The AI Help Assistant chat is rendering raw [[nav:revenue]], [[nav:open-ros]], [[nav:fees-saved]] etc. as plain bold text instead of converting them into tappable navigation links. This is broken and needs to be fixed.

THE PROBLEM

When the AI responds with text like `[[nav:revenue]]`, the chat is displaying the literal text "[[nav:revenue]]" in bold. It should be parsing that token and rendering a clickable button/link that navigates the user to that section of the app.

HOW IT SHOULD WORK

1. The Claude API system prompt tells Claude to use `[[nav:key]]` markup in responses
2. When the AI response comes back, the frontend message renderer needs to PARSE those tokens
3. Each `[[nav:key]]` token gets replaced with a styled, tappable button that:
    - Shows a label like "→ Today's Revenue" (with an arrow icon)
    - Is styled as a small blue pill/badge (blue background, rounded, inline with text)
    - When tapped: closes the chat panel, navigates to that section of the app, and highlights the target element

WHAT TO DO

Find the component that renders AI assistant chat messages. It's currently just dumping the raw text (or rendering markdown). You need to add a parser that:

Step 1: Create a navigation map

Create a map of nav keys to routes and labels. Example:

```typescript
```

```
const NAV_MAP = {
  'revenue': { label: "Today's Revenue", route: '/dashboard', hash: 'revenue', highlightId: 'stat-revenue' },
  'open-ros': { label: 'Open Work Orders', route: '/work-orders', highlightId: 'open-ros' },
  'total-customers': { label: 'Total Customers', route: '/customers', highlightId: 'total-customers' },
  'appointments': { label: "Today's Appointments", route: '/schedule', highlightId: 'appointments' },
  'fees-saved': { label: 'Fees Saved', route: '/dashboard', hash: 'fees-saved', highlightId: 'fees-saved-card' },
  'work-orders': { label: 'Work Orders', route: '/work-orders' },
  'estimates': { label: 'Estimates', route: '/estimates' },
  'customers': { label: 'Customers', route: '/customers' },
  'vehicles': { label: 'Vehicles', route: '/vehicles' },
  'inspections': { label: 'Inspections (DVI)', route: '/inspections' },
  'invoices': { label: 'Invoices', route: '/invoices' },
  'parts': { label: 'Parts', route: '/parts' },
  'schedule': { label: 'Schedule', route: '/schedule' },
  'reports': { label: 'Reports', route: '/reports' },
  'settings': { label: 'Settings', route: '/settings' },
};
```

Scan the entire codebase for every route and dashboard element and add ALL of them to this map. Don't just use the ones I listed — include everything navigable in the app.

Step 2: Build the message parser

Before rendering an AI message, run the text through a parser that replaces `[[nav:key]]` tokens with React components:

```typescript
```

```
function parseAIMessage(text: string, onNavigate: (key: string) => void): ReactNode[] {
  const parts = text.split(/(\[\[nav:[a-z0-9-]+\]\])/g);
  return parts.map((part, i) => {
    const match = part.match(/\[\[nav:([a-z0-9-]+)\]\]/);
    if (match) {
      const key = match[1];
      const target = NAV_MAP[key];
      if (target) {
        return (
          <button
            key={i}
            onClick={() => onNavigate(key)}
            className="inline-flex items-center gap-1 px-2 py-0.5 mx-0.5 text-blue-600 font-semibold text-sm bg-blue-50
          >
            <span className="text-xs opacity-70">→</span>
            {target.label}
          </button>
        );
      }
    }
    return <span key={i}>{part}</span>;
  });
}
```

Step 3: Handle the navigation action

When the user taps a nav link:

1. Close the AI chat panel
2. Wait 300ms for the close animation
3. Navigate to the target route using the router
4. If there's a highlightId, scroll to that element and add a temporary highlight effect (blue outline pulse for 2 seconds)
5. Optionally show a small toast notification confirming the navigation

Step 4: Use the parser in the chat message component

Find where AI messages are rendered (likely mapping over messages and rendering a bubble for each). Replace the raw text rendering with the parsed version:

```
tsx
```

```
// BEFORE (broken):
<div className="message-bubble">{message.content}</div>

// AFTER (fixed):
<div className="message-bubble">
  {message.role === 'assistant'
    ? parseAIMessage(message.content, handleNavigate)
    : message.content
  }
</div>
```

IMPORTANT

- Only parse messages with role "assistant" — user messages should render as plain text
- The nav link buttons must be INLINE with the surrounding text, not block elements
- Handle the case where a [[nav:key]] has no matching entry in NAV_MAP — just render the label text without a link
- Make sure this works on mobile — the tap targets should be at least 32px tall
- Test by asking the AI "What do my stats mean?" and verify every [[nav:...]] token renders as a blue tappable pill, not raw text