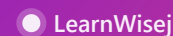


AI Coding Exercise

Build this module's IconDesk 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.



The assistant is the fast pair of hands. You are still the architect.

An assistant can produce this module's sample in minutes. What it cannot do is decide whether the result is the right design for your application — that call stays with you. You supply the judgement: brief it properly, then check what comes back against what Module 7 just taught you.

What you are asking for

Finish the IconDesk capstone. Reference an icon-font stylesheet from Default.html, set AllowHtml on a small toolbar or card and render its icons as font glyphs, then build a DataGridView column whose cells hold icon-font actions and read which element the user clicked so Edit and Delete do different things. Build a summary page that demonstrates every mechanism the course covered at once: a theme image, an official pack icon, a custom pack icon, an embedded logo and a recoloured SVG. Then write the project's production icon policy and run a QA pass in a light and a dark theme.

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

- ✓ Icon-font stylesheet referenced from Default.html and one AllowHtml toolbar or card rendering font glyphs
- ✓ DataGridView column whose cells render icon-font actions and respond to the clicked element
- ✓ IconDesk summary page demonstrating a theme image, an official pack icon, a custom pack icon, an embedded asset and a recoloured SVG
- ✓ Written production icon policy covering accessibility, licensing, caching and external dependencies
- ✓ QA pass run in a light and a dark theme with the findings recorded

Why it is shaped this way

After six modules IconDesk can draw an icon five different ways, and every one of them works. What the project still lacks is a decision rule: when a new screen needs a Save symbol, nobody should have to guess whether it comes from the theme, from an official Wisej-4 pack or from IconDesk.Icons. This module adds the last option, icon fonts rendered as HTML, and then writes the policy that says which one to reach for and what to check before a screen ships.

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
icon font	A web font whose glyphs are pictures instead of letters, chosen with CSS class names and drawn by the browser inside HTML content rather than through any image property.
AllowHtml	The per-control switch that tells Wisej.NET to interpret a control's text as markup; without it the tags appear as literal characters, because text is plain text by default.
Default.html head	The application's host page, where one stylesheet link makes the icon font's CSS and font files available to every screen the session opens.
role-based element click	A click inside HTML content can report which inline element was hit, so one grid cell can offer an Edit target and a Delete target without two separate controls.
CDN versus self-hosting	Pointing the link at a public CDN is a production decision about availability, privacy, Content Security Policy and licensing; internal applications often host the files themselves.
layered icon policy	A written rule giving every kind of symbol one home: theme images for semantics, one primary pack for commands, the custom pack for product artwork, raster files for photos.
icon QA pass	A walk through the finished screens in a light and a dark theme at more than one display scale, checking that each icon is still visible and still has a text alternative.

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 7 is about. You can open [module-07.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 IconDesk 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 7 – "Icon Fonts, HTML Content and Production Icon Strategy" – of the LearnWisej Icons and Images 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/icons-and-images/specs/module-07.md>

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

1. Project "IconDesk", 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: Finish the IconDesk capstone. Reference an icon-font stylesheet from `Default.html`, set `AllowHtml` on a small toolbar or card and render its icons as font glyphs, then build a `DataGridView` column whose cells hold icon-font actions and read which element the user clicked so Edit and Delete do different things. Build a summary page that demonstrates every mechanism the course covered at once: a theme image, an official pack icon, a custom pack icon, an embedded logo and a recoloured SVG. Then write the project's production icon policy and run a QA pass in a light and a dark theme.
3. Produce these deliverables, and tell me where each one lives in the solution:
 - Icon-font stylesheet referenced from `Default.html` and one `AllowHtml` toolbar or card rendering font glyphs
 - `DataGridView` column whose cells render icon-font actions and respond to the clicked element
 - IconDesk summary page demonstrating a theme image, an official pack icon, a custom pack icon, an embedded asset and a recoloured SVG
 - Written production icon policy covering accessibility, licensing, caching and external dependencies
 - QA pass run in a light and a dark theme with the findings recorded
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:

- The stylesheet link is in `Default.html` and the class names are right, but one Label shows the raw tags while the Button next to it shows the glyph. What differs between the two controls, and why is the setting per control rather than per application?
- Your action column works, and a colleague asks to put the customer's own note text into the same cell so it can be styled. What do you tell them, and what would you build instead?
- On the summary page, which of the five images change when the application switches theme and which stay exactly as drawn? What does that tell you about where each one is stored?

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:

- ✓ The stylesheet link is in Default.html and the class names are right, but one Label shows the raw tags while the Button next to it shows the glyph. What differs between the two controls, and why is the setting per control rather than per application?
- ✓ Your action column works, and a colleague asks to put the customer's own note text into the same cell so it can be styled. What do you tell them, and what would you build instead?
- ✓ On the summary page, which of the five images change when the application switches theme and which stay exactly as drawn? What does that tell you about where each one is stored?

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 IconDesk 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 IconDesk 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 IconDesk 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.