Perfect! This is **exactly** what we needed. I can now pinpoint the issue. Let me break down what's happening:

**🎯 Root Cause Identified**

**The Problem:**

**File:** GalleryCard.tsx:25  
**Error:** Cannot destructure property 'trek' of 'undefined'

Your GalleryCard component is receiving **undefined** props when it tries to destructure { trek, onClick }. This is causing:

1. The component to throw an error
2. React to attempt recovery (re-render)
3. The same undefined data is passed again
4. Infinite error → recovery → error loop
5. Eventually: **Maximum call stack exceeded**

**The Stack Pattern Shows:**

text

mu (react-dom) → gu (react-dom) → mu → gu → mu → gu...

This is React's reconciliation loop getting stuck because:

* PublicGallery.tsx:426 and PublicGallery.tsx:447 are triggering renders
* GalleryCard keeps failing to render
* React keeps trying to recover

**🔧 The Fix**

**Option 1: Add Defensive Checks in GalleryCard.tsx (Quick Fix)**

**Current code (line ~25):**

tsx

*// GalleryCard.tsx*

**const** GalleryCard = ({ trek, onClick }) => {

*// Component tries to use trek immediately*

**return** <div onClick={() => onClick(trek)}>...</div>

}

**Fix - Add null check:**

tsx

*// GalleryCard.tsx*

**const** GalleryCard = ({ trek, onClick }) => {

*// Guard against undefined trek*

**if** (!trek) {

console.warn('GalleryCard: trek is undefined');

**return** **null**; *// or return a placeholder card*

}

**return** <div onClick={() => onClick(trek)}>...</div>

}

**Option 2: Fix the Data Source in PublicGallery.tsx (Root Fix)**

The real issue is in **PublicGallery.tsx** at lines **426** and **447** where you're mapping/rendering the gallery cards.

**Look for code like this:**

tsx

*// PublicGallery.tsx:426 or 447*

{treks.map((trek) => (

<GalleryCard trek={trek} onClick={handleClick} />

))}

**The problem:** treks array contains undefined or null items.

**Fix - Filter out undefined items:**

tsx

*// PublicGallery.tsx*

{treks

.filter(trek => trek != **null**) *// Remove undefined/null items*

.map((trek) => (

<GalleryCard key={trek.id} trek={trek} onClick={handleClick} />

))}

**Option 3: Fix the Data Fetching (Best Fix)**

**Find where you fetch/set the treks data:**

tsx

*// PublicGallery.tsx - likely in useEffect around line 426/447*

**const** fetchTreks = **async** () => {

**const** { data, error } = **await** supabase

.from('treks')

.select('\*');

setTreks(data); *// ❌ If data is null/contains nulls, this breaks*

}

**Fix - Clean the data before setting:**

tsx

**const** fetchTreks = **async** () => {

**const** { data, error } = **await** supabase

.from('treks')

.select('\*');

**if** (error) {

console.error('Error fetching treks:', error);

setTreks([]); *// Set empty array on error*

**return**;

}

*// Filter out any null/undefined items and validate required fields*

**const** validTreks = (data || []).filter(trek =>

trek && trek.id && trek.name *// Add required fields*

);

setTreks(validTreks);

}

**📋 Step-by-Step Implementation**

**Step 1: Quick Fix (Immediate - stops the crash)**

**File: src/components/GalleryCard.tsx** (or wherever GalleryCard is defined)

Add this at the **very start** of the component:

tsx

**const** GalleryCard = ({ trek, onClick }) => {

*// IMMEDIATE FIX: Guard against undefined*

**if** (!trek) {

console.error('GalleryCard received undefined trek');

**return** **null**;

}

*// Rest of your component code...*

}

**Step 2: Fix Data Source (Proper fix)**

**File: src/pages/PublicGallery.tsx**

Find the section around **line 426 and 447** (where the error originates). Look for:

1. **Where treks are mapped:**

tsx

*// BEFORE (around line 426 or 447)*

{treks.map((trek) => (

<GalleryCard trek={trek} onClick={handleClick} />

))}

tsx

*// AFTER - Add filter*

{treks

.filter(trek => trek != **null**)

.map((trek) => (

<GalleryCard key={trek.id} trek={trek} onClick={handleClick} />

))}

1. **Where treks are fetched/set:**

tsx

*// Find this pattern*

**const** fetchGalleryData = **async** () => {

**const** { data, error } = **await** supabase...

setTreks(data); *// ❌ PROBLEM HERE*

}

tsx

*// FIX IT*

**const** fetchGalleryData = **async** () => {

**try** {

**const** { data, error } = **await** supabase

.from('treks')

.select('\*');

**if** (error) **throw** error;

*// Clean and validate data*

**const** validTreks = (data || []).filter(trek =>

trek &&

trek.id &&

trek.name *// Add other required fields*

);

setTreks(validTreks);

} **catch** (error) {

console.error('Error fetching gallery:', error);

setTreks([]); *// Set empty array on error*

}

}

**Step 3: Check Initial State**

**In PublicGallery.tsx**, find where you initialize treks state:

tsx

*// BEFORE - might be undefined initially*

**const** [treks, setTreks] = useState();

tsx

*// AFTER - always start with empty array*

**const** [treks, setTreks] = useState([]);

**🔍 Why This Happened**

Based on the stack trace:

1. **PublicGallery.tsx:426/447** triggered a render with treks data
2. One or more items in the treks array was undefined
3. **GalleryCard.tsx:25** tried to destructure { trek } from undefined
4. This threw an error
5. React's error boundary tried to recover
6. The same bad data was still there
7. Loop: error → recovery → error → recovery → **STACK OVERFLOW**

**🧪 Test After Fix**

After applying fixes:

1. **Clear browser cache** and reload
2. **Open DevTools Console**
3. **Navigate to /gallery**
4. You should see:
   * ✅ No "Cannot destructure property 'trek'" errors
   * ✅ No infinite mu/gu React loops
   * ✅ Gallery loads normally
   * ⚠️ Maybe some "GalleryCard received undefined trek" warnings (which is fine - tells you which data is bad)

**🎯 Immediate Action Plan**

**Quick win (5 minutes):**

bash

*# 1. Edit GalleryCard.tsx - add null check at top*

*# 2. Edit PublicGallery.tsx - add .filter(trek => trek != null) before .map*

*# 3. Rebuild and deploy*

npm run build

*# Deploy*

**Proper fix (10 minutes):**

bash

*# Also fix the data fetching/initialization*

*# - Initialize state with []*

*# - Validate data in fetch function*

*# - Filter out bad records*

npm run build

*# Deploy*

**❓ Questions for You**

To help debug why you're getting undefined treks:

1. **Check your Supabase query** - is it returning null rows?
2. **Check if you have test data** with missing required fields
3. **Are you doing any data transformation** that might introduce undefined values?

Would you like me to:

1. ✅ **See your current GalleryCard.tsx code** to write the exact fix?
2. ✅ **See your PublicGallery.tsx fetchTreks function** to fix the data source?
3. ✅ **Help you write a data validation utility** to prevent this across all components?

**The good news:** This is a **data quality issue**, not an infinite loop in hooks! Much easier to fix. 🎉