Report on Testing, Error Handling, and Backend Integration for My Website

Report on Testing, Error Handling, and Backend Integration for My Website

1. Functional Testing

Unit Testing: Test small components like buttons and forms using Jest.

Integration Testing: Verify frontend-backend interactions, like adding products to the cart.

End-to-End Testing: Simulate user flows using Cypress (e.g., search and checkout).

Performance Testing: Use Lighthouse to check website speed and responsiveness.

Functional Testing: Ensure that individual features of the website, such as login, registration, or checkout, function as expected by testing each feature's behavior and output.

---

2. Error Handling

Frontend Errors: Use try-catch blocks for API calls and show user-friendly error messages (e.g., "Failed to load products.").

Global Error Handling: Implement React Error Boundaries to catch unexpected errors.

Backend Errors: Ensure clear API error responses (e.g., 404 for "Product Not Found").

Monitoring: Use tools like Sentry for error tracking.

---

3. Backend Integration

Secure APIs: Use RESTful APIs made over HTTPS.

Data Validation: Ensure valid inputs from the frontend (e.g., valid price or quantity).

Optimized Queries: Use Sanity queries for efficient product fetching.

Caching: Implement caching for frequently requested data like product lists.

API Testing: Test API endpoints using Postman.

---

4. Deployment and Monitoring

Automate testing and deployment with GitHub Actions.