

AI Coding Exercise

Build this module's FieldOps 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



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 a FieldOps 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

Package FieldOps for Windows: a WebView2 shell with a native window, custom title and tray icon, a bridge that lets the app open a native folder picker to export work-order reports and show a native notification when a new order arrives, a configuration that points the shell at the remote server with a local fallback, and an installer with an update check on launch.

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

- ✓ WebView2 desktop shell with window and tray
- ✓ Native bridge for folder picker and notifications
- ✓ Remote and local server configuration
- ✓ Installer package
- ✓ Update check on launch

Why it is shaped this way

FieldOps technicians spend an entire shift inside the app: a laptop in the van, a docking station at the depot. The browser build from the first two modules does the work, but it has no icon in the Start menu, no presence in the notification area when a job is assigned, no way to drop an exported report into a folder the technician picked, and nothing IT can push through their usual software channel. This module wraps that same application, unchanged, in a Windows shell that supplies exactly those four things.

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
WebView2 control	The Chromium engine Microsoft ships as the redistributable Evergreen Runtime, embedded in a WinForms or WPF window so your page renders exactly as in Edge.
desktop shell	A small native executable that owns the window, the tray icon and the OS calls, and points a WebView2 control at your Wisej.NET application.
window chrome	The title, icon, minimum size, remembered position and single-instance behaviour that make the app feel installed rather than embedded.
message bridge	The two-way channel between page and shell: <code>PostWebMessageAsJson</code> down, <code>WebMessageReceived</code> up, or a host object published with <code>AddHostObjectToScript</code> .
JavaScript interop	How server-side C# reaches that channel: <code>Application.Eval</code> and <code>Application.Call</code> run script in the session's browser context and the answer returns as a client event.
server topology	Whether the shell navigates to a remote company server or starts a self-hosted Wisej.NET process on loopback — decided at launch and logged, never guessed.
update channel	A version endpoint the shell asks at start-up, plus the installer it fetches when the answer is newer than the version it is running.

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 FieldOps 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 – "Desktop Packaging with a WebView Shell" – of the LearnWisej Hybrid & Native Packaging 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/hybrid-native-packaging/specs/module-03.md>

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

1. Project "FieldOps", 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: Package FieldOps for Windows: a WebView2 shell with a native window, custom title and tray icon, a bridge that lets the app open a native folder picker to export work-order reports and show a native notification when a new order arrives, a configuration that points the shell at the remote server with a local fallback, and an installer with an update check on launch.
3. Produce these deliverables, and tell me where each one lives in the solution:
 - WebView2 desktop shell with window and tray
 - Native bridge for folder picker and notifications
 - Remote and local server configuration
 - Installer package
 - Update check on launch
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:

- A technician starts two exports within a second of each other and the shell answers both folder dialogs. What in your bridge design stops the second path from being written into the first report, and what would go wrong without it?
- Your Export button now opens a native folder dialog. Describe precisely what that same button does when the page is opened in Edge instead of the shell, and where in your code that difference is decided.
- The version endpoint is unreachable because the van has no signal. Argue for what the shell should do on that launch, and explain what a technician loses under each of the alternatives you rejected.

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:

- ✓ A technician starts two exports within a second of each other and the shell answers both folder dialogs. What in your bridge design stops the second path from being written into the first report, and what would go wrong without it?
- ✓ Your Export button now opens a native folder dialog. Describe precisely what that same button does when the page is opened in Edge instead of the shell, and where in your code that difference is decided.
- ✓ The version endpoint is unreachable because the van has no signal. Argue for what the shell should do on that launch, and explain what a technician loses under each of the alternatives you rejected.

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.

- ✓ **Add the second case.** *"Extend this FieldOps 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/hybrid-native-packaging/specs/module-03.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 FieldOps 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.