Day 5 - Testing, Error Handling, and Backend Integration Refinement

Prepared By Aamna Ansari

Hackathon Day – 5

Testing, Error Handling, and Backend Integration Refinement

Day 5 is dedicated to ensuring your marketplace is fully optimized, tested, and ready for real-world deployment. This phase focuses on:

Comprehensive Testing: Conduct rigorous testing of all components, including backend integrations, to identify and resolve any issues.

Performance Optimization: Fine-tune your application to ensure fast load times, seamless functionality, and scalability for handling customer traffic.

Error Handling: Implement robust error-handling mechanisms to ensure smooth recovery from potential issues and enhance system reliability.

User Experience Refinement: Polish the user interface and workflows to deliver a seamless and intuitive experience for end-users.

Key Objective:

1. Comprehensive Testing

Feature Validation: Thoroughly validate all features and backend integrations to ensure they function as intended.

Testing Scope: Conduct functional, non-functional, and user acceptance testing (UAT) to cover all use cases and meet user expectations.

2**. Error Handling**

User-Friendly Feedback: Implement clear and helpful error messages with fallback mechanisms to maintain a seamless user experience.

API Error Management: Gracefully handle API errors by incorporating fallback UI elements to ensure system resilience.

3. **Performance Optimization**

System Efficiency: Optimize the application for speed, responsiveness, and reliability to handle real-world traffic.

4. Cross-Browser and Device Compatibility

Seamless Experience: Test the application across multiple browsers and devices to ensure consistent functionality and user experience.

5. Security Testing

Vulnerability Assessment: Identify and mitigate vulnerabilities to protect sensitive data and maintain system integrity.

6. Documentation

Professional Test Reports: Create comprehensive, CSV-based testing reports summarizing results, identified issues, and resolutions.

Deployment Guide: Prepare detailed deployment documentation to ensure a smooth launch process.

Functional Testing

Ensure that all core features of the marketplace work seamlessly to deliver an error-free and intuitive user experience. Below are the key features to test:

Key Features to Test:

Product Listing:

Verify that all products are displayed accurately with correct details such as name, price, and images.

Ensure pagination or infinite scrolling works correctly if implemented.

Product Details:

Confirm that product detail pages display accurate information, including price, description, images, and availability.

Dynamic Routing:

Ensure seamless navigation to individual product detail pages from product listings.

Add to Cart:

Verify that items can be added to the cart without errors.

Confirm that users can update quantities, remove items, and view the cart summary correctly.

Add to Wishlist:

Test that products can be added to and removed from the wishlist.

Responsive Design:

Validate that all features and pages are fully responsive, adapting seamlessly to different screen sizes (mobile, tablet, desktop).

Check for usability and consistency across devices.

Key Areas to Address in Error Handling

Effective error handling ensures a smooth user experience even when issues occur. Below are key areas to address, along with actionable examples:

1. Network Failures

Objective: Display meaningful and user-friendly messages when connectivity issues arise.

Action:

Detect network issues and provide clear feedback to users.

Suggest actionable steps, such as checking the internet connection.

Example:

"Unable to connect to the server. Please check your internet connection and try again."

2. Invalid or Missing Data:

Objective: Handle scenarios where API responses return incomplete or invalid data gracefully.

Action:

Validate all incoming data from APIs.

Show appropriate fallback messages if essential information is missing.

Example:

"Some information is missing. Please refresh the page or try again later."

3. Unexpected Server Errors

Objective: Provide a generic yet user-friendly fallback for unhandled server-side errors.

Action:

Use default error pages or modals to notify users without exposing technical details.

Log errors for developers to debug.

Example:

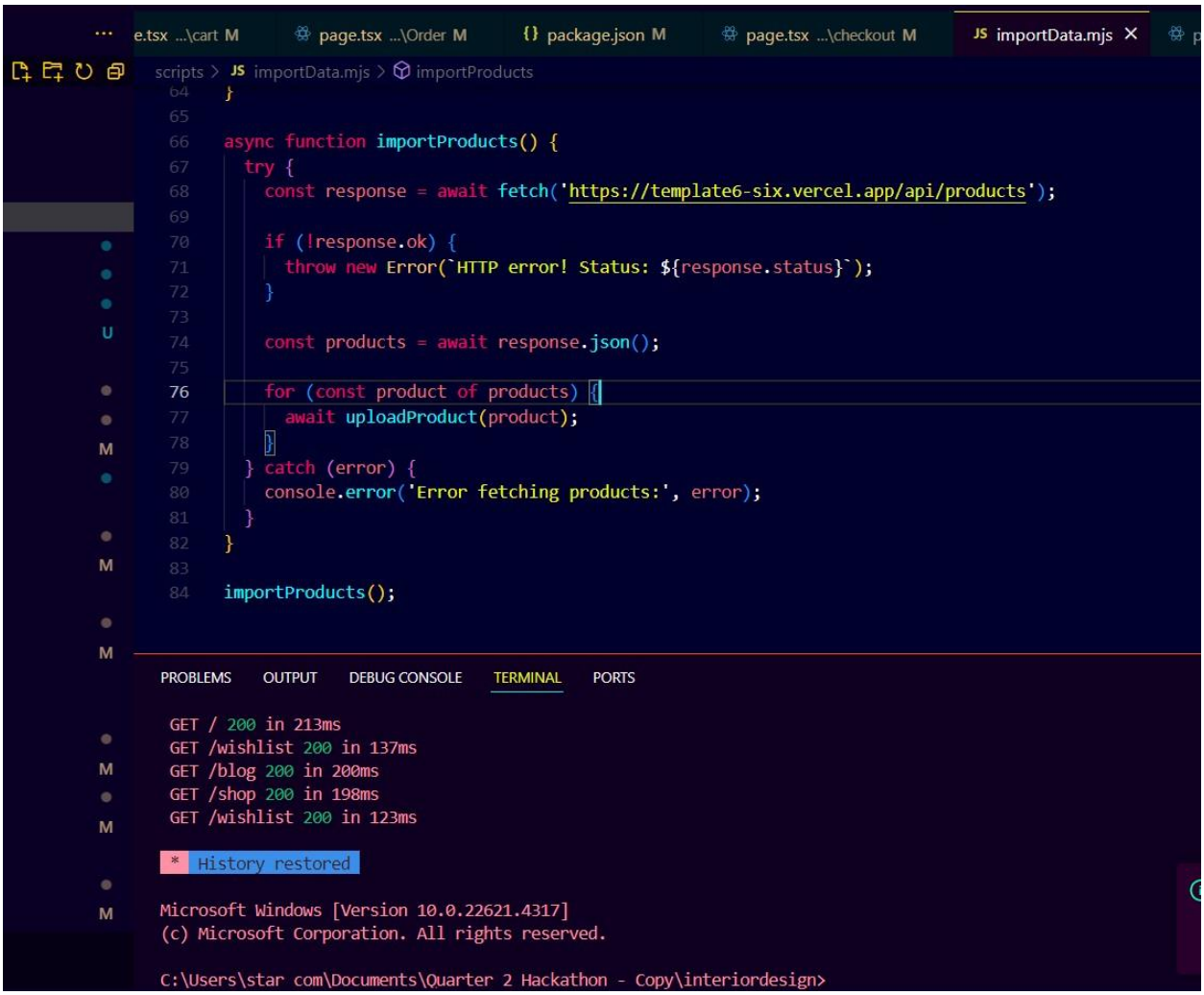
"Something went wrong on our end. We're working to resolve this. Please try again shortly."

4. API Error Handling

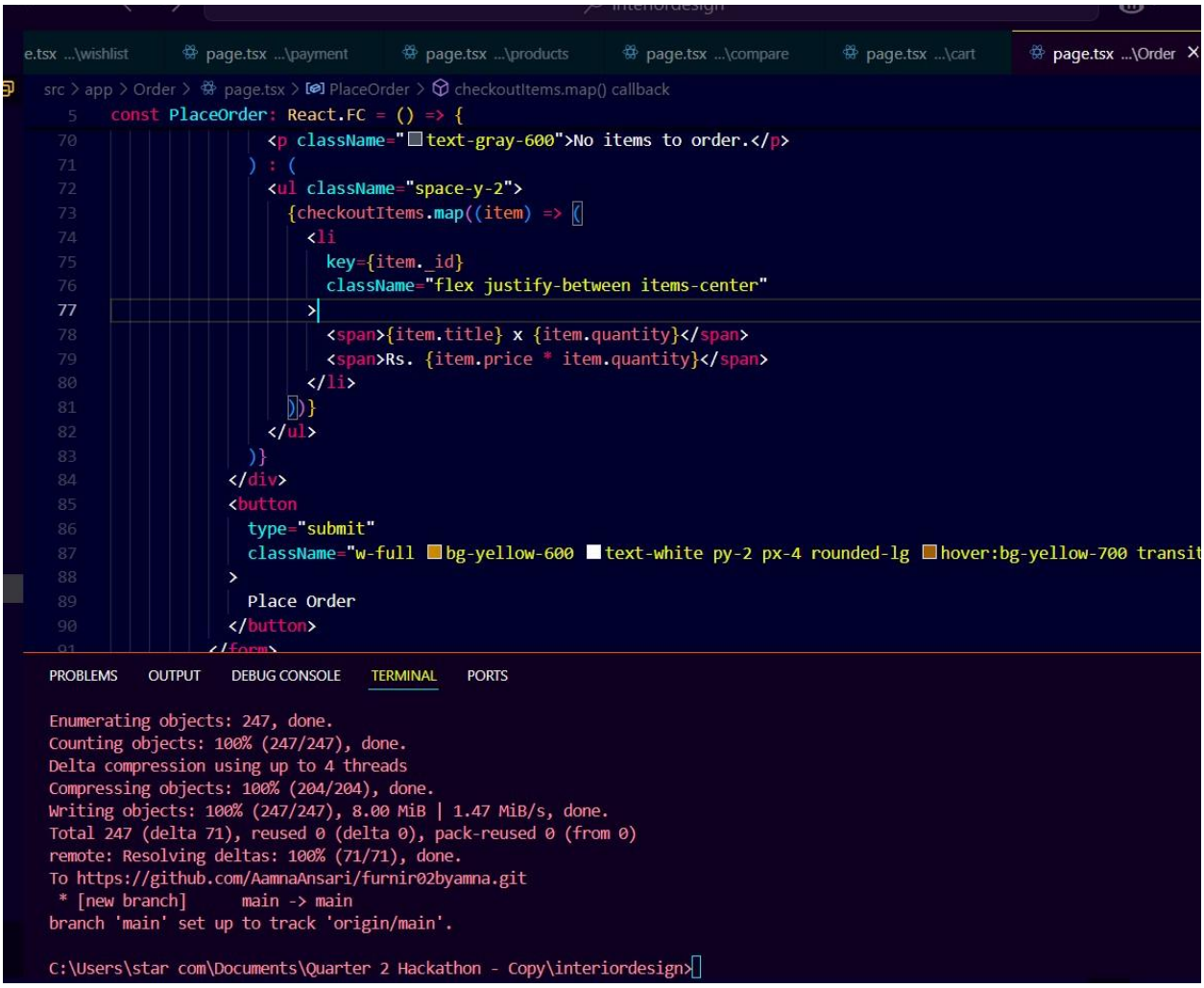
Objective: Ensure API errors are managed gracefully to maintain application stability.

Action:

Wrap API calls in try-catch blocks to handle exceptions effectively.


e.tsx ...\cart M
page.tsx ...\Order M
package.json M
page.tsx ...\checkout M
importData.mjs
scripts > importData.mjs > importProducts
}
async function importProducts() {
try {
const response = await fetch('https://template6-six.vercel.app/api/products');
if (!response.ok) {
throw new Error(`HTTP error! Status: ${response.status}`);
}
const products = await response.json();
for (const product of products) {
await uploadProduct(product);
}
} catch (error) {
console.error('Error fetching products:', error);
}
}
importProducts();
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
GET / 200 in 213ms
GET /wishlist 200 in 137ms
GET /blog 200 in 200ms
GET /shop 200 in 198ms
GET /wishlist 200 in 123ms
History restored
Microsoft Windows [Version 10.0.22621.4317]
(c) Microsoft Corporation. All rights reserved.
C:\Users\star com\Documents\Quarter 2 Hackathon - Copy\interiordesign>

UI for Order Placement:



Performance Optimization

Optimize the marketplace to improve load times, responsiveness, and overall performance.

Key Areas to Address

Optimize Assets:

Image Optimization: Compress images using tools like TinyPNG or ImageOptim to reduce size without losing quality.

Lazy Loading: Implement lazy loading for images and assets to defer loading until needed, improving initial load times.

Minimize JavaScript and CSS:

Remove Unused Code: Eliminate unused CSS and JavaScript to further reduce file sizes.

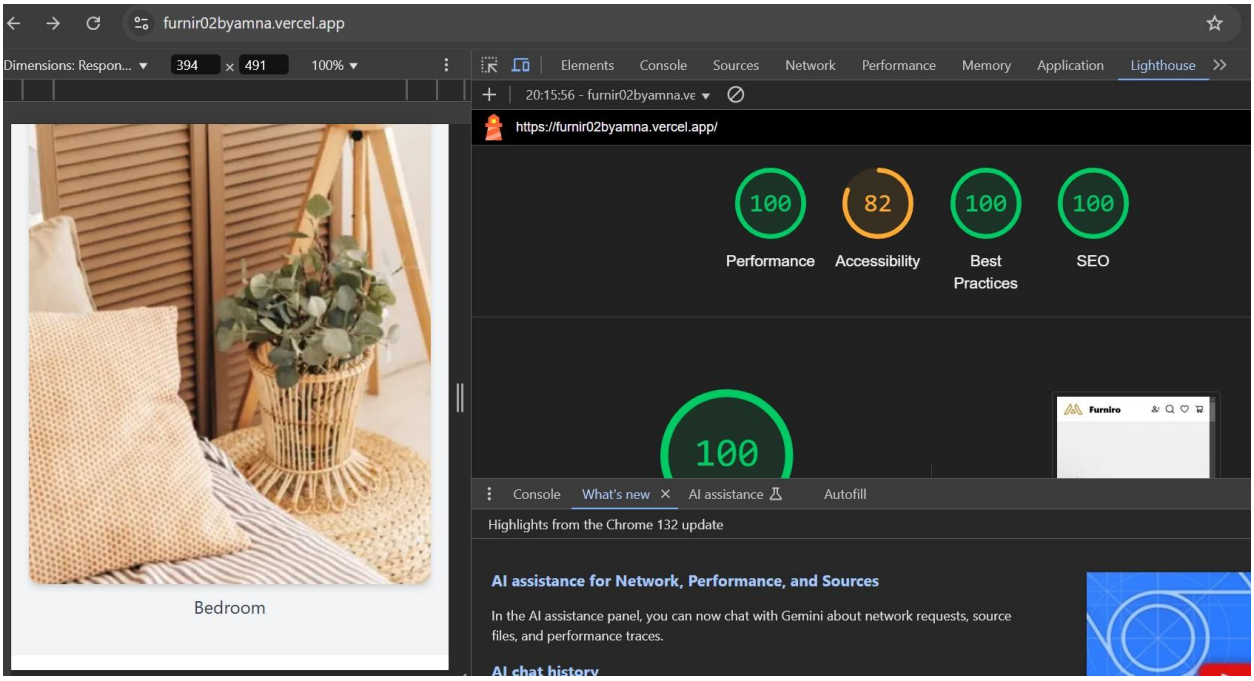
Implement Caching Strategies:

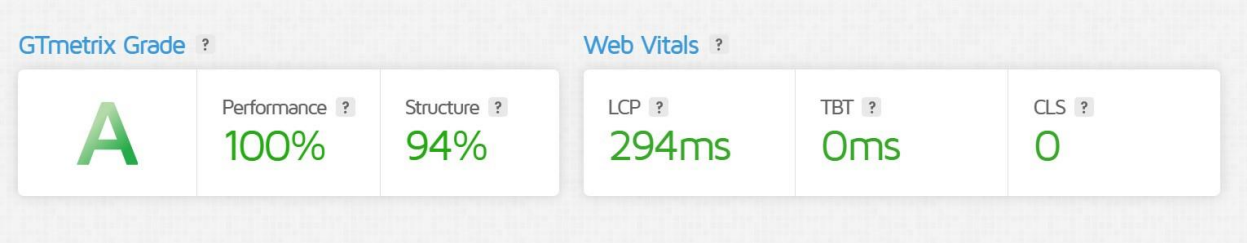
Browser Caching: Store static assets in the user's browser to avoid repeated network requests.

Service Workers: Cache resources and improve offline functionality using service workers.

Analyze Performance: Google Lighthouse: Use Lighthouse to audit the website's performance and identify areas to improve, such as image optimization and resource loading.

Webpage Test & GTMetrix: Use these tools to identify performance bottlenecks and improve page load speeds.





CSV Table:

Clipboard | Font | Alignment | Number | Styles | Cells | Editing

15

Test Case ID	Test Case Description	Test Steps	Expected Result	Actual Result	Status	Severity Level	Remarks
TC001	Products Listing	Open product page > Verify products	All Products should be displayed correctly	All Products displayed correctly as expected	Passed	High	No issues found
TC002	Add to Cart	Click on "add to cart" for any product	Products should be added to cart	Products shown in cart as expected	Passed	High	Handled gracefully
TC003	Cart Operations	Add and remove items from cart	Cart updates with added product or removing products	Cart updates as expected	Passed	High	Works as expected
TC004	Dynamic Routing	Click on Products to navigate to its detail page	The correct detail page should load	Correct page loads with right details	Passed	Medium	Test successful
TC005	Error Handling (Invalid data)	Show alerts if data is invalid	An appropriate error message should be displayed	Error message displayed for invalid input	Passed	Critical	Clear message displayed
TC006	Responsive Design	Testing website on (mobile and desktops)	The design should be adjustable in all devices	Website design adapts correctly to various screen sizes	Passed	Medium	Test successful
TC007	Product Details Page	Click on products to view its details	Product detail page should load correctly	Page loaded without any issues.	Passed	High	Test successful
TC008	Product Comparison Page	To compare products	Product comparison page should load correctly	Page loaded without any issues.	Passed	High	Handled gracefully

Conclusion

Day 5 focused on preparing the marketplace for deployment by ensuring all components were thoroughly tested, optimized, and refined. Comprehensive functional, error handling, and performance tests were conducted, validating key features like product listing, cart operations, and dynamic routing.

Robust error handling mechanisms and fallback UI elements were implemented to enhance reliability.

Performance optimization efforts resulted in faster load times and improved responsiveness across devices and browsers. The team also documented all findings and resolutions in a professional format, ensuring readiness for real-world deployment.

Thank You! Hope you like my work