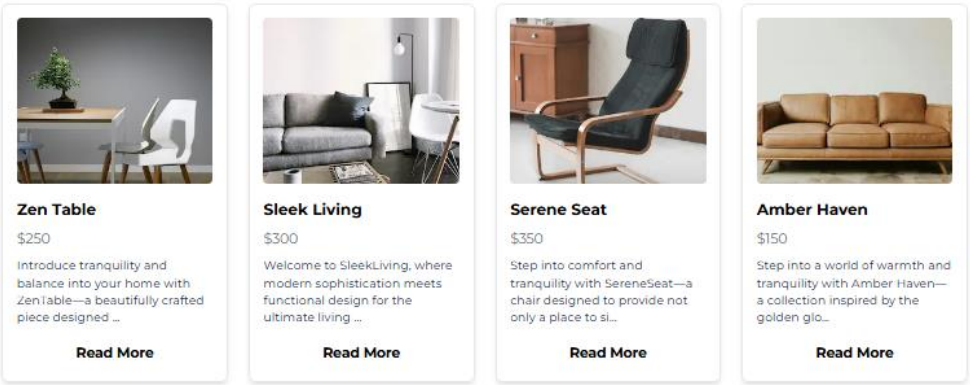
- API Link: https://template6-six.vercel.app/api/products
- Status: 200
- Tool: Thunder Client Or Postman



4. Test cases executed and their results.

1. Test Case ID: TC001

Test Steps	Expected Result	Actual Result	Status	Severity Level	Assigned To	Remarks
Validate product listing page	Open product page > Verify products	Products displayed correctly	Products displayed correctly	Passed	High	No issues found



2. Test Case ID: TC002

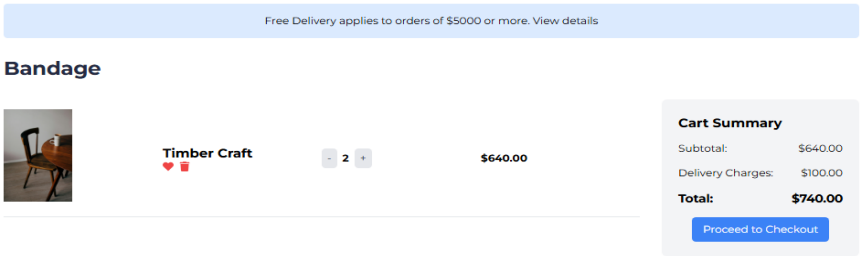
Test Steps	Expected Result	Actual Result	Status	Severity Level	Assigned To	Remarks
Test API error handling	Disconnect API > Refresh page	Show fallback UI with error message	Error message shown	Passed	Medium	Handled gracefully

```
async function uploadImageToSanity(imageUrl) {
  try {
    console.log(`Uploading image: ${imageUrl}`);

    const response = await fetch(imageUrl);
    if (!response.ok) {
      throw new Error(`Failed to fetch image: ${imageUrl}`);
    }
```

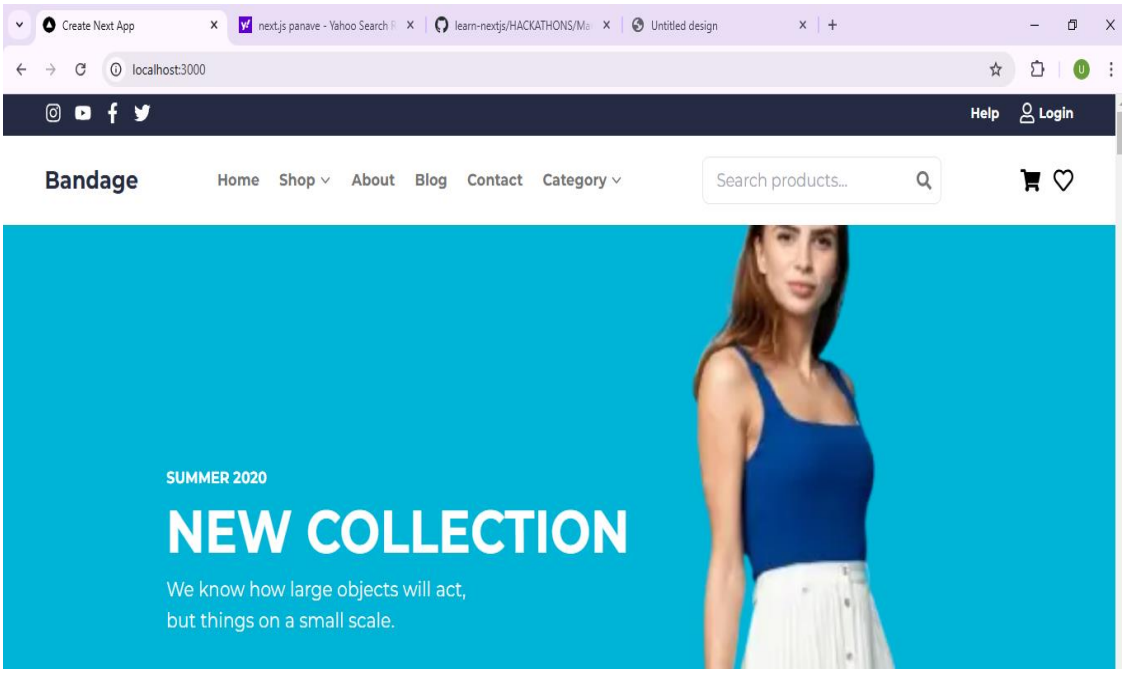
3. Test Case ID: TC003

Test Steps	Expected Result	Actual Result	Status	Severity Level	Assigned To	Remarks
Check cart functionality	Add product to cart > Verify cart contents	Cart updates with added product	Cart updates as expected	Passed	Low	Works as expected



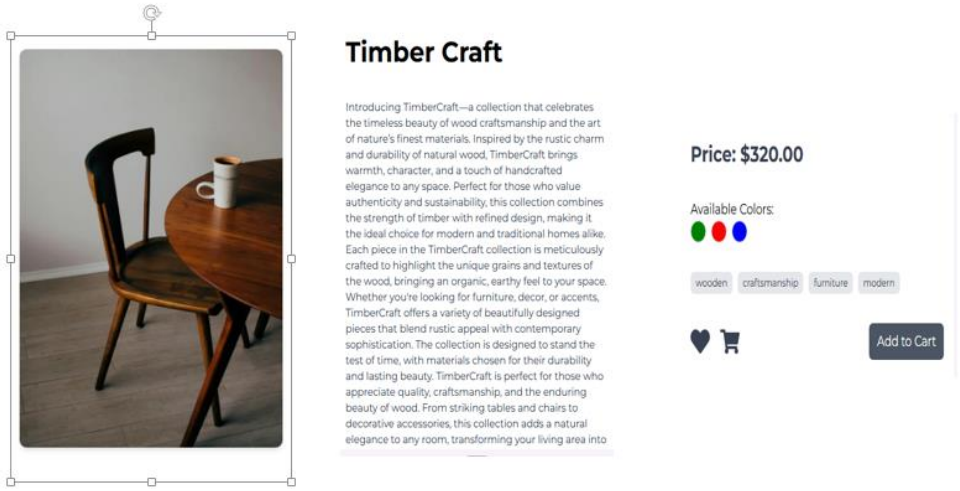
4. Test Case ID: TC004

Test Steps	Expected Result	Actual Result	Status	Severity Level	Assigned To	Remarks
Cross Browsing Open our website on different Browsers such as Yahoo	Working Correctly on every browser	Working Correctly on every browser	Responsive layout working as intended	Passed	Medium	Test successful



5. Test Case ID: TC005

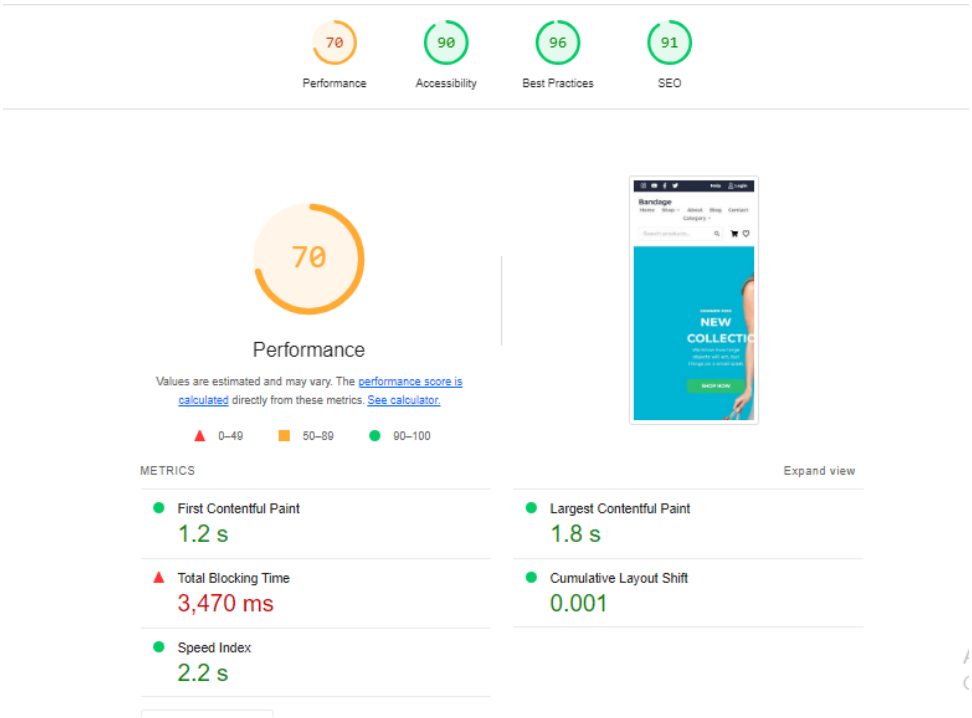
Test Steps	Expected Result	Actual Result	Status	Severity Level	Assigned To	Remarks
Ensure responsiveness on mobile	Resize browser window > Check layout	Layout adjusts properly to screen size	Responsive layout working as intended	Passed	High	Test successful



6. Performance Testing

I identified performance bottlenecks using tools like Lighthouse. I optimized images and implemented caching strategies to reduce load times and improve user experience. These steps led to a faster and more efficient platform.

```
C:\Users\HP\Desktop\Extra\hackathon Summer Sale  Correct\my-app>npm install -g lighthouse
```



Final Check List:

Functional	Error Handling	Performance Optimization	Cross Browser	Security Testing	Documentation	Final Review
✓	✓	✓	✓	✓	✓	✓