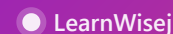


AI Coding Exercise

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.



You are not going to type this sample. You are going to direct it.

Almost nobody writes Wisej.NET code from a blank file any more — an assistant writes the first draft and the developer decides whether it is right. That only works if you know the material, which is what the rest of this module gave you. Here you hand that knowledge to ChatGPT or Claude, get this module's sample back as a working Visual Studio solution, and review it the way you would review a colleague's pull request.

What you are asking for

Add a ClientProfiles.json to the Adaptive Operations Console with Phone, Phone Landscape, Tablet, Tablet Landscape, Small Desktop and Desktop profiles ordered narrow to broad. In the designer, use responsive properties to hide the navigation rail and details panel on phone, switch the toolbar buttons to icon-only Display, and re-dock the details region under the grid on tablet. Handle ResponsiveProfileChanged to show the active profile name in the status label, log each change and open the details editor as a modal form on phone instead of a side panel.

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

- ✓ ClientProfiles.json with Phone, Phone Landscape, Tablet, Tablet Landscape, Small Desktop and Desktop profiles in first-match order
- ✓ Responsive property values assigned in the designer to hide the navigation rail and details panel on phone
- ✓ Icon-only toolbar Display and re-docked details region for tablet set through responsive properties
- ✓ ResponsiveProfileChanged handler that shows and logs the active profile name
- ✓ Details editor opened as a modal form on phone through the profile handler

Why it is shaped this way

The Adaptive Operations Console now has a themed shell built from Flow, Table and Flex containers that survives a resize without a Resize handler, but it shows the same screen to everyone: on a 390-pixel phone the navigation rail, the ticket grid and the details editor compete for the same width. CSS media queries cannot fix this, because the controls live on the server: hiding a panel, switching a button to icon-only or re-docking a region is a server-side property change, and a phone user who loses the details panel needs a

different way to edit a ticket, not a smaller copy of the desktop one. This module adds the mechanism Wisej.NET owns for that: client profiles, responsive properties and the ResponsiveProfileChanged event. 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
ClientProfile	A named set of device-related matching rules (browser width, screen width, device type, user agent, landscape) that describes how a class of client is used, not a marketing device name.
ClientProfiles.json	The file Wisej.NET reads; profiles are evaluated from top to bottom and the first one that matches wins, so the order of the entries is part of the configuration.
Application.ActiveProfile	The profile currently selected for this session; its Name is what the status label shows and what the profile handler branches on.
responsive property	A normal.NET property marked with ResponsivePropertyAttribute that holds one value per client profile and applies the matching value whenever the active profile changes.
designer-first adaptation	Responsive values are assigned only in the Visual Studio designer, which keeps them serialized, visible and deterministic; code is for adaptations that depend on data or runtime state.
ResponsiveProfileChanged	The event raised on Application, on containers such as Form, Page, Desktop and UserControl and on DataGridView and ListView when the active profile changes; ResponsiveProfileChangedEventArgs.CurrentProfile carries the new profile.
responsive QA matrix	The test plan listing width, height, orientation, device type, touch, scrollbars, font scaling and the profile name you expect at each size.

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 6 is about. You can open [module-06.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 6 – "ClientProfile and Responsive 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-06.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: Add a `ClientProfiles.json` to the Adaptive Operations Console with Phone, Phone Landscape, Tablet, Tablet Landscape, Small Desktop and Desktop profiles ordered narrow to broad. In the designer, use responsive properties to hide the navigation rail and details panel on phone, switch the toolbar buttons to icon-only Display, and re-dock the details region under the grid on tablet. Handle `ResponsiveProfileChanged` to show the active profile name in the status label, log each change and open the details editor as a modal form on phone instead of a side panel.
3. Produce these deliverables, and tell me where each one lives in the solution:
 - `ClientProfiles.json` with Phone, Phone Landscape, Tablet, Tablet Landscape, Small Desktop and Desktop profiles in first-match order
 - Responsive property values assigned in the designer to hide the navigation rail and details panel on phone
 - Icon-only toolbar Display and re-docked details region for tablet set through responsive properties
 - `ResponsiveProfileChanged` handler that shows and logs the active profile name
 - Details editor opened as a modal form on phone through the profile handler
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:

- If Small Desktop were moved to the top of `ClientProfiles.json`, which of the other five profiles could still be matched, and what would a tablet in landscape see?
- The tablet re-dock lives in the designer and the phone modal editor lives in code. What decides which mechanism owns each change, and what would go wrong if you swapped them?
- A tester rotated a phone four times and your handler ran four times. Which state must be identical after the fourth run and after the first, and how does your code guarantee it?

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:

- ✓ If Small Desktop were moved to the top of ClientProfiles.json, which of the other five profiles could still be matched, and what would a tablet in landscape see?
- ✓ The tablet re-dock lives in the designer and the phone modal editor lives in code. What decides which mechanism owns each change, and what would go wrong if you swapped them?
- ✓ A tester rotated a phone four times and your handler ran four times. Which state must be identical after the fourth run and after the first, and how does your code guarantee it?

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



[Open the Module 6 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 6`.



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.

- ✓ **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-06.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."*
- ✓ **Review it like a pull request.** *"Walk me through this AdaptiveOps 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."*



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.