GIAIC Karachi, Pakistan

GIAIC
Karachi, Pakistan
+92-333-2236799
azmataliakbar@gmail.com

Day – 5 21-01-2025

Hackathon – 3

Testing, Error Handling, and Backend Integration Refinement

This document outlines essential steps and considerations for testing, error handling, and refining backend integration during the development of an e-commerce website using Next.js. The following sections provide detailed insights, accompanied by tables for clarity.

1. **Functional Testing**
2. **Error Handling**
3. **Security Improvements**
4. **Performance Optimization**
5. **Responsiveness**
6. **Cross-Browser Compatibility**
7. **Device Testing**

1. Functional Testing

Functional testing ensures that core features of the website work as intended.

Feature	Tested Functionality	Result
Product Display	Verify products load dynamically from the backend.	Pass / Fail
Add to Cart	Test adding single and multiple items to the cart.	Pass / Fail
Checkout Process	Validate order placement and payment flow.	Pass / Fail
Navigation Links	Check header and footer links for proper routing.	Pass / Fail
User Registration & Login	Test user sign-up, login, and logout functionality.	Pass / Fail

2. Error Handling

Error handling improves user experience by displaying meaningful messages during failures.

Steps to Implement Error Messages:

1. **Frontend Validation:**
    - Add checks for required fields in forms (e.g., email, password).
    - Provide real-time feedback for invalid input (e.g., "Email is required").
2. **Backend Error Handling:**
    - Catch errors during API requests (e.g., fetch failure).
    - Use try...catch blocks in asynchronous functions.
3. **Example Code for Error Handling:**

```
try {
  const response = await fetch('/api/checkout', {
    method: 'POST',
    body: JSON.stringify(orderData),
  });
  if (!response.ok) {
        throw new Error('Failed to process the order. Please try again later.');
      }
      const data = await response.json();
      console.log('Order Successful:', data);
    } catch (error) {
      console.error('Error:', error.message);
      setErrorMessage(error.message); // Display error to the user
    }
```

Error Scenario	Message Displayed
Form Submission Error	"Please fill in all required fields."
API Fetch Failure	"Unable to fetch data. Please try again later."
Invalid Login Credentials	"Incorrect email or password. Please try again."

3. Security Improvements

To secure the e-commerce application:

Aspect	Implementation
Secure Authentication	Use JWT tokens and encrypt sensitive data.
HTTPS Enforcement	Use SSL certificates for secure communication.
SQL Injection Prevention	Use parameterized queries in backend APIs.
Input Validation	Sanitize user inputs to avoid XSS and CSRF attacks.

4. Performance Optimization

Performance is critical for user retention and satisfaction.

Techniques:

Area	Optimization
Image Loading	Use Next.js Image Optimization for lazy loading.
API Calls	Implement caching and reduce redundant API calls.
Code Splitting	Split bundles to load only required components.
Page Speed Testing	Use tools like Lighthouse to analyze and improve.

5. Responsiveness

Ensure the website is responsive across all devices.

Device	Tested Viewports
Mobile Phones	375x667, 360x640
Tablets	768x1024, 800x1280
Laptops/Desktops	1024x768, 1920x1080

Tools for Testing:

- Browser Developer Tools
- Responsive Design Mode

6. Cross-Browser Compatibility

Verify the website functions properly across different browsers.

Browser	Version Tested
Google Chrome	Latest
Mozilla Firefox	Latest
Safari	Latest (Mac and iOS)
Microsoft Edge	Latest

7. Device Testing

Device testing ensures the application performs well on physical devices.

Test Type	Details
Device Testing	Test on real devices (phones, tablets, desktops).
User Acceptance Testing (UAT)	Get feedback from end-users to ensure usability.

Additional Notes

- **CSV Format Representation:** All the data above can be converted into CSV format for further analysis and tracking.

Topic	Detail
Functional Testing	Core features tested.
Error Handling	Added user-friendly error messages.
Security Improvements	Enhanced authentication and input validation.
Performance Optimization	Improved API calls and image optimization.
Responsiveness	Designed for multiple devices and viewports.
Cross-Browser Compatibility	Ensured proper rendering in popular browsers.
Device Testing	Tested on real devices and through UAT.

With these steps, Next.js e-commerce project will ensure robust functionality, security, and an excellent user experience.

Prepared By: Azmat Ali