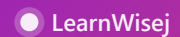


AI Coding Exercise

Build this module's AdaptiveOps 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 — including against our own reference build on GitHub.



Same module, built the way you will actually build things.

Every other section of this module explains how "FlowLayoutPanel, TableLayoutPanel, FlexLayoutPanel, and Extended Layout Properties" works. This one asks you to produce it — by directing an assistant with the module's own specification, then verifying the result runs and does what the lab brief describes. It is the workflow you will use on real work, practised on something whose right answer you already know.

What you are asking for

Build three versions of the Adaptive Operations Console content area: a FlowLayoutPanel filter and metric-card area that wraps and uses FillWeight and FlowBreak, a TableLayoutPanel ticket editor with aligned labels and editors driven by RowStyles and ColumnStyles, and a FlexLayoutPanel dashboard shell where the list and details regions share space by FillWeight with AlignX, AlignY, MinimumSize and MaximumSize set. Build one of the three in code with the Wisej.Web.Markup fluent extensions and compare how each survives a browser resize.

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

- ✓ FlowLayoutPanel filter and metric-card area using WrapContents, FillWeight and FlowBreak
- ✓ TableLayoutPanel ticket editor with RowStyles, ColumnStyles and spans for aligned fields
- ✓ FlexLayoutPanel dashboard shell with FillWeight, AlignX, AlignY, MinimumSize and MaximumSize
- ✓ One region built in code with the Wisej.Web.Markup fluent extensions
- ✓ Resize comparison screenshots and a note on which container fits each region

Why it is shaped this way

Module 4 gave the Adaptive Operations Console a shell of docked regions, but docking only decides where a region sits, not how the content inside it behaves when the browser is 1400 pixels wide on Monday and 700 on a tablet on Tuesday. A row of metric cards should wrap, a ticket editor should keep its labels and editors aligned however long the German translation is, and the list and details regions should share the workspace in a proportion you decide. Each is a different content behaviour, and Wisej.NET ships a container for each: FlowLayoutPanel, TableLayoutPanel and FlexLayoutPanel. Picking the wrong one is how teams end up writing resize code again.

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
FlowLayoutPanel	Arranges children in automatic rows or columns along FlowDirection, wraps when WrapContents is true and ignores each child's Location, Dock and Anchor, so chips and cards reflow on their own.
TableLayoutPanel	Places children in cells addressed by Row and Column, with spans for cells that stretch, so labels and editors stay aligned while each child can still dock or anchor inside its cell.
FlexLayoutPanel	Lays children out in one row or one column, always fills its client area and shares the remaining space by weight, which suits major regions that split the workspace proportionally.
FillWeight	An extended child property read by the flow and flex engines: a weight above zero stretches the child into the remaining space in proportion to the other weighted children; zero keeps its own size.
FlowBreak	An extended FlowLayoutPanel child property that forces the next child onto a new row or column, so a filter bar ends a line deliberately instead of relying on the width.
Row and column styles	RowStyles and ColumnStyles declare whether a row or column auto-sizes, takes a fixed pixel size or shares the rest by percentage; this is where a form's proportions live.
Margin and Padding	Padding is the container's inner space around all children; Margin is the child's distance to its neighbours, and only engines that honour margins will use it.

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 5 is about. You can open [module-05.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 AdaptiveOps 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 5 – "FlowLayoutPanel, TableLayoutPanel, FlexLayoutPanel, and Extended Layout Properties" – of the LearnWisej Theming & Responsive Layouts 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/theming-and-responsive-layouts/specs/module-05.md>

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

1. Project "AdaptiveOps", 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: Build three versions of the Adaptive Operations Console content area: a FlowLayoutPanel filter and metric-card area that wraps and uses FillWeight and FlowBreak, a TableLayoutPanel ticket editor with aligned labels and editors driven by RowStyles and ColumnStyles, and a FlexLayoutPanel dashboard shell where the list and details regions share space by FillWeight with AlignX, AlignY, MinimumSize and MaximumSize set. Build one of the three in code with the Wisej.Web.Markup fluent extensions and compare how each survives a browser resize.
3. Produce these deliverables, and tell me where each one lives in the solution:
 - FlowLayoutPanel filter and metric-card area using WrapContents, FillWeight and FlowBreak
 - TableLayoutPanel ticket editor with RowStyles, ColumnStyles and spans for aligned fields
 - FlexLayoutPanel dashboard shell with FillWeight, AlignX, AlignY, MinimumSize and MaximumSize
 - One region built in code with the Wisej.Web.Markup fluent extensions
 - Resize comparison screenshots and a note on which container fits each region
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:

- You gave the details region FillWeight 1 but no MinimumSize, and at 700 pixels it vanished. Explain what the flex engine did with the space and why the fix is a size constraint rather than a different weight.
- The Apply button wraps to the second row at 1024 pixels even though there is room for it. Which extended property did that, and how would the FilterBar behave differently if you removed it and made WrapContents false?
- A Margin of 8 on the cards in the FlowLayoutPanel produced gaps, but the same Margin on the docked navigation rail did nothing. Which engine honours margins in each case, and which property creates the gap for the rail?

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:

- ✓ You gave the details region FillWeight 1 but no MinimumSize, and at 700 pixels it vanished. Explain what the flex engine did with the space and why the fix is a size constraint rather than a different weight.
- ✓ The Apply button wraps to the second row at 1024 pixels even though there is room for it. Which extended property did that, and how would the FilterBar behave differently if you removed it and made WrapContents false?
- ✓ A Margin of 8 on the cards in the FlowLayoutPanel produced gaps, but the same Margin on the docked navigation rail did nothing. Which engine honours margins in each case, and which property creates the gap for the rail?

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.



We already built this one — compare yours with ours

The Module 5 reference implementation lives in the public samples repository. Build yours first, then open ours: the differences are the interesting part.



[Open the Module 5 sample on GitHub](#)

Or clone the whole set with `git clone https://github.com/iceteagroup/LearnWisej-Samples` and open `Theming and Responsive Layouts Course/Module 5`.



Our sample is a starting point, not a finished product

It was built the same way you are building yours — by prompt, then corrected — and it deliberately stops where the module stops. Expect stand-ins where production code would go, thin error handling and no tests. Treat that as an invitation: use what you learned in this module, plus the assistant, to push it further. Improving code you did not write is the skill this course is actually teaching.

Three prompts to take it further

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

- ✓ **Make it survive a bad day.** *"Review this AdaptiveOps 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."*
- ✓ **Add the second case.** *"Extend this AdaptiveOps project with a second case that follows the same pattern as the first — another entity, screen or control of the same shape, wired the same way. Factor out what is genuinely shared into a base class or service, but only where it earns its place; two similar things do not always want an abstraction. Show me the diff and justify each thing you extracted."*
- ✓ **Prove it actually works.** *"Go through this project deliverable by deliverable against the module specification at <https://learnwisej.com/assets/courses/theming-and-responsive-layouts/specs/module-05.md> and show me the evidence that each one is genuinely done — the code path, what to click, and what I should see. Where a deliverable is only partly met, say so plainly instead of arguing it through. Then add whatever is missing."*



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.