

AI Coding Exercise

Build this module's ValidationClinic 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.

 LearnWisej



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

Create the ValidationClinic customer intake form with txtName and txtEmail TextBoxes, a Save button and a Cancel button. Attach Validating handlers that set e.Cancel = true and an ErrorProvider message when the name is blank or the email has no @, clear the message when the value is valid, and trim the name in its Validated handler. Set the form's AutoValidate mode, make sure Cancel does not cause validation so the user can always leave, and call ValidateChildren() at the top of btnSave_Click so Save stops when any field is invalid. Finish with a short lab note that explains the focus behavior you chose.

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

- ✓ Customer intake form with txtName and txtEmail, a Save button and a Cancel button
- ✓ Validating handlers that set e.Cancel = true and an error message for a blank name and an invalid email
- ✓ Validated handler that trims the accepted name
- ✓ Save handler that calls ValidateChildren() and stops when it returns false
- ✓ Lab note naming the AutoValidate mode chosen and why Cancel still works on an invalid form

Why it is shaped this way

The ValidationClinic customer intake form currently accepts anything: a blank name, an email with no @, or a Save click on a form nobody filled in. The first idea most beginners have is an if statement inside the Save button, but that gives the user no feedback while they work and turns Save into a wall of errors at the very end. This module puts validation where it belongs in a Wisej.NET form: on each editor when the user leaves it, and once more when the user clicks Save.

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
validation	The decision whether a value is acceptable before the application uses it, followed by visible feedback and, when the value is wrong, a refusal to continue.
Validating	The event a control raises when the user tries to leave it (or when code asks for validation); its handler reads the value and is the one place where accept or reject is decided.
CancelEventArgs.Cancel	The flag a Validating handler sets to true to reject the value: the control counts as invalid, focus can be kept in it, and ValidateChildren() reports failure.
Validated	The event raised only after Validating finished without cancelling; the place for tidy-up such as trimming text, never for saving.
AutoValidate	A property of a form, page or user control that decides what happens to focus when a child control fails validation.
CausesValidation	A per-control switch; set to false on a button such as Cancel, clicking it does not validate the field the user is leaving.
ValidateChildren()	A container method that runs validation for its child controls, including fields the user never visited, and returns false if any handler cancelled.

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 1 is about. You can open [module-01.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 ValidationClinic 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 1 – "Validation Fundamentals in Wisej.NET" – of the LearnWisej Validation 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/validation-in-wisej-net/specs/module-01.md>

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

1. Project "ValidationClinic", 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: Create the ValidationClinic customer intake form with `txtName` and `txtEmail` TextBoxes, a Save button and a Cancel button. Attach Validating handlers that set `e.Cancel = true` and an ErrorProvider message when the name is blank or the email has no @, clear the message when the value is valid, and trim the name in its Validated handler. Set the form's AutoValidate mode, make sure Cancel does not cause validation so the user can always leave, and call `ValidateChildren()` at the top of `btnSave_Click` so Save stops when any field is invalid. Finish with a short lab note that explains the focus behavior you chose.
3. Produce these deliverables, and tell me where each one lives in the solution:
 - Customer intake form with `txtName` and `txtEmail`, a Save button and a Cancel button
 - Validating handlers that set `e.Cancel = true` and an error message for a blank name and an invalid email
 - Validated handler that trims the accepted name
 - Save handler that calls `ValidateChildren()` and stops when it returns false
 - Lab note naming the AutoValidate mode chosen and why Cancel still works on an invalid form
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 remove `e.Cancel = true` from `txtEmail_Validating` but keep the `SetError` call. Describe exactly what the user sees when they tab away from "maria.example.com" and then click Save.
- Which of your checks run while the user is editing, and which run only because Save called `ValidateChildren()`? Give one user action that reaches Save with a field whose handler has never run.
- You switch the form from `EnablePreventFocusChange` to `EnableAllowFocusChange`. What changes for the user, what stays the same, and does `btnCancel.CausesValidation = false` still matter?

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 remove `e.Cancel = true` from `txtEmail_Validating` but keep the `SetError` call. Describe exactly what the user sees when they tab away from "maria.example.com" and then click Save.
- ✓ Which of your checks run while the user is editing, and which run only because Save called `ValidateChildren()`? Give one user action that reaches Save with a field whose handler has never run.
- ✓ You switch the form from `EnablePreventFocusChange` to `EnableAllowFocusChange`. What changes for the user, what stays the same, and does `btnCancel.CausesValidation = false` still matter?

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 ValidationClinic 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 ValidationClinic 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 ValidationClinic 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.