# Error Handling

## Error Handling Patterns

**1. Server Actions**:

// app/actions/initiatives.ts  
'use server'  
  
import { z } from 'zod'  
  
export async function createInitiative(formData: FormData) {  
 try {  
 // Validate  
 const parsed = initiativeSchema.safeParse(Object.fromEntries(formData))  
 if (!parsed.success) {  
 return {  
 success: false,  
 error: {  
 type: 'validation',  
 message: 'Invalid input',  
 issues: parsed.error.flatten(),  
 },  
 }  
 }  
  
 // Insert  
 const { data, error } = await supabase  
 .from('initiatives')  
 .insert(parsed.data)  
 .select()  
 .single()  
  
 if (error) {  
 logger.error({ error }, 'Database error creating initiative')  
 return {  
 success: false,  
 error: {  
 type: 'database',  
 message: 'Failed to create initiative',  
 },  
 }  
 }  
  
 revalidatePath('/dashboard')  
 return { success: true, data }  
  
 } catch (error) {  
 logger.error({ error }, 'Unexpected error creating initiative')  
 Sentry.captureException(error)  
 return {  
 success: false,  
 error: {  
 type: 'unknown',  
 message: 'An unexpected error occurred',  
 },  
 }  
 }  
}

**2. Client Error Boundaries**:

// app/error.tsx  
'use client'  
  
import { useEffect } from 'react'  
import \* as Sentry from '@sentry/nextjs'  
  
export default function Error({  
 error,  
 reset,  
}: {  
 error: Error & { digest?: string }  
 reset: () => void  
}) {  
 useEffect(() => {  
 Sentry.captureException(error)  
 }, [error])  
  
 return (  
 <div className="flex min-h-screen items-center justify-center">  
 <div className="text-center">  
 <h2 className="text-2xl font-bold">Something went wrong!</h2>  
 <button  
 onClick={() => reset()}  
 className="mt-4 rounded bg-blue-600 px-4 py-2 text-white"  
 >  
 Try again  
 </button>  
 </div>  
 </div>  
 )  
}

**3. User-Friendly Error Messages**:

// lib/errors/messages.ts  
export const ERROR\_MESSAGES = {  
 validation: {  
 required: 'This field is required',  
 minLength: (min: number) => `Must be at least ${min} characters`,  
 email: 'Invalid email address',  
 },  
 database: {  
 duplicate: 'A record with this value already exists',  
 notFound: 'Record not found',  
 constraint: 'Cannot delete: record is referenced elsewhere',  
 },  
 auth: {  
 unauthorized: 'You do not have permission to perform this action',  
 unauthenticated: 'Please log in to continue',  
 },  
} as const