Perfect! Now I have the complete picture. Let me create a comprehensive analysis document:

**🎯 COMPLETE FORENSIC ANALYSIS - Maximum Call Stack Issue**

**✅ GOOD NEWS: I Found the Root Causes**

After analyzing all your files, I've identified **3 critical issues** and **2 contributing factors**:

**🔴 CRITICAL ISSUE #1: Missing Safety Check in GalleryCard (PARTIALLY FIXED)**

**Current State (GalleryCard.tsx):**

tsx

**export** **const** GalleryCard: React.FC<GalleryCardProps> = ({ trek, onClick }) => {

*// Safety check for undefined trek*

**if** (!trek) {

**return** **null**;

}

*// ... rest of code*

**✅ Status:** **ALREADY FIXED!** You have the null check in place.

**However**, there's a problem with the prop destructuring on line 25:

tsx

*// Line 25 in original - This is where the error points*

**const** { trek, onClick } = props; *// If this line exists, it will fail*

**Verification Needed:** The error trace shows line 25 is the problem. Your current code shows the safety check AFTER destructuring. We need to ensure the component signature itself doesn't destructure before the check.

**🔴 CRITICAL ISSUE #2: Data Merging Can Produce Undefined Items**

**Location: PublicGallery.tsx Lines 410-423**

tsx

**const** merged = (treks ?? []).map((t) => ({

trek\_id: t.trek\_id,

name: t.name,

description: t.description ?? **null**,

location: t.location ?? **null**,

start\_datetime: t.start\_datetime,

difficulty: t.difficulty ?? **null**,

max\_participants: t.max\_participants ?? **null**,

images: imagesByTrek[t.trek\_id] ?? [],

video: videosByTrek[t.trek\_id] ?? **null**,

user\_contributions: userContributionsByTrek[t.trek\_id] ?? [],

tags: tagsByTrek[t.trek\_id] ?? [],

base\_price: t.base\_price ?? 0,

}));

**⚠️ PROBLEM:** If t itself is undefined or null in the treks array, this .map() will create objects with all undefined values, which then crash GalleryCard.

**Why This Happens:**

* Supabase query might return nulls in some edge cases
* The treks ?? [] doesn't filter out null items INSIDE the array
* treks could be [trek1, null, trek2, undefined, trek3]

**🔴 CRITICAL ISSUE #3: filteredItems Filter Isn't Comprehensive Enough**

**Location: PublicGallery.tsx Line 532**

tsx

**const** filteredItems = useMemo(() => {

**let** filtered = items.filter(Boolean); *// Remove undefined/null items first*

*// ...*

**✅ Good:** You have filter(Boolean) which removes falsy values.

**⚠️ PROBLEM:** This doesn't catch objects with undefined required fields:

tsx

{

trek\_id: **undefined**, *// ❌ This will crash*

name: **undefined**, *// ❌ This will crash*

start\_datetime: **undefined** *// ❌ This will crash*

}

The object itself is truthy, so filter(Boolean) passes it through, but it has undefined required fields.

**🟡 CONTRIBUTING FACTOR #1: useState Initial State**

**Location: PublicGallery.tsx Line 72**

tsx

**const** [items, setItems] = useState<PastTrek[]>([]);

**✅ GOOD:** You're initializing with an empty array [], not undefined. This is correct!

**🟡 CONTRIBUTING FACTOR #2: Rendering Section**

**Location: PublicGallery.tsx Lines 752-765**

tsx

{filteredItems.map((trek) => (

<GalleryCard

key={trek.trek\_id}

trek={{

trek\_id: trek.trek\_id,

name: trek.name,

description: trek.description,

location: trek.location,

start\_datetime: trek.start\_datetime,

images: trek.images,

}}

onClick={() => handleTrekClick(trek)}

/>

))}

**⚠️ PROBLEM:** If trek has undefined fields, you're creating an object with undefined properties and passing it to GalleryCard.

**The destructuring trek={{...}} creates:**

tsx

{

trek\_id: **undefined**, *// If trek.trek\_id is undefined*

name: **undefined**, *// If trek.name is undefined*

*// ... etc*

}

This object is NOT null/undefined itself, so GalleryCard's if (!trek) check passes, but then trying to use trek.name inside the component fails.

**🎯 THE COMPLETE FIX (3 Changes)**

**Fix #1: Strengthen filteredItems Filter**

**File: PublicGallery.tsx**  
**Location: Line 532**

tsx

*// BEFORE (current)*

**const** filteredItems = useMemo(() => {

**let** filtered = items.filter(Boolean); *// Remove undefined/null items first*

*// ...*

tsx

*// AFTER (strengthened)*

**const** filteredItems = useMemo(() => {

*// Remove null/undefined items AND items with missing required fields*

**let** filtered = items.filter(item =>

item &&

item.trek\_id != **null** &&

item.name &&

item.start\_datetime

);

*// ...*

**Why This Works:**

* Checks the item exists
* Checks trek\_id is not null/undefined
* Checks name is not empty/undefined
* Checks start\_datetime exists (required for date display)

**Fix #2: Add Validation to Merged Data**

**File: PublicGallery.tsx**  
**Location: After line 423, before setState**

tsx

*// CURRENT (lines 410-429)*

**const** merged = (treks ?? []).map((t) => ({

trek\_id: t.trek\_id,

name: t.name,

*// ... rest of mapping*

}));

**if** (append) {

setItems((prev) => [...prev, ...merged]);

} **else** {

setItems(merged);

}

tsx

*// FIXED - Add filtering after merge*

**const** merged = (treks ?? [])

.filter(t => t != **null**) *// ✅ Filter out null/undefined BEFORE mapping*

.map((t) => ({

trek\_id: t.trek\_id,

name: t.name,

description: t.description ?? **null**,

location: t.location ?? **null**,

start\_datetime: t.start\_datetime,

difficulty: t.difficulty ?? **null**,

max\_participants: t.max\_participants ?? **null**,

images: imagesByTrek[t.trek\_id] ?? [],

video: videosByTrek[t.trek\_id] ?? **null**,

user\_contributions: userContributionsByTrek[t.trek\_id] ?? [],

tags: tagsByTrek[t.trek\_id] ?? [],

base\_price: t.base\_price ?? 0,

}))

.filter(item => *// ✅ Filter out incomplete items AFTER mapping*

item.trek\_id != **null** &&

item.name &&

item.start\_datetime

);

**if** (append) {

setItems((prev) => [...prev, ...merged]);

} **else** {

setItems(merged);

}

**Fix #3: Add Defensive Rendering Check**

**File: PublicGallery.tsx**  
**Location: Line 752**

tsx

*// CURRENT*

{filteredItems.map((trek) => (

<GalleryCard

key={trek.trek\_id}

trek={{

trek\_id: trek.trek\_id,

name: trek.name,

*// ...*

}}

onClick={() => handleTrekClick(trek)}

/>

))}

tsx

*// FIXED - Add inline validation*

{filteredItems

.filter(trek => trek && trek.trek\_id && trek.name) *// ✅ Extra safety*

.map((trek) => (

<GalleryCard

key={trek.trek\_id}

trek={{

trek\_id: trek.trek\_id,

name: trek.name,

description: trek.description,

location: trek.location,

start\_datetime: trek.start\_datetime,

images: trek.images,

}}

onClick={() => handleTrekClick(trek)}

/>

))}

**📊 WHY THE STACK OVERFLOW HAPPENS**

Here's the exact sequence:

text

1. fetchTreks runs (line 252-437)

↓

2. Supabase returns: [trek1, null, trek2] (one null item)

↓

3. merged = treks.map(...) creates:

[

{trek\_id: 1, name: "Trek 1", ...},

{trek\_id: undefined, name: undefined, ...}, // ❌ Bad object from null

{trek\_id: 2, name: "Trek 2", ...}

]

↓

4. setItems(merged) stores bad data

↓

5. filteredItems = items.filter(Boolean)

- Passes all 3 items (objects are truthy even with undefined fields)

↓

6. Render: filteredItems.map(trek => <GalleryCard trek={{...}} />)

- Creates GalleryCard with trek={{trek\_id: undefined, name: undefined}}

↓

7. GalleryCard receives trek = {trek\_id: undefined, name: undefined}

- if (!trek) passes (object exists)

- But then: const startDate = new Date(trek.start\_datetime)

- trek.start\_datetime is undefined

↓

8. Error thrown: "Cannot read property..."

↓

9. React error boundary tries to recover

↓

10. Re-renders with SAME bad data still in state

↓

11. [LOOP: Steps 6-10 repeat infinitely]

↓

12. Maximum call stack exceeded

**🚀 IMPLEMENTATION PLAN**

**Step 1: Apply Fix #2 (Highest Impact)**

Edit PublicGallery.tsx around line 410-429:

tsx

**const** merged = (treks ?? [])

.filter(t => t != **null**) *// ADD THIS LINE*

.map((t) => ({

trek\_id: t.trek\_id,

name: t.name,

description: t.description ?? **null**,

location: t.location ?? **null**,

start\_datetime: t.start\_datetime,

difficulty: t.difficulty ?? **null**,

max\_participants: t.max\_participants ?? **null**,

images: imagesByTrek[t.trek\_id] ?? [],

video: videosByTrek[t.trek\_id] ?? **null**,

user\_contributions: userContributionsByTrek[t.trek\_id] ?? [],

tags: tagsByTrek[t.trek\_id] ?? [],

base\_price: t.base\_price ?? 0,

}))

.filter(item => *// ADD THESE 4 LINES*

item.trek\_id != **null** &&

item.name &&

item.start\_datetime

);

**Step 2: Apply Fix #1 (Defense in Depth)**

Edit PublicGallery.tsx line 532:

tsx

**const** filteredItems = useMemo(() => {

*// Remove null/undefined items AND items with missing required fields*

**let** filtered = items.filter(item =>

item &&

item.trek\_id != **null** &&

item.name &&

item.start\_datetime

);

**Step 3: Test Immediately**

bash

npm run build

*# Deploy*

*# Test /gallery page*

**Step 4: Optional Extra Safety (Fix #3)**

If still seeing issues, add inline filter at line 752.

**🔬 ROOT CAUSE SUMMARY**

| **Issue** | **Location** | **Impact** | **Fix Priority** |
| --- | --- | --- | --- |
| **Null items in Supabase response** | fetchTreks line 295 | Creates undefined objects | **HIGH** |
| **No validation after .map()** | Line 410-423 | Bad data enters state | **CRITICAL** |
| **Weak filter(Boolean)** | Line 532 | Doesn't catch objects with undefined fields | **HIGH** |
| **No render-time validation** | Line 752 | Last line of defense missing | **MEDIUM** |

**✅ EXPECTED OUTCOME**

After applying fixes:

* ✅ Null/undefined items filtered OUT before entering state
* ✅ Objects with missing required fields filtered OUT
* ✅ GalleryCard only receives valid complete data
* ✅ No more stack overflow errors
* ✅ Gallery page loads correctly

**🧪 VALIDATION**

After deploying, check:

javascript

*// In browser console on /gallery page*

console.log(window.\_\_ITW\_DEBUG\_\_.errors); *// Should be empty*

If you still see errors, run:

javascript

*// Check what's in state*

*// (You'll need React DevTools for this)*