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

Marketplace Name: Ras Healthcare

Functional Testing

Core Functionalities Tested:

- **Product Listing Page**: Verified that products are displayed accurately with correct images, prices, and descriptions.
- **Search Functionality**: Tested search for keywords to ensure relevant products are retrieved.
- **Cart Operations**: Validated adding, updating, and removing items in the cart.
- **Checkout Process**: Ensured order placement works seamlessly, including customer data validation.

Tools Used:

- **Cypress**: For end-to-end testing of user flows.
- **Jest**: For unit testing individual components and backend logic.

Summary:

All major functionalities were tested. Bugs were identified in the search functionality (e.g., irrelevant results for specific terms) and have been logged for resolution.

Error Handling

Implemented Fallback Mechanisms:

1. **API Failure Handling:**
    - Displayed user-friendly error messages ("Unable to load data. Please try again later.").
    - Automatic retries for transient failures.
2. **Form Validation Errors:**

    - Real-time validation using `react-hook-form` and `zod`.
    - Error messages displayed below input fields for user clarity.
3. **Cart Issues:**
    - Graceful degradation in case of cart API unavailability with a "Save for Later" option.

Examples of Error Handling:

- **Search API Timeout:** Displayed a "Search is currently unavailable" message.
- **Invalid Coupon Code:** Shown "Invalid or expired coupon" warning.

Performance Testing

Tools Used:

- **Lighthouse**: For measuring page performance and identifying bottlenecks.
- **GTmetrix**: To optimize loading times and identify slow assets.

Key Optimizations:

1. Minimized image sizes using next/image with optimized caching.
2. Enabled server-side rendering (SSR) in Next.js for faster page loads.
3. Implemented lazy loading for non-critical components.

Cross-Browser and Device Testing

Browsers Tested:

- Google Chrome
- Mozilla Firefox
- Microsoft Edge
- Safari

Devices Tested:

- **Mobile:** Samsung Galaxy S21, iPhone 14
- **Desktop:** Windows 10, macOS Ventura

Tools Used:

- **BrowserStack:** For simulating multiple browsers and devices.
- Manual testing for key flows.

Security Testing

Actions Taken:

1. **API Security:** Implemented API key restrictions and HTTPS-only communication.
2. **Input Validation:** Sanitized all user inputs to prevent SQL injection and XSS attacks.
3. **Data Encryption:** Ensured sensitive data, like passwords, is securely encrypted using Appwrite's encryption features.

Summary:

All high-priority vulnerabilities were mitigated, and regular scans are scheduled for maintenance.

Testing Documentation

Report Format:

- **Generated CSV File:** Contains detailed test cases, steps, and results.
- **Screenshots:** To be added for API calls, frontend displays, and backend data validation.

Integration Refinement Notes:

- Backend integration was improved by utilizing Appwrite's SDK.
- API responses were optimized to include only necessary data, reducing payload size.
- Email notifications were configured via Resend for order confirmations.

Attachments:

- Screenshots (to be added):
    - API calls demonstrating successful data retrieval and error handling.
    - Frontend display of populated product, cart, and order data.

    - Backend (Appwrite) showing accurate data migration.
- Code Snippets (to be added):
    - API integration logic.
    - Error handling utilities.
    - Performance optimization scripts.