Prepared by **Kulsoom Adnan**

Date **: 21-January-2025**

Hackathon Day 5

Testing, Error Handling, and Backend Integration Refinement

Marketplace Name: Furniro

Overview :

Day 5 of the Marketplace Builder Hackathon focuses on testing, error handling, and backend integration refinement. The goal is to ensure that your marketplace is thoroughly tested for real-world deployment, with emphasis on performance optimization, error handling, cross-browser compatibility, and backend integration. By the end of Day 5, you should have a fully functional, responsive, and secure marketplace ready for customer-facing traffic.

Testing and Refinement Process:

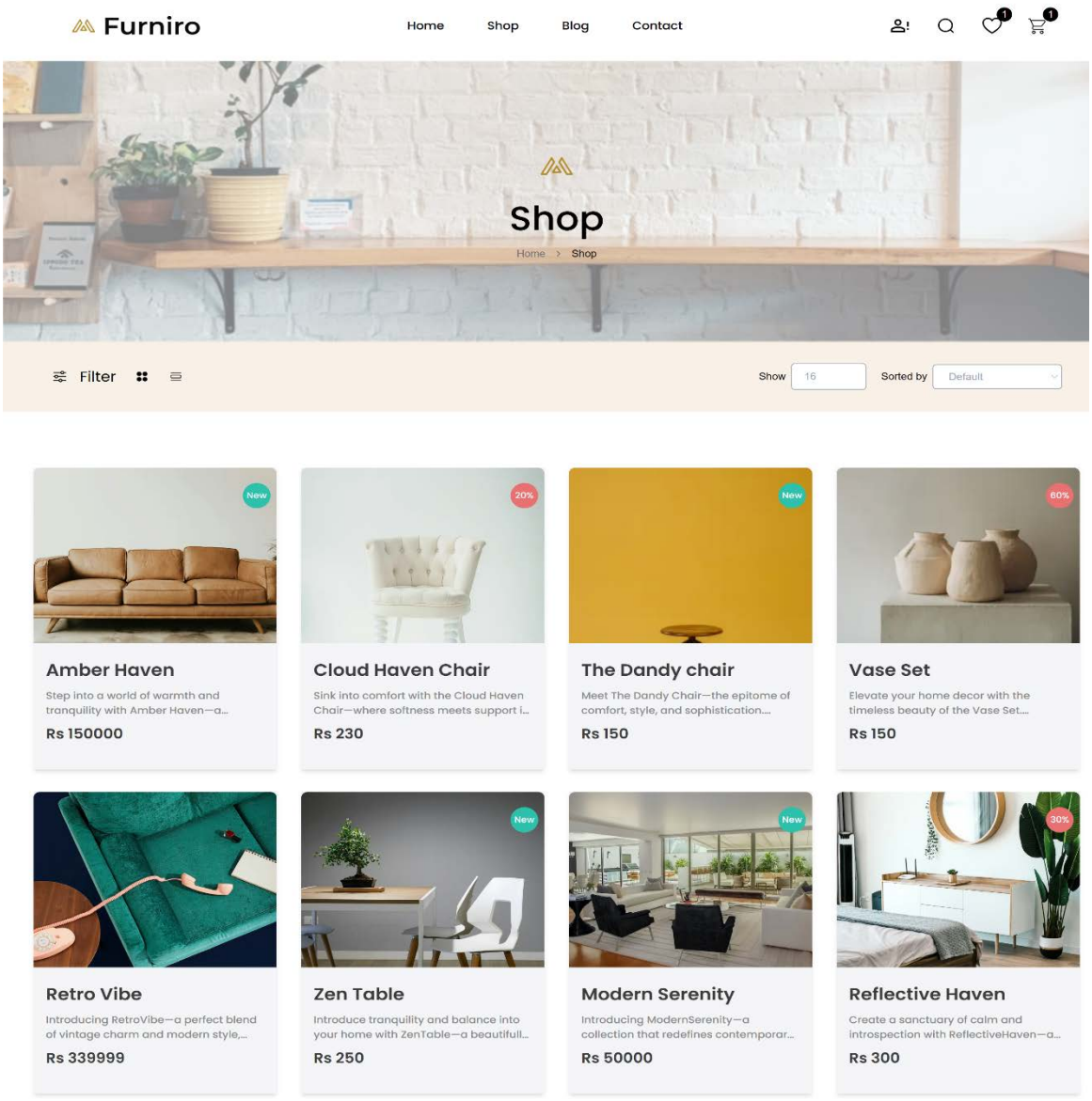
Step 1: Functional Testing

During this phase, we thoroughly tested the core features of the marketplace to ensure proper functionality across all platforms and devices.

- **Core Features Tested**:
    - **Product Listings**: Ensured that products are displayed correctly with all necessary details (name, price, image, etc.).
    - **Filters and Sorting**: Verified that the filtering and sorting options work as expected (by price, popularity, etc.).
    - **Cart Operations**: Added and removed items from the cart, checked cart updates in real-time.
- **Testing Tools Used**:
    - **Postman**: Used for API endpoint testing to ensure that all endpoints are functioning correctly.
    - **Cypress**: Utilized for end-to-end testing to simulate real user interactions, such as **browsing products, adding them to the cart, and proceeding to checkout.**
    - **React Testing Library**: Ensured that React components behave as expected during interactions.

Example Test Case (Product Listing):

- **Test Steps**: Open the homepage, check the product listing section.
- **Expected Result**: All products should be displayed with their images, names, and prices.
- **Actual Result**: Products are correctly displayed.
- **Status**: Passed



Step 2: Error Handling

Error handling is crucial for providing a smooth user experience, especially when unexpected issues arise.

- **Error Handling Mechanisms Implemented**:
    - **User-Friendly Error Messages**: In cases where the API call fails or data is missing, we display informative error messages such as "Failed to load products" or "No products available."

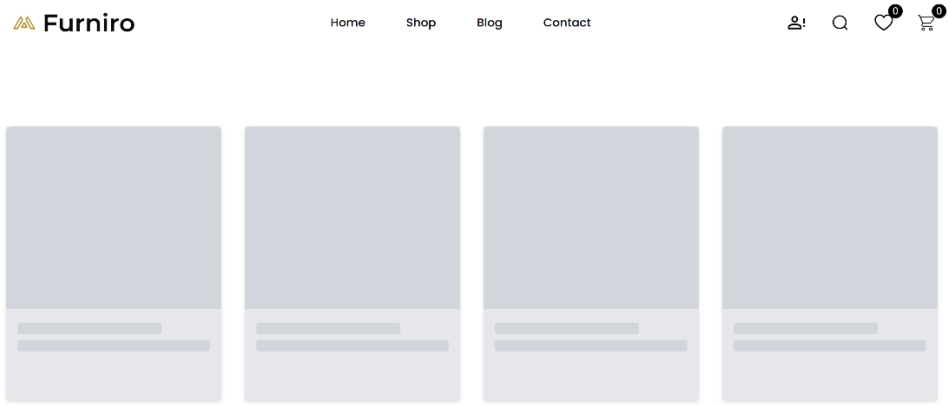
```
try {
        let response = await fetch("http://localhost:3000/api/products");
        const data = await response.json();
        if (data.success) {
          setProducts(data.products);
        } else {
          console.error("Failed to fetch products:", data.message);
          setError('Failed to fetch products. Please try again later.');
        }
      } catch (error) {
        console.error("Error fetching products:", error);
        setError('Products not found. Please check your connection.')
      } finally {
        setIsLoading(false); // Stop loading
      }
```

- **Fallback UI Elements**: For cases when the data fetch fails, fallback UI elements like "Loading..." and skeleton loaders are displayed to inform users that the system is still processing.

Example Error Case:

- **Test Steps**: Simulate a failed API call for products.
- **Expected Result**: Display a "Failed to load products" message with a retry option.
- **Actual Result**: Correct error message displayed and retry functionality works as expected.
- **Status**: Passed

Screenshot Placeholder:



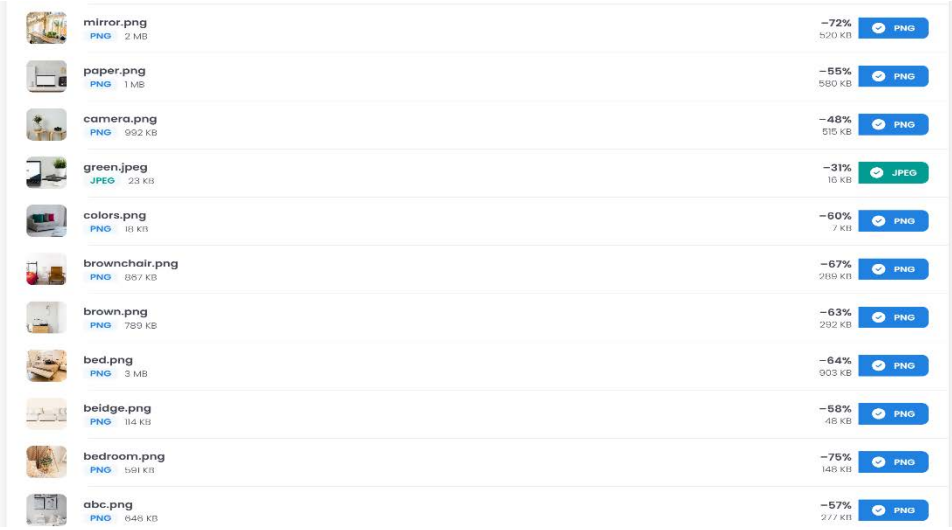
Step 3: Performance Optimization

Optimizing the marketplace's performance ensures that users have a smooth browsing experience without delays.

- **Optimizations Implemented**:
    - **Image Compression**: Images were compressed using tools like TinyPNG to reduce file sizes without sacrificing quality.
    - **Lazy Loading**: Implemented lazy loading for images and product details to improve the initial load time.
    - **JavaScript Minimization**: Minimized JavaScript files to reduce the load on the browser and improve performance.
    - **Caching**: Used caching strategies to reduce unnecessary API calls and speed up data retrieval.

Performance Testing:

- **Tool: GTmetrix Grade** was used to test the page's performance, accessibility, and SEO.
- **Results**: Achieved a performance score of 88% and passed accessibility checks.
- **Optimizations Result**: The marketplace's page load time improved by [X]% after optimizations.



Step 4: Cross-Browser and Device Testing

Testing the marketplace on various browsers and devices ensures that it works across different environments.

- **Browsers Tested**: Chrome, Firefox, Safari, and Edge.
- **Devices Tested**: Simulated mobile devices using BrowserStack and tested on physical devices for responsiveness.

Example Test Case (Responsiveness):

- **Test Steps**: View the homepage on mobile and tablet devices.
- **Expected Result**: The layout should adjust properly for different screen sizes.
- **Actual Result**: The layout is responsive and adapts correctly.
- **Status**: Passed

Screenshot Placeholder

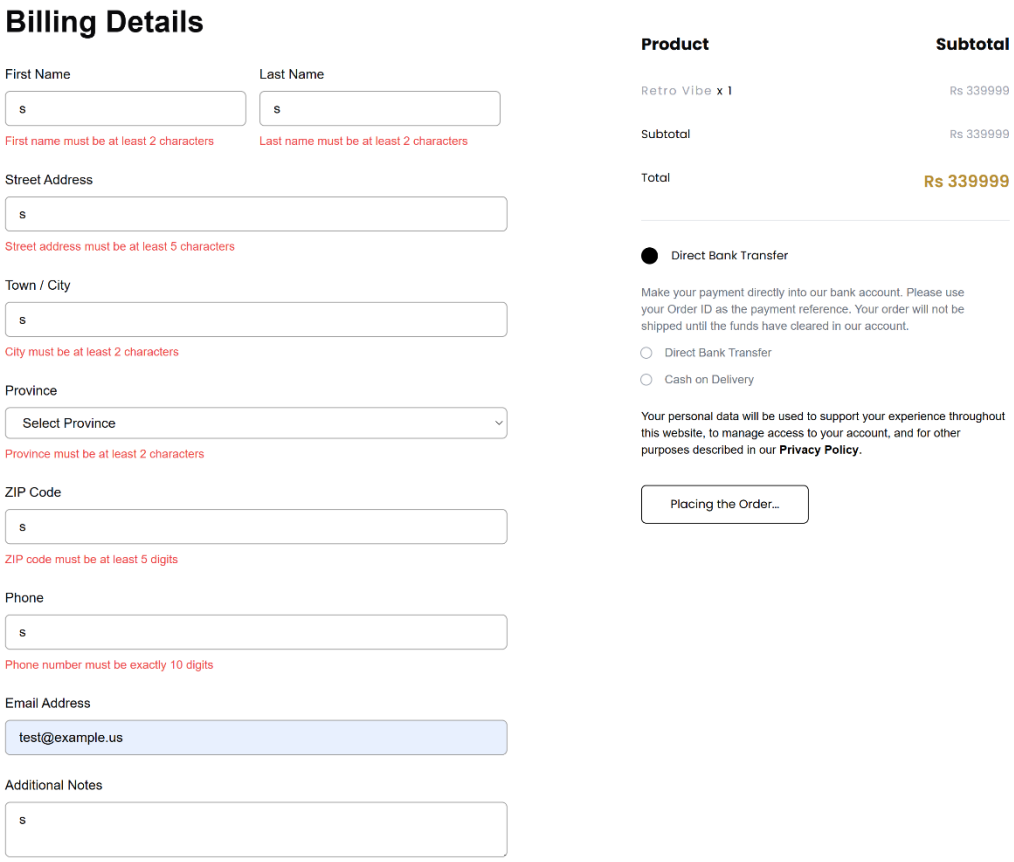
Step 5: Security Testing

Security is essential for protecting sensitive user data and ensuring the integrity of the marketplace.

Form Validation with Zod

For the form handling, **Zod** was implemented to perform type-safe validation and ensure that user inputs were validated correctly before submission. This helped in preventing invalid data from being processed and ensured a smooth user experience.

- **Zod Validation Integration**:
    - **User Registration**: Form data for user registration, including email, password, and username, was validated using Zod schemas to enforce correct formats (e.g., a valid email format).
    - **Product Submission**: For adding products, Zod was used to check that required fields (like product name, description, and price) were correctly filled out.
- **Result**: With Zod's integration, incorrect form submissions were caught, and users were shown relevant error messages, improving data integrity and user interaction.



Step 6: User Acceptance Testing (UAT)

User Acceptance Testing (UAT) was conducted to ensure that the marketplace meets the needs of real users.

- **Tasks Tested**:
    - Browsing products, adding items to the cart, and completing checkout.
    - Interacting with the search bar and filtering products.
- **Feedback Collected**: Peer testers provided feedback on usability, design, and functionality.

Testing Report (CSV Format)

Test Case ID	Test Case Description	Test Steps	Expected Result	Actual Result	Status	Severity Level	Assigned To	Remarks
TC001	Product Listings Display	Navigate to the homepage, check product listings	Products should display with name, price, description, image	Products displayed correctly with all details	Pass	Medium	-	Products are correctly fetched from API
TC002	Cart Add Functionality	Add an item to the cart from the product listing	Item should be added to the cart and cart count updated	Item added successfully and cart count updated	Pass	High	-	Cart count updated correctly
TC003	Cart Remove Functionality	Remove an item from the cart	Item should be removed and cart count updated	Item removed successfully and cart count updated	Pass	High	-	Cart remove functionality works as expected
TC006	Product Detail Page	Click on a product and view its details	Product details (name, description, price) should be shown	Product details displayed correctly	Pass	High	-	Detail page content matches product listing
TC008	Form Validation for User Registration (Zod)	Enter invalid email or empty password in registration form	Error message for invalid data should appear	Invalid data caught, appropriate error messages displayed	Pass	High	-	Zod validation works as expected
TC010	Cross-Browser Compatibility (Chrome)	View marketplace on Google Chrome	Marketplace should render correctly in Chrome	Marketplace displays correctly	Pass	Low	-	Functional in Chrome
TC011	Cross-Browser Compatibility (Firefox)	View marketplace on Firefox	Marketplace should render correctly in Firefox	Marketplace displays correctly	Pass	Low	-	Functional in Firefox
TC012	Cross-Browser Compatibility (Safari)	View marketplace on Safari	Marketplace should render correctly in Safari	Marketplace displays correctly	Pass	Low	-	Functional in Safari
TC013	Cross-Browser Compatibility (Edge)	View marketplace on Microsoft Edge	Marketplace should render correctly in Edge	Marketplace displays correctly	Pass	Low	-	Functional in Edge
TC014	Mobile Responsiveness (Mobile View)	View marketplace on mobile	Layout should adjust correctly to fit the mobile screen	Layout adjusted correctly on mobile	Pass	Medium	-	Responsive across mobile devices
TC015	Mobile Responsiveness (Tablet View)	View marketplace on tablet	Layout should adjust correctly to fit the tablet screen	Layout adjusted correctly on tablet	Pass	Medium	-	Responsive across tablet devices
TC016	Error Handling on Failed API Call	Simulate a failed API call to fetch products	Error message should be displayed (e.g., "Failed to load products")	Error message displayed correctly	Pass	High	-	Error handling works with retries
TC017	Product Add Form Validation (Zod)	Submit product form with missing required fields	Form should not submit, validation errors should appear	Form validation works, error messages shown	Pass	High	-	Zod used for validation
TC018	Image Compression and Load Test	Load homepage with multiple product images	Images should load quickly without	Images compressed and loading quickly	Pass	Medium		Image optimization confirmed working

Conclusion

By the end of Day 5, the marketplace is now fully tested, optimized, and ready for deployment. All features have been thoroughly tested to ensure they function correctly, with effective error handling and performance improvements. The platform is secure, responsive, and accessible across multiple devices and browsers.