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

Objective:

Prepare the marketplace for real-world deployment by ensuring thorough testing, robust error handling, and performance optimization.

Key Tasks:

1. Functional Testing

Test core functionalities:

- Product listing, filters, search, cart operations, and dynamic routing.
  Tools: Cypress, React Testing Library, Postman.
  Write and execute test cases, simulate user actions, and validate expected results.

2. Error Handling

Implement error handling with `try-catch` blocks.
Display user-friendly fallback UI for API failures (e.g., "No products available").

3. Performance Optimization

Optimize images using TinyPNG/ImageOptim.
Implement lazy loading and minimize JavaScript/CSS.
Run Lighthouse and GTmetrix audits to fix bottlenecks.

4. Cross-Browser & Device Testing

Test on Chrome, Firefox, Safari, and Edge.
Use BrowserStack/LambdaTest for responsive testing.
Manually test on at least one mobile device.

5. Security Testing

Validate inputs to prevent SQL injection/XSS attacks.
Ensure API communication is secure (use HTTPS, hide API keys in `.env`).

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

Objective:

Prepare the marketplace for real-world deployment by ensuring thorough testing, robust error handling, and performance optimization.

Key Tasks:

1. Functional Testing

✅Test core functionalities:

- Product listing, filters, search, cart operations, and dynamic routing.
  ✅Tools: Cypress, React Testing Library, Postman.
  ✅Write and execute test cases, simulate user actions, and validate expected results.

2. Error Handling

✅Implement error handling with `try-catch` blocks.
✅Display user-friendly fallback UI for API failures (e.g., "No products available").

3. Performance Optimization

✅Optimize images using TinyPNG/ImageOptim.
✅Implement lazy loading and minimize JavaScript/CSS.
✅Run Lighthouse and GTmetrix audits to fix bottlenecks.

4. Cross-Browser & Device Testing

✅Test on Chrome, Firefox, Safari, and Edge.
✅Use BrowserStack/LambdaTest for responsive testing.
✅Manually test on at least one mobile device.

5. Security Testing

✅Validate inputs to prevent SQL injection/XSS attacks.
✅Ensure API communication is secure (use HTTPS, hide API keys in `.env`).
✅Run security scans using OWASP ZAP/Burp Suite.

6. User Acceptance Testing (UAT)

✅Simulate real-world scenarios (browsing, adding to cart, checkout).
✅Collect feedback and refine usability.

7. Documentation & Submission

Create a CSV-based testing report with test cases, results, and fixes.
Submit a professional report (PDF/Markdown) summarizing testing efforts.
Upload the updated repository with a README detailing testing steps.

Final Deliverables:

Fully tested and optimized marketplace.
Error handling with clear messages and fallback UI.
Performance improvements for faster load times.
Responsive and cross-browser tested application.
Well-documented test reports and resolutions.

Run security scans using OWASP ZAP/Burp Suite.

6. User Acceptance Testing (UAT)

Simulate real-world scenarios (browsing, adding to cart, checkout).
Collect feedback and refine usability.

7. Documentation & Submission

Create a CSV-based testing report with test cases, results, and fixes.
Submit a professional report (PDF/Markdown) summarizing testing efforts.
Upload the updated repository with a README detailing testing steps.

Final Deliverables:

Fully tested and optimized marketplace.
Error handling with clear messages and fallback UI.
Performance improvements for faster load times.
Responsive and cross-browser tested application.
Well-documented test reports and resolutions.