# Initial Prompt

**SuiteView - Windows Forms Dockable Toolbar Application**

**Project Overview:** Create a Windows Forms application named "SuiteView" - a lightweight, extensible dockable toolbar designed to host custom widgets and tools. The application should be production-ready with polished UI/UX and multi-monitor support.

**Technical Stack:**

* Framework: .NET 8.0 (Windows Forms)
* Target: Single executable file
* Maximum file size: 50MB
* Data persistence: JSON file (%AppData%\SuiteView\settings.json)

**UI/UX Specifications**

**Main Window - Toolbar:**

* **Default dimensions:** 80px (width) × 300px (height)
* **Default position:** Bottom-right corner of primary monitor
* **Resizable:** Yes, allow users to resize via drag handles
* **Window style:** Borderless custom window with rounded corners (optional)
* **Always-on-top:** Yes
* **Title bar components:**
  + "SuiteView" label (styled, top-left)
  + Close button (X) (top-right) - minimizes to tray, does not exit

**Color Scheme:** Design a modern, bold 2-3 tone color scheme with green as the primary color. Suggested default scheme:

* Primary: Vibrant green (#00D9A0 or similar)
* Secondary: Dark slate/charcoal (#2C3E50)
* Accent: White or light cyan for text/borders

**Settings Menu - 5 Preset Color Themes:** Create 5 professional preset themes (suggest names like "Emerald", "Matrix", "Forest", "Mint", etc.). Each should have 2-3 coordinated colors with green as the dominant color. Store theme selection in user preferences.

**Core Features**

**1. Docking System:**

* Drag toolbar to any screen edge (top, bottom, left, right)
* **Instant snap** when within 50 pixels of screen edge
* **Visual feedback:** Show semi-transparent preview/outline indicating where toolbar will dock during drag
* Persist dock position and orientation between sessions
* Auto-adjust orientation: horizontal layout when docked top/bottom, vertical when docked left/right

**2. Multi-Monitor Support:**

* Detect all connected monitors and handle monitor configuration changes
* Allow docking on any monitor
* Handle scenarios: laptop with 2 external monitors, laptop lid closed/open
* Gracefully reposition if saved monitor is disconnected

**3. System Tray Integration:**

* Custom application icon in system tray (always present)
* Right-click context menu:
  + "Show SuiteView" / "Hide SuiteView" (toggle)
  + "Settings"
  + "Exit" (true application exit)
* Double-click tray icon: Toggle toolbar visibility
* **Startup behavior:** Application starts minimized to tray (toolbar hidden by default)

**4. Settings Menu:** Implement a settings window/panel with:

* **Color Themes:** Dropdown or radio buttons to select from 5 preset themes
* **Launch on Windows Startup:** Checkbox to enable/disable auto-start with Windows
* **Opacity:** Slider to adjust toolbar transparency (70-100%)
* **Always on Top:** Toggle (default: enabled)
* Save/Cancel buttons

**Data Storage (JSON)**

Store the following in settings.json:

json

{

"windowPosition": { "x": 0, "y": 0 },

"windowSize": { "width": 80, "height": 300 },

"dockPosition": "BottomRight", *// Top, Bottom, Left, Right, BottomRight, etc.*

"monitorIndex": 0,

"selectedTheme": "Emerald",

"launchOnStartup": false,

"opacity": 100,

"alwaysOnTop": true,

"isVisible": false

}

**Architecture & Code Quality**

**Structure:**

* Separate classes for:
  + Main Form (toolbar)
  + Settings Form
  + JSON configuration manager
  + System tray manager
  + Docking logic handler
  + Theme manager

**Best Practices:**

* Clean, well-commented code with XML documentation
* Proper error handling and logging
* Implement IDisposable for resource cleanup
* Use async/await where appropriate
* Follow SOLID principles for extensibility

**Extensibility:**

* Design the toolbar content area to easily accommodate future widgets/buttons/tools
* Use a Panel or FlowLayoutPanel as the main content container
* Include placeholder comment: // Future widgets and tools will be added here

**Additional Requirements**

1. **Window Startup Registry:** When "Launch on Startup" is enabled, add entry to HKEY\_CURRENT\_USER\Software\Microsoft\Windows\CurrentVersion\Run
2. **Graceful Degradation:** Handle missing JSON file on first launch with sensible defaults
3. **Smooth UX:**
   * Fade in/out animations when showing/hiding (optional, 200ms)
   * Smooth drag experience with no flickering
   * Responsive UI (no freezing during operations)
4. **Icon:** Include a simple green-themed application icon (can be generated programmatically or use stock icon temporarily)

**Success Criteria**

* Application compiles and runs without errors on Windows 10/11
* All docking positions work correctly across multiple monitors
* Settings persist across application restarts
* System tray integration works flawlessly
* File size under 50MB
* Professional, polished appearance with cohesive color scheme
* Ready to extend with additional widgets and functionality

Please implement this application following C# and Windows Forms best practices. Provide clear comments explaining the docking logic, multi-monitor handling, and theme system for future maintenance.

The code will be written on a Linux subsytem on a Windows machine using WSL and ubuntu as the linux os. VS Code using Claude Code will be used to write the code. The SuiteView application will be run only on Windows machines

# Prompt for TAI Data tool

I want a form that will have a tree view at the left side. The first Entry with say LifeProd\_Library. When I click on it, the code will query the database to find all the tables in the target SQL database (UL\_Rates) using the ODBC set up on the local computer. All these tables will be listed in the right side fo the along with the dates they they were last scanned (when this is first run none of the tables will be scanned). Then I want to be able to select one or more tables and click “Scan”. When this clicked the code will scan the table specifications for each table to find all the field names and datatypes and store that information in our json data structure along with the date of the scan. I will then be able to click a “Move to View” button. When this is clicked, all the selected tables will but listed in the tree view under My\_Library. And you can expand each table in the tree and see all the fields for that table and the datatypes for those fields. I want this tool to be able to run indepent of SuiteView, but I want a button in suiteview that will open this tool. And I want the same look and feel and color theme for this tool as for suiteview. And I will continue to add features to this tool, this prompt is just to get the initial setup started. This is a big new feature so spend some time thinking about how the code base will need to adapted considering this new feature.