## **🧱 System-Wide Error Handling & QA Feedback Hooks**

### **🎯 Objective**

Ensure that **every major user-facing module** fails **gracefully, visibly, and traceably** — across both desktop and mobile views. Build a **feedback-first system** where bugs can be reported, logged, and traced *without breaking the user journey.*

### **🔹 1. ERROR HANDLING (SYSTEM-WIDE)**

#### **🔸 Purpose**

Prevent silent failures. Replace blank screens, JSON errors, or console-only errors with proper **UI feedback** and **logs for debugging.**

#### **🔸 Key Requirements**

| **Area** | **What Should Happen** |
| --- | --- |
| **404s** | Show a branded “Page Not Found” with a soft prompt to return to dashboard. |
| **API Failures** | If any fetch (CRM, Dispatch, etc.) fails, show a **friendly toast** or error block: *“We’re having trouble loading this right now. Try again or contact Admin.”* |
| **UI-level Errors** | Forms should never freeze. If a form submit fails, show: *“Something went wrong. Please review inputs or retry.”* |
| **Empty States** | If the DB returns no results, show “No Data Found” with a tip (e.g., “Click + to add new”) instead of white space. |
| **Auth/Session Expiry** | If session expires, redirect to login with message: *“You’ve been logged out for security. Please log in again.”* |
| **Mobile Handling** | All error components should **scale responsively**. Never cut off or overflow. |
| **Console Logging** | Errors should be logged in console with endpoint, payload (redacted), and response code. |
| **Loading States** | Add visible spinners or “Loading...” on all list fetches, forms, and graphs. No UI should just hang. |

### **🔹 2. QA FEEDBACK & ISSUE REPORTING**

#### **🔸 Goal**

Allow **any user** (Admin, Team Member, Tester) to **report bugs instantly** without leaving the app — and track those reports centrally.

#### **🔸 Key UX Flow**

1. **Sticky Button (Bottom-Right Corner)**
   * Label: 🐞 Report Issue
   * Shown on every page **except login and error pages**
   * Clicking opens a small **modal or drawer**
2. **Bug Report Modal Fields:**
   * **Page Auto-Captured** (e.g., /dispatch/loads)
   * **Issue Description** (textarea)
   * **Screenshot Upload (optional)** *(Optional, but must accept drag-drop or mobile picker)*
   * **Urgency Tag** (Low, Medium, High)
   * **Submit** button
3. **After Submit:**
   * Show “Thank you, we’ve logged your issue. Our team will review it shortly.”
   * Send the issue to:  
     + bugs table in DB
     + (Optional) Slack webhook or email if enabled in admin panel
4. **Admin Access to View Reported Bugs:**
   * Admin Module → Tab: Bug Reports
   * List view of: Page, Description, Screenshot, Submitted by, Timestamp, Status

### **🔹 3. CONNECTING TO EXISTING SYSTEM**

| **Area** | **Connection** |
| --- | --- |
| CRM, Dispatch, Invoices | All must have fetch try/catch + visible fallback state. |
| Login & Auth | Expiry = redirect to login with proper message. |
| QA Module | Bugs reported via feedback flow can be linked to QA tickets. |
| Admin Panel | Bug Reports tab can be used during sprints to plan fix cycles. |

### **🧪 Suggested QA Tests**

| **Test** | **Expected Result** |
| --- | --- |
| Break /crm/leads API | Should show soft error message, not JSON dump |
| Go to wrong URL | Should land on 404 page with return home button |
| Try to add a lead with missing fields | Show field errors, not a freeze |
| Submit bug via button | Should store in DB and confirm |
| Collapse form mid-use on mobile | Should restore state or reset cleanly |

### **✅ What to TELL Replit (no code)**

“Hey Replit, we’re moving into the **System-Wide Error Handling and Bug Reporting** step. This involves:

* Creating global error handling components for all pages (not individual fixes)
* Ensuring no fetch ever fails silently — use branded UI error messages
* Adding a Report Bug button globally that opens a modal for users to submit issues with optional screenshot and urgency tag
* This should be connected to a bugs table or JSON log, and an admin panel view to track
* DO NOT overwrite working components — just wrap fetches and add fallbacks
* Mobile compatibility must be tested  
   Let’s do this modularly and not touch the current working CRM/Dispatch form logic unless needed. Just add clean wrappers and fallback states.”