**Enhanced Presentation of a Key Quality Issue**

**Issue: Array Index Used as Key in React Components**

**Location**:

* Message.jsx (lines 37, 45)
* SendMessage.jsx (lines 118, 124)

**Problem**:  
Using array indices as key props in React is an anti-pattern that can cause:

* Performance degradation during re-renders
* Incorrect component state when list order changes
* Difficulty for React to properly track component identity

**Impact**:

* Potential UI inconsistencies when messages are reordered
* Unnecessary re-renders impacting performance
* Possible state management bugs in dynamic lists

**Solution**:

1. Use unique identifiers from your data instead of indices:

jsx

Copy

// In Message.jsx

{message.messages.map((msg) => (

<div key={`${msg.role}-${msg.content.substring(0,10)}`}>

{/\* ... \*/}

</div>

))}

1. If no natural unique ID exists, generate stable keys:

jsx

Copy

// In SendMessage.jsx

{defaultModels.map((model) => (

<li className="model" key={model.split('/').pop()}>

{model}

</li>

))}

1. For the uploaded file content, use a hash of the content as key:

jsx

Copy

import { sha256 } from 'js-sha256';

// When processing the file

const contentHash = sha256(JSON.stringify(json));

**Benefits**:

* More efficient re-rendering
* Correct component state preservation
* Better performance with large lists
* More predictable behavior when data changes

**Implementation Plan**:

1. Audit all .map() operations in the codebase
2. Identify natural unique keys in each dataset
3. Implement key generation strategy where needed
4. Test with dynamic data changes to verify behavior

**Verification**:

* Verify no console warnings about key props
* Test with dynamic list reordering
* Profile performance before/after changes