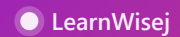


AI Coding Exercise

Build this module's VisualOperationsStudio sample with ChatGPT or Claude from a ready-made prompt that points the model at this module's specification, then review, run and extend what it gives you.



Brief it, read it, then break it.

This exercise has three parts, and only the first is about prompting. You brief an assistant with the module specification and get a VisualOperationsStudio solution back; you read what it wrote and decide whether it really does what the module asked; then you push it further than the module asked. The second and third parts are where the learning is.

What you are asking for

Give VisualOperationsStudio an operations grid with two user-painted columns. Add explicit columns to gridOperations, set UserPaint = true on the Health and Trend columns only, and handle CellPaint once: return immediately when e.RowIndex is negative or e.ColumnIndex is not one of the two targets, then draw a threshold-coloured progress bar for Health and a polyline sparkline for Trend inside e.ClipRectangle. Read each row's values from the bound MachineStatus object rather than searching a list inside the handler, respect the selected row's colours, and keep the handler free of side effects. Bind 1,000 generated rows, scroll, and then build the same status indicator a second time as an AllowHtml cell so you can state in writing which one you would ship.

The module asks for these deliverables, and the prompt below asks the assistant for exactly them:

- ✓ gridOperations with explicit columns and UserPaint = true on only the Health and Trend columns
- ✓ One CellPaint handler that guards e.RowIndex and e.ColumnIndex before drawing anything
- ✓ Threshold-coloured progress cell and a polyline sparkline cell drawn inside e.ClipRectangle
- ✓ Row values read from the bound MachineStatus object with no per-cell lookup, allocation or control creation
- ✓ An AllowHtml status column built alongside, plus a note comparing both on a 1,000-row scroll

Why it is shaped this way

Module 2 gave VisualOperationsStudio a TelemetryGauge: one control, one machine, a few hundred repaints an hour. The operations grid is a different animal. A shift supervisor opens a list of a thousand machines and wants to see at a glance which ones are near their limit and which have been drifting all morning. Numbers in a column will not do that, and one gauge per row would be a thousand server-side controls in a single session. The grid itself has to draw, once per visible cell, and stop the moment the cell scrolls away.

These are the terms the assistant has to get right. If what comes back uses them loosely, it has not understood the module:

Term	What it means here
UserPaint	The explicit switch: DataGridViewCell.UserPaint = true for one cell, DataGridViewColumn.UserPaint = true for a whole column. Without it the grid never asks you to paint.
CellPaint	The grid-level event raised for user-painted cells when they need to be drawn; you subscribe once on the grid, not once per row, and route inside the handler.
DataGridViewCellPaintEventArgs	Derives from PaintEventArgs, so it carries Graphics and ClipRectangle, and adds RowIndex and ColumnIndex so one handler can serve several columns.
no WinForms helpers	The documented Wisej.NET event arguments do not expose the WinForms convenience methods PaintBackground() and PaintContent(). Your handler owns the whole visual result of that cell.
e.ClipRectangle as the surface	The cell's local destination rectangle. Columns can be resized and the grid can be laid out responsively, so every coordinate you draw has to be derived from it rather than from a constant.
pure render helpers	Methods shaped like DrawProgressCell(Graphics g, Rectangle rect, double value), so the handler becomes routing code and the same helper can later render into an off-screen bitmap.
AllowHtml cell content	The browser-native alternative: icons, spans, labels and CSS rendered by the client, and the reason the documentation reserves CellPaint for visuals too complex for HTML.

Build it — pick your assistant

Both buttons open a new tab with the **same ready-made prompt** already filled in. The prompt points the model at this module's specification — a single Markdown file holding the lesson, the lab brief and the deliverables — so it works from the real material instead of guessing what Module 3 is about. You can open [module-03.md](#) yourself to see exactly what it will read.



Use ChatGPT

Opens [chatgpt.com](#) with this module's prompt loaded. Works with GPT-5 or Codex; any plan that can browse the web will fetch the spec.



Use Claude

Opens [claude.ai](#) with the same prompt loaded. Claude Code users can paste it into a terminal in an empty folder instead.



Send it as-is and you get a project, not a snippet

The prompt ends by asking for the whole solution as a single `.zip` with build-and-run steps. Download it, open the solution in Visual Studio, press F5 and VisualOperationsStudio comes up on `localhost`. You need the .NET SDK and the `Wisej-4` NuGet package; everything else the prompt asks the assistant to write itself.

The prompt, in full — read it, then edit it if you want a different sample

I am working through Module 3 – "DataGridView Cell Painting" – of the LearnWisej Drawing & Painting in Wisej.NET course, and I want to build the module sample with you.

Start by reading the module specification. It is a single Markdown file holding the full lesson, the lab brief and the deliverables:

<https://learnwisej.com/assets/courses/drawing-and-painting/specs/module-03.md>

Then build a complete Wisej.NET 4 project I can open in Visual Studio:

1. Project "VisualOperationsStudio", C#, net8.0 or later, NuGet package Wisej-4 4.1.x, Kestrel host calling `app.UseWisej()`.
2. Build what the specification's lab asks for: Give VisualOperationsStudio an operations grid with two user-painted columns. Add explicit columns to `gridOperations`, set `UserPaint = true` on the Health and Trend columns only, and handle `CellPaint` once: return immediately when `e.RowIndex` is negative or `e.ColumnIndex` is not one of the two targets, then draw a threshold-coloured progress bar for Health and a polyline sparkline for Trend inside `e.ClipRectangle`. Read each row's values from the bound `MachineStatus` object rather than searching a list inside the handler, respect the selected row's colours, and keep the handler free of side effects. Bind 1,000 generated rows, scroll, and then build the same status indicator a second time as an `AllowHtml` cell so you can state in writing which one you would ship.
3. Produce these deliverables, and tell me where each one lives in the solution:
 - `gridOperations` with explicit columns and `UserPaint = true` on only the Health and Trend columns
 - One `CellPaint` handler that guards `e.RowIndex` and `e.ColumnIndex` before drawing anything
 - Threshold-coloured progress cell and a polyline sparkline cell drawn inside `e.ClipRectangle`
 - Row values read from the bound `MachineStatus` object with no per-cell lookup, allocation or control creation
 - An `AllowHtml` status column built alongside, plus a note comparing both on a 1,000-row scroll
4. Follow the specification wherever it names a Wisej.NET class, property or event – use exactly that one. Where it leaves a choice open, tell me which API you picked and why.
5. Keep the server authoritative: state and validation live in .NET, and the browser renders and reports. Never move a decision into JavaScript because it is easier there.
6. Make the result demonstrable: I should be able to run it and see the behaviour this module is about, including what happens when something goes wrong.

Before you write any code, restate in your own words what you are going to build and where each piece will live, so I can correct the design while it is still cheap.

When the code is done, package the whole solution as a single .zip I can download, and give me the exact steps to build and run it.

Finally, answer these questions from the specification, using your own implementation as the evidence:

- A colleague pastes a WinForms cell-painting sample that starts by calling `e.PaintBackground(...)`. Explain what the Wisej.NET event arguments give you instead, and what the handler now has to decide for itself.
- Your sparkline helper hard-codes a 100-pixel plot width and looks fine on your screen. Describe two user actions that break it, and what the helper should read instead.
- The HTML status column and the painted Health column both show a critical machine. Argue

which one you would ship for a grid that support staff search and read with a screen reader, and what you would lose either way.

Ask me about anything that is ambiguous before you start.

Then review what came back

Code that compiles is not yet code that is right. The interesting questions are design questions: is this the Wisej.NET control the job actually calls for, does it follow the lifecycle the framework gives you, is the authoritative state on the server rather than parked in the browser because that was convenient? Read the result as a reviewer, not as a customer.

This module gives you the checklist. Answer each one against the code you just received:

- ✓ A colleague pastes a WinForms cell-painting sample that starts by calling `e.PaintBackground(...)`. Explain what the Wisej.NET event arguments give you instead, and what the handler now has to decide for itself.
- ✓ Your sparkline helper hard-codes a 100-pixel plot width and looks fine on your screen. Describe two user actions that break it, and what the helper should read instead.
- ✓ The HTML status column and the painted Health column both show a critical machine. Argue which one you would ship for a grid that support staff search and read with a screen reader, and what you would lose either way.

Anything you had to correct is worth writing down. A short note of the conventions and decisions you settled on, pasted into the first message of your next session, is worth more than any single clever prompt.



No reference sample for this course yet — yours is the first

The [LearnWisej samples repository](#) carries worked builds for most courses, but not yet for this one. That means there is nothing to compare against and no shortcut: the specification and your own review are the whole safety net. Keep what you build — and if it turns out well, we would like to see it.

Three prompts to take it further

Point your assistant at the project you just built and try these. Each one is a review-and-extend exercise, not a rewrite.

- ✓ **Review it like a pull request.** *"Walk me through this VisualOperationsStudio code as if you were reviewing a colleague's pull request and did not write it. Where does state live, what happens end to end on one user action, what would you send back for changes, and which parts would you not be comfortable putting in front of a customer? Be specific and quote the lines you mean — do not summarise the design back to me."*
- ✓ **Swap a piece without moving the seams.** *"Pick the part of this VisualOperationsStudio project that is most obviously a stand-in for something real — a fake data source, an in-memory store, a simplified service — and replace it with a production-shaped implementation. The contract either side of it must not change: same types, same calls, same behaviour under failure. Tell me what the real implementation forced you to handle that the stand-in did not."*

- ✓ **Make it survive a bad day.** *"Review this `VisualOperationsStudio` project for the failure modes it does not handle yet — bad or missing input, a dependency that is slow or unavailable, work arriving after the user navigated away, and two actions racing each other. For each one, tell me first whether it is a real risk in `Wisej.NET 4` and why, and only then write the fix. Where you decide something needs no handling, say so and explain the reasoning rather than adding defensive code."*



Carry it into the lab

The hands-on lab that closes this module asks for the same deliverables you just had the assistant produce. Use what you have here as the shape of your answer and the evidence behind it — not as the answer itself. The lab is graded on your reasoning, and that part has to be yours.