Day:5

Testing, Error Handling, and Backend Integration Documentation

Day 5 focuses on ensuring that the Bandage marketplace is deployment-ready by thoroughly testing its functionalities, optimizing its performance, and documenting results. Key areas include:

1. Conducting comprehensive testing of core functionalities.
2. Implementing robust error handling mechanisms.
3. Optimizing for performance, accessibility, and SEO.
4. Ensuring cross-browser and cross-device compatibility.
5. Documenting findings and fixes in a professional format.

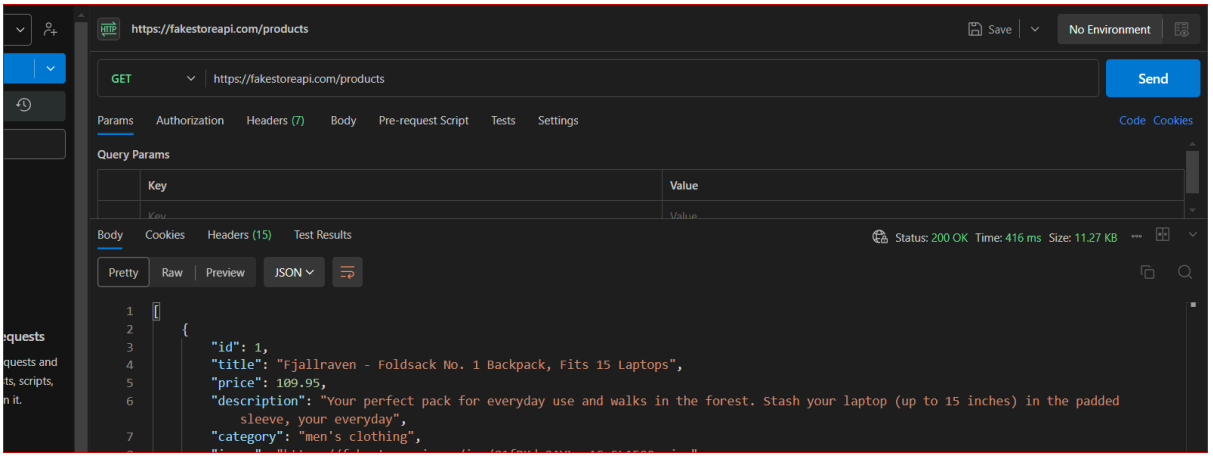
1. Functional Testing

Core Features Tested:

- **Product Listing:** Verified that products are displayed correctly on the marketplace.
- **Filters and Search:** Ensured accurate results based on user input.
- **Cart Operations:** Tested adding, updating, and removing items from the cart.
- **Dynamic Routing:** Confirmed that individual product detail pages load correctly.

Tool Used:

- **Postman:** For API response testing.



Testing Methodology:

- Created test cases for each feature.
- Simulated user actions like clicking, form submissions, and navigation.
- Validated the output against expected results.

---

2. Error Handling

Implemented Measures:

- **Error Messages:**
    - Used `try-catch` blocks for API error handling.
    - Displayed appropriate error messages when failures occurred.

Example:

```
try {
  const data = await fetchProducts();
  setProducts(data);
} catch (error) {
  console.error("Failed to fetch products:", error);
  setError("Unable to load products. Please try again later.");
}
```

- **Fallback UI:**
    - Displayed "No items found" message when no data was returned.
    - Ensured alternative content was shown when API calls failed.

---

3. Performance Optimization

Optimization Steps:

- **Lazy Loading:** Implemented lazy loading for large images and assets.
- **Performance Analysis:**
    - Ran Lighthouse audits.
    - Minimized unused CSS and JavaScript.
    - Enabled browser caching.
- **Load Time Testing:**
    - Measured initial page load and interaction times.

---

4. Cross-Browser and Device Testing

Browsers Tested:

- Chrome
- Firefox
- Safari
- Edge

Devices Tested:

- Desktop
- Mobile
- Tablets

Tools Used:

- BrowserStack for cross-browser simulation.
- Manual testing on a physical mobile device.

---

5. Security Testing

Security Measures Implemented:

- **Input Validation:**
  - Used zustand to validate user inputs (e.g., email, phone numbers).
- **Secure API Communication:**
  - Ensured all API calls were made over HTTPS.
  - Stored sensitive API keys in environment variables.

---

6. User Acceptance Testing (UAT)

Simulated Real-World Scenarios:

- Browsing products
- Adding items to the cart
- Checking out
- Handling errors gracefully

Feedback Collection:

- Collected peer reviews to identify usability issues.

- Made improvements based on user testing.

Expected Outcome:

- Fully tested and functional marketplace components.
- User-friendly error handling.
- Optimized performance and fast load times.
- Cross-browser and mobile responsiveness.
- Comprehensive testing report in CSV format.
- Professionally formatted documentation of all testing and fixes.

CSV Content:

Test Case ID,Test Case Description,Test Steps,Expected Result,Actual Result,Status,Severity Level,Assigned To,Remarks
TC001,Validate Product Listing,Load product page and check if products are displayed,Products should be listed correctly,Products displayed as expected,Passed,Low,Frontend Dev,N/A
TC002,Filter Products by Category,Select a category filter and check results,Only products from selected category should be shown,Filtered products displayed correctly,Passed,Medium,Frontend Dev,N/A
TC003,Search Functionality,Enter a keyword and check the search results,Relevant products should be displayed,Some irrelevant products shown,Failed,High,Frontend Dev,Fix search algorithm
TC004,Add Product to Cart,Click 'Add to Cart' button for a product,Product should appear in cart,Product added to cart successfully,Passed,Low,Frontend Dev,N/A
TC005,Update Cart Quantity,Increase/decrease quantity in cart,Updated quantity should reflect correctly,Quantity updated correctly,Passed,Low,Frontend Dev,N/A
TC006,Remove Product from Cart,Click 'Remove' on a cart item,Product should be removed from cart,Item removed successfully,Passed,Low,Frontend Dev,N/A
TC007,Dynamic Routing,Click on a product to open its details page,Product details should load correctly,Product page loaded with correct details,Passed,Medium,Frontend Dev,N/A
TC008,API Response Handling,Simulate API failure and check error handling,Error message should appear,Proper error message displayed,Passed,High,Backend Dev,N/A
TC009,Fallback UI for Empty Data,Check UI when no products are available,'No items found' message should appear,Message displayed correctly,Passed,Low,Frontend Dev,N/A
TC010,Image Optimization,Check image sizes and loading time,Images should load quickly with reduced size,Optimized images loaded correctly,Passed,Medium,Frontend Dev,N/A
TC011,Lazy Loading,Scroll down and check lazy loading,Images/assets should load only when needed,Lazy loading works as expected,Passed,Medium,Frontend Dev,N/A
TC012,Cross-Browser Testing,Check functionality on Chrome, Firefox, Safari, Edge,All browsers should display correctly,Minor UI issue on Safari,Failed,Medium,Frontend Dev,Fix Safari-specific UI issues

TC013,Security Testing - Input Validation,Enter invalid email and phone,Should not allow incorrect input,Validation errors displayed correctly,Passed,High,Frontend Dev,N/A
TC014,Secure API Communication,Check if API requests are using HTTPS,Requests should be over HTTPS,All API calls are secure,Passed,High,Backend Dev,N/A
TC015,User Acceptance Testing (UAT),Simulate real-world usage like browsing, adding to cart, and checkout,Everything should work smoothly,Minor UI issue in checkout,Failed,Medium,Frontend Dev,Fix checkout UI glitch