ErrorBoundary UI States

CodeScribe AI - Production-Ready Error Handling Component

Component: `ErrorBoundary.jsx` | Tests: 48 passing | Date: October 14, 2025

Overview

The ErrorBoundary component provides a robust safety net for the entire React application, catching JavaScript errors anywhere in the component tree and displaying a user-friendly fallback UI instead of crashing the app. It features intelligent error handling with different modes for development and production environments.

🎯 Core Features

- Catches all React rendering errors
- Prevents app crashes
- User-friendly error messages
- Multiple recovery options
- Error count tracking

🔧 Development Mode

- Detailed error messages
- Full stack traces
- Component stack display
- Expandable technical details
- Enhanced debugging info

🚀 Production Mode

- Clean, professional UI
- Unique error IDs
- No sensitive info exposed
- Support contact info
- User-friendly guidance

1. Development Mode UI

When running in development (`import.meta.env.DEV === true`), the ErrorBoundary displays detailed technical information to help developers debug issues quickly.

Something went wrong

The application encountered an unexpected error

We're sorry for the inconvenience. The application ran into a problem and couldn't continue. Here's what you can try:

- Try again - sometimes temporary issues resolve themselves
- Reload the page to start fresh
- Check your internet connection
- Clear your browser cache if the problem persists

Try Again | Reload Page | Go Home

▼ **Technical Details (Development Mode)**

Error Message:

```
Error: Cannot read properties of undefined (reading 'map')
```

Stack Trace:

```
at DocumentationPanel (src/components/DocPanel.jsx:42:15)
at div
at App (src/App.jsx:120:5)
at ErrorBoundary (src/components/ErrorBoundary.jsx:86:16)
```

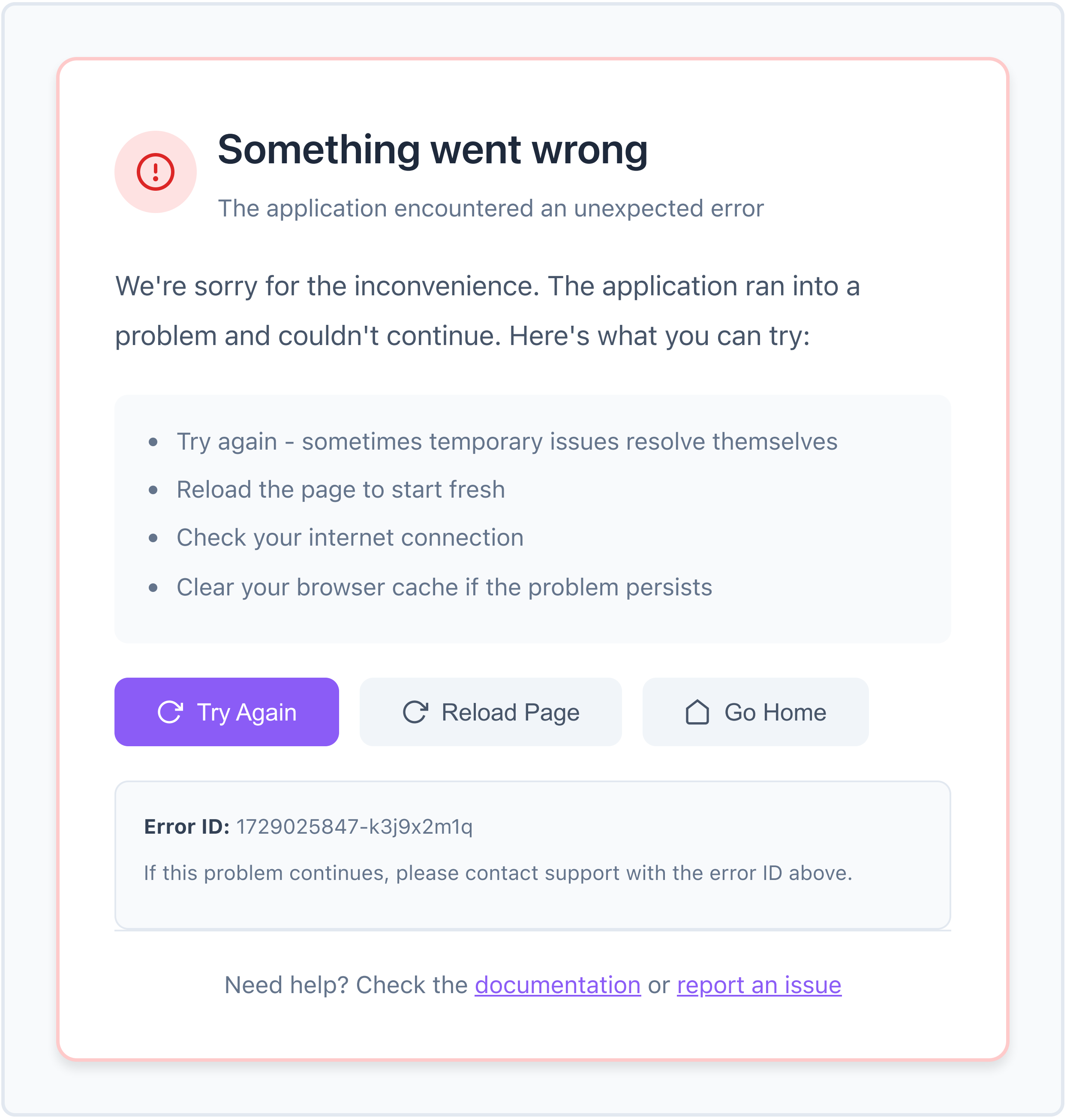
Component Stack:

```
in DocumentationPanel (at App.jsx:42)
in div (at App.jsx:38)
in App (at main.jsx:10)
in ErrorBoundary (at main.jsx:9)
```

Need help? Check the documentation or report an issue

2. Production Mode UI

When running in production (`import.meta.env.DEV === false`), the ErrorBoundary hides technical details and shows a clean, user-friendly interface with a unique error ID.



Something went wrong

The application encountered an unexpected error

We're sorry for the inconvenience. The application ran into a problem and couldn't continue. Here's what you can try:

- Try again - sometimes temporary issues resolve themselves
- Reload the page to start fresh
- Check your internet connection
- Clear your browser cache if the problem persists

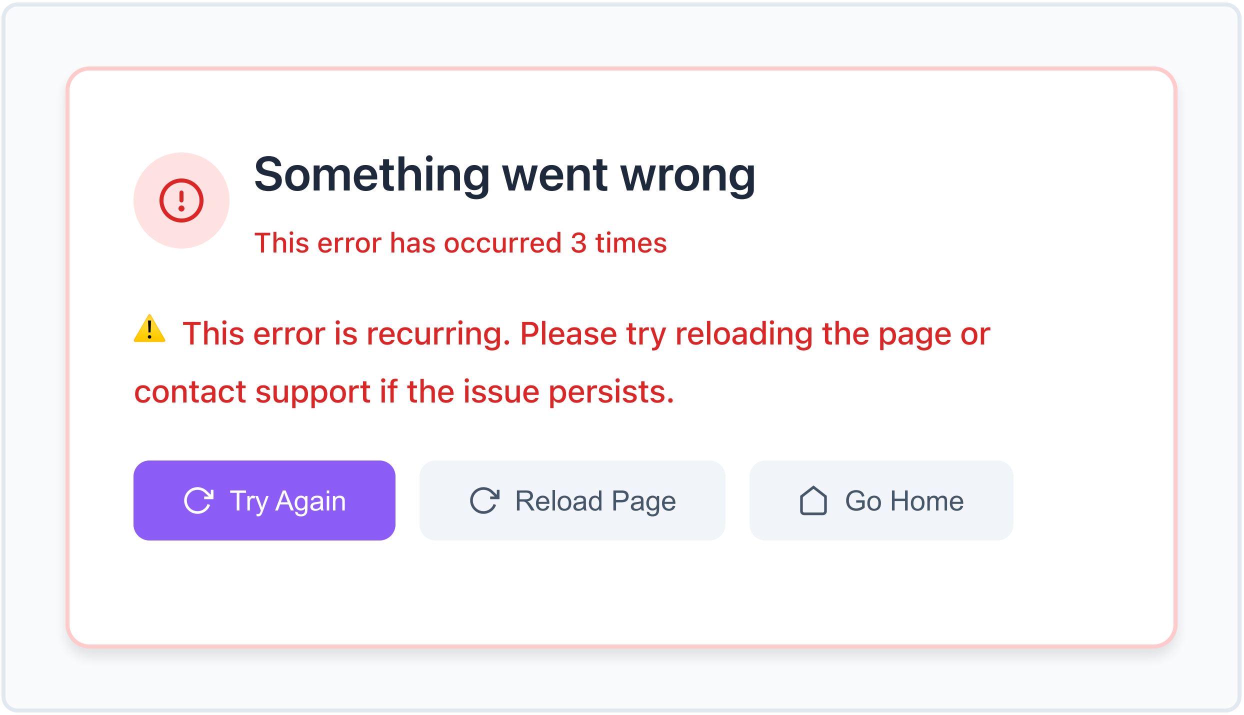
Error ID: 1729025847-k3j9x2m1q

If this problem continues, please contact support with the error ID above.

Need help? Check the documentation or report an issue

3. Multiple Errors Tracking

The ErrorBoundary tracks how many times an error has occurred, helping identify persistent issues and providing better context to users and developers.



Something went wrong

This error has occurred 3 times

⚠️ This error is recurring. Please try reloading the page or contact support if the issue persists.

4. Development vs Production Comparison

Key differences between development and production error displays to optimize for debugging vs user experience.

Feature	Development Mode	Production Mode
Error Message Display	✓ Full error message	✓ User-friendly message
Stack Trace	✓ Visible in collapsible section	❌ Hidden (security)
Component Stack	✓ Full component tree	❌ Hidden (security)
Error ID	❌ Not shown	✓ Unique ID for support
Technical Details	✓ Expandable section with all details	❌ Hidden
Recovery Actions	✓ Try Again, Reload, Go Home	✓ Try Again, Reload, Go Home
User Guidance	✓ Helpful suggestions	✓ Helpful suggestions
Console Logging	✓ Detailed logs	✓ Basic logs (can send to monitoring)
Support Information	✓ Documentation links	✓ Support contact with error ID

5. Recovery Actions

Three recovery options provide users with different ways to resolve errors and continue using the application.

🔄 Try Again

- Resets error state
- Re-renders child components
- Clears error details
- Preserves application state
- Best for temporary issues

Use case: Network glitches, race conditions, temporary API failures

🔃 Reload Page

- Full page refresh
- Clears all state
- Reloads all resources
- Most aggressive reset
- Guaranteed fresh start

Use case: Persistent errors, corrupted state, multiple error occurrences

🏠 Go Home

- Navigates to root path
- Clears current route state
- Returns to safe state
- Keeps app running
- Gentle recovery option

Use case: Route-specific errors, deep-link issues, broken feature pages

6. Technical Implementation

The ErrorBoundary uses React's error boundary lifecycle methods and modern error handling patterns.

📦 Component Structure

- **Class Component:** Uses React.Component for error boundary lifecycle
- **State Management:** Tracks hasError, error, errorInfo, errorCount
- **Lifecycle Methods:**
  - `getDerivedStateFromError()` - Updates state to trigger fallback UI
  - `componentDidCatch()` - Logs error details and updates state
- **Recovery Methods:** handleReset(), handleReload(), handleGoHome()
- **Environment Detection:** Uses import.meta.env.DEV to switch modes

✅ Test Coverage (48 Tests)

- Normal rendering without errors
- Error catching and display
- Error count tracking
- Recovery action functionality
- Development vs Production modes
- Styling and layout verification
- Accessibility (WCAG AA compliant)
- Edge cases (null children, long errors)
- Responsive design
- Keyboard navigation

7. Integration & Usage

The ErrorBoundary is integrated at the root level to catch errors from the entire application tree.

main.jsx Implementation

```
import { StrictMode } from 'react'
import { createRoot } from 'react-dom/client'
import './index.css'
import App from './App.jsx'
import ErrorBoundary from './components/ErrorBoundary.jsx'

createRoot(document.getElementById('root')).render(
  <StrictMode>
    <ErrorBoundary>
      <App />
    </ErrorBoundary>
  </StrictMode>,
)
```

💡 **Best Practice:** Place ErrorBoundary as high as possible in the component tree to catch errors from all child components. Consider multiple boundaries for different sections if you need granular error handling.

Summary

The ErrorBoundary component provides production-ready error handling with a focus on user experience and developer debugging capabilities.

48 Tests Passing	**3** Recovery Options
2 Display Modes	**100%** Coverage

CodeScribe AI • October 14, 2025

Component Location: client/src/components/ErrorBoundary.jsx