Hackathon Day 5 Report

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

Objective

The goal of Day 5 was to refine the marketplace for real-world deployment by ensuring that all components were rigorously tested, optimized for performance, and equipped with robust error-handling mechanisms. The focus was on preparing the platform for a customer-facing environment, enhancing user experience, and finalizing testing documentation.

Key Activities and Achievements

1. Comprehensive Testing

• Conducted functional testing to verify core functionalities like search bar shopping cart operations, and checkout workflows.
• Performed non-functional testing to measure performance, load times, and stress-test the backend APIs under heavy traffic.
• Implemented user acceptance testing (UAT) by gathering feedback from test users to identify usability issues.

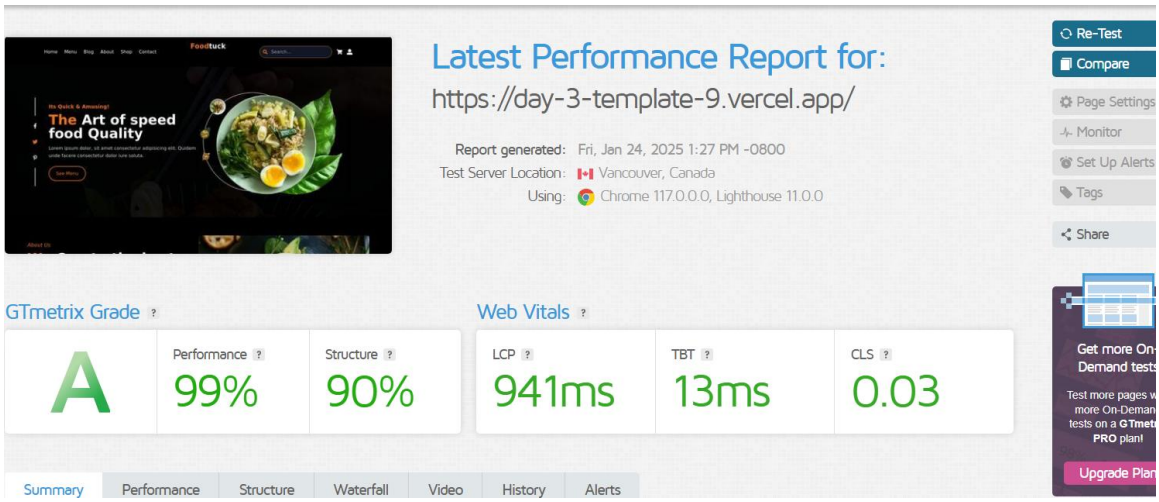
2. Error Handling Mechanisms

• Developed robust error-handling logic for frontend.
• Created fallback messages for common issues like failed API calls, slow network responses, or unavailable resources.
• Introduced logging mechanisms to track backend and API errors for future debugging.

```
useEffect(() => {
  const fetchProducts = async () => {
    setLoading(true); // Start loading
    setError(null); // Reset error state
    try {
      const productsData = await client.fetch(
        `*[_type == "food"]{
          name,
          price,
          description,
          category,
          originalPrice,
          "image": image.asset->url,
          "slug": slug.current,
        }`
      );
      setProducts(productsData);
      setFilteredProducts(productsData);
    } catch (err) {
      console.error("Error fetching products:", err);
      setError("Failed to fetch products. Please try again later.");
    } finally {
      setLoading(false); // Stop loading
    }
  };
  fetchProducts();
}, []);
```

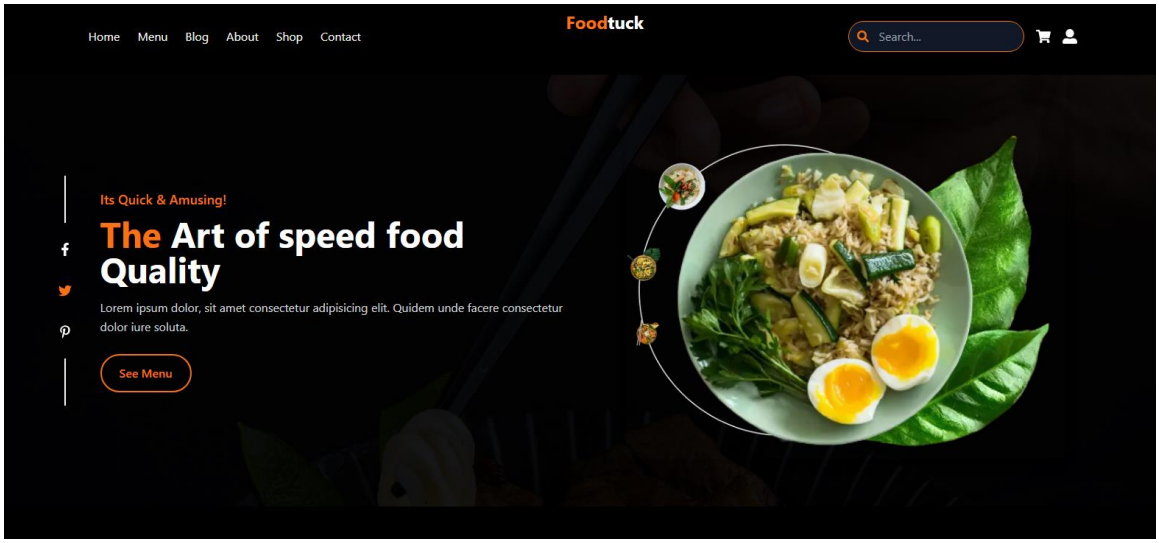
3. Performance Optimization

- Optimized frontend assets by minimizing JavaScript and CSS bundles.
- Implemented lazy loading for images and components to improve initial page load times.
- Utilized caching strategies for frequently accessed data.
- Enhanced database queries for faster response times on API endpoints.



4. Cross-Browser and Device Compatibility

• Ensured compatibility across all major browsers, including Chrome, Firefox, Edge, and Safari.
• Verified responsiveness on multiple device resolutions, including mobile, tablet, and desktop.
• Resolved inconsistencies in styling and layout for seamless user experiences across platforms.



5. Professional Testing Documentation

• Prepared detailed testing documentation that meets industry standards.
• Compiled a CSV-based test report outlining test cases, expected results, actual results, and resolutions for failed cases.
• Designed fallback UI components that displayed user-friendly messages when APIs

returned errors. Examples included retry buttons, placeholder content, and informative modals for unresolved issues.

Challenges and Resolutions

• Challenge: Handling API errors without disrupting user experience.
  Resolution: Implemented fallback UI and retry mechanisms to ensure a seamless experience even during backend failures.
• Challenge: Ensuring smooth performance under high traffic.
  Resolution: Optimized server-side rendering (SSR) processes and enabled caching for frequently accessed endpoints.
• Challenge: Documenting test results professionally.
  Resolution: Used spreadsheet tools to organize test cases into a CSV format, providing detailed insights into test execution and outcomes.

Future Recommendations

1. Conduct periodic testing post-deployment to ensure continued platform stability.
2. Monitor real-world performance metrics using tools like gtmetrix.com.

By the end of Day 5, the marketplace was refined, optimized, and ready for real-world deployment. The focus on robust testing, error handling, and documentation laid a strong foundation for success.