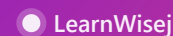


## AI Coding Exercise

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.



### 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 an IconDesk 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

Fill an IconGallery page in IconDesk using the designer Image Selector instead of typing source strings by hand. Assign images to five controls: one named theme image, one project SVG referenced by relative URL, one absolute URL and at least two icon-pack assets if a pack is installed. Recolor one monochrome SVG with the ?color= suffix and a theme color name, open the generated Designer.cs and read the image-source strings the picker produced, then switch the application between two Wisej themes and record which assets changed and which stayed exactly the same.

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

- ✓ IconGallery page with five controls whose images were assigned through the designer Image Selector
- ✓ One named theme image, one relative project SVG, one absolute URL and at least two icon-pack assets among them
- ✓ Monochrome SVG recoloured through the?color= suffix with a theme colour name
- ✓ Designer.cs excerpt showing the image-source strings the picker generated
- ✓ Theme-switch note recording which assets followed the theme and which stayed fixed

### Why it is shaped this way

IconDesk now has a command panel and an image lab, and every picture in them is a shipped file plus a path somebody typed. That holds until the application is re-themed, deployed under a different virtual directory, or asked to show the same symbol in a warning colour on one screen and a neutral one on another. This module moves the origin of an image out of the project folder and into the theme, a URL or the designer, so the string a control carries still means something after the artwork, the theme or the server changes.

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                                                                                                                                                                                     |
|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| named theme image     | A semantic name such as icon-print that the active theme maps to artwork, so a different theme can supply a different picture with no C# change.                                                       |
| client theme cache    | Theme images travel with the theme and are already in the browser when a control asks for one, which makes a theme the right home for a small set of command symbols and the wrong home for thousands. |
| relative URL source   | An image-source string resolved against the application: the simplest form for a logo deployed with the app.                                                                                           |
| absolute URL source   | A complete external address that turns the image into a dependency on another system, with availability, privacy, CORS and Content Security Policy consequences.                                       |
| embedded base64 image | Image bytes written into the theme definition itself, so the artwork needs no second request, at the cost of a larger theme.                                                                           |
| Image Selector        | The Visual Studio design-time picker that browses project images, theme images and referenced icon-pack resources, then serializes the chosen source string into Designer.cs.                          |
| ?color= suffix        | A suffix on an image source that asks Wisej.NET to recolour a compatible monochrome SVG with a CSS colour or a theme colour name.                                                                      |

## 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 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 3 – "Theme Images, URLs, SVG Color and the Designer Image Picker" – 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-03.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: Fill an `IconGallery` page in `IconDesk` using the designer Image Selector instead of typing source strings by hand. Assign images to five controls: one named theme image, one project SVG referenced by relative URL, one absolute URL and at least two icon-pack assets if a pack is installed. Recolor one monochrome SVG with the `?color=` suffix and a theme color name, open the generated `Designer.cs` and read the image-source strings the picker produced, then switch the application between two Wisej themes and record which assets changed and which stayed exactly the same.
3. Produce these deliverables, and tell me where each one lives in the solution:
  - `IconGallery` page with five controls whose images were assigned through the designer Image Selector
  - One named theme image, one relative project SVG, one absolute URL and at least two icon-pack assets among them
  - Monochrome SVG recoloured through the `?color=` suffix with a theme colour name
  - `Designer.cs` excerpt showing the image-source strings the picker generated
  - Theme-switch note recording which assets followed the theme and which stayed fixed
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 gallery renders correctly on your machine and shows nothing for the absolute URL on a colleague's. Which of the five mechanisms could produce that difference, and what would you check first?
- Two of your icons refuse to change colour when the suffix is added. What is inside those files, and when is editing them the wrong answer?
- A designer asks for the same symbol in the gallery, a toolbar and a grid cell. Does the theme key, the relative URL or the pack resource cost the least to change later, and why?

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 gallery renders correctly on your machine and shows nothing for the absolute URL on a colleague's. Which of the five mechanisms could produce that difference, and what would you check first?
- ✓ Two of your icons refuse to change colour when the suffix is added. What is inside those files, and when is editing them the wrong answer?
- ✓ A designer asks for the same symbol in the gallery, a toolbar and a grid cell. Does the theme key, the relative URL or the pack resource cost the least to change later, and why?

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.

- ✓ **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."*
- ✓ **Add the second case.** *"Extend this IconDesk 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."*



### **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.