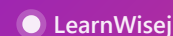


AI Coding Exercise

Build this module's OpsMonitor 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

Profile the two wait-heavy scenarios in WisejPerfLab. Run the ticket search under the Database tool and match each query to the UI action: the grid load issues one customer query per displayed row and the details query pulls large text columns nobody shows. Replace both with a single no-tracking projection query that selects only the grid columns and pages the result. Then run the CSV export under File I/O and .NET Async, find the blocking .Result call in the click handler and the repeated small writes, and move generation into an Application.StartTask background workflow that streams the file and reports throttled progress instead of pushing every step.

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

- ✓ Database trace matching queries to UI actions, with the N+1 query count before the fix
- ✓ Single no-tracking projection query with paging replacing the per-row and over-fetching queries
- ✓ File I/O and .NET Async evidence for the export, naming the blocking call and the write pattern
- ✓ Export moved to a background task that streams output and pushes progress only at meaningful steps
- ✓ Before/after query count, query duration and export wall-clock time, with remaining risks

Why it is shaped this way

Module 2 measured 1,840 ms of elapsed time and found only 300 ms of CPU. Module 3 removed most of that CPU and the screen is still not fast, because the missing 600 ms was never computation at all — it was waiting. This is the most under-diagnosed category in a business application: the server is idle, the CPU graph is flat, and the user is looking at a spinner. Somewhere a query is running once per row, a details form is pulling a column nobody displays, or a click handler is blocking on.Result while a thread-pool

thread does nothing. The tools that see this are Database, File I/O,.NET Async and Instrumentation, and this module is where WisejPerfLab's remaining latency finally has a name.

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
Database tool	Records the ADO.NET and EF Core queries a scenario issues, with their durations and order, so each query can be matched to the UI action that caused it.
N+1 query	One query to list, then one more per returned item; the list feels fine with ten rows and collapses at a thousand, and it is invisible to CPU profiling.
over-fetching	Selecting columns or rows the screen never shows — large text and binary columns are the usual offenders, and they cost query time, transfer and memory.
no-tracking query	A read-only query that does not set up change tracking, appropriate for grids and lists that display data rather than edit it.
.NET Async tool	Shows async activities and where they wait, which is how an accidentally sequential or blocked chain becomes visible.
blocking on async	Calling.Result or.Wait() from a handler; it holds a thread while waiting and, under load, produces symptoms that look like thread-pool starvation.
background export	Long file or report generation moved off the immediate click path with Application.StartTask, reporting progress at meaningful intervals rather than every step.

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 OpsMonitor 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 – "Database, File I/O, Async and External Waits" – of the LearnWisej Performance & Profiling 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/performance-and-profiling/specs/module-06.md>

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

1. Project "OpsMonitor", 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: Profile the two wait-heavy scenarios in WisejPerfLab. Run the ticket search under the Database tool and match each query to the UI action: the grid load issues one customer query per displayed row and the details query pulls large text columns nobody shows. Replace both with a single no-tracking projection query that selects only the grid columns and pages the result. Then run the CSV export under File I/O and .NET Async, find the blocking `.Result` call in the click handler and the repeated small writes, and move generation into an `Application.StartTask` background workflow that streams the file and reports throttled progress instead of pushing every step.
3. Produce these deliverables, and tell me where each one lives in the solution:
 - Database trace matching queries to UI actions, with the N+1 query count before the fix
 - Single no-tracking projection query with paging replacing the per-row and over-fetching queries
 - File I/O and .NET Async evidence for the export, naming the blocking call and the write pattern
 - Export moved to a background task that streams output and pushes progress only at meaningful steps
 - Before/after query count, query duration and export wall-clock time, with remaining risks
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 search now issues one query instead of two hundred and one, but the elapsed time only halved. Where is the rest of the time, and which tool would you use to find out?
- Your colleague says the export is fine because `.Result` "only blocks one thread". Explain what happens at fifty concurrent exports and which screens the user would notice slowing down.
- Which of the things this lab caches is safe to share between sessions and which is not, and what is the rule you used to decide?

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 search now issues one query instead of two hundred and one, but the elapsed time only halved. Where is the rest of the time, and which tool would you use to find out?
- ✓ Your colleague says the export is fine because.Result "only blocks one thread". Explain what happens at fifty concurrent exports and which screens the user would notice slowing down.
- ✓ Which of the things this lab caches is safe to share between sessions and which is not, and what is the rule you used to decide?

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 `Performance and Profiling 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 OpsMonitor 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/performance-and-profiling/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 OpsMonitor 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.