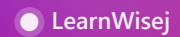


Capstone — Mini Helpdesk Application

The Capstone Project · Readable Code & Peer Review



The Capstone Project

What the capstone mini helpdesk must include — navigation, CRUD, dialogs, validation, theme polish — and how its pieces fit together.



Lesson goal

Understand the capstone: a small Wisej.NET **helpdesk/ticket app** that combines everything from the course — navigation, CRUD, dialogs, validation, theme polish, architecture notes and a deployment checklist.

1. What the capstone should prove

- The app has a clear shell with left navigation and separate screens.
- Tickets can be viewed, created, edited and deleted using a `DataGridView` and dialogs.
- Validation is close to the UI, but reusable business rules live outside random button events.
- The app has a consistent theme, clear spacing, readable labels and predictable command placement.
- The project includes architecture notes, a peer code-review checklist, next steps and deployment notes.

2. Capstone feature checklist

Feature area	Required evidence
Navigation shell	Dashboard, Tickets, Architecture, Code Review, Deployment and Next Steps open from one consistent left nav.
Ticket CRUD	Create, edit and delete ticket records; a <code>DataGridView</code> refreshes after changes.
Dialogs & validation	Create/Edit Ticket dialogs validate required fields and show clear success/error feedback.
Service layer	<code>TicketService</code> handles ticket operations instead of putting all logic in UI events.
UI polish	Theme applied, spacing cleaned up, labels readable, buttons grouped consistently.
Deployment readiness	Release checklist covers configuration, security, logging, theme/static resources and target notes.

3. Architecture and data flow

The capstone ties the whole app together. You should be able to explain how data moves through it end to end.

```
Program.cs      → starts Window1
Window1.cs     → shell, navigation, pages
Ticket.cs      → ticket data model
TicketDialog   → create/edit form + validation
TicketService  → CRUD logic and a fake repository
DataGridView   → displays tickets
Theme          → polished UI
Deployment     → release checklist
```

The flow: the user clicks Create/Edit, the dialog opens, validation runs, `TicketService` updates the ticket list, the `DataGridView` refreshes, and the status or deployment notes record what changed.



You're ready for the next reading

You know what the capstone must include and how its pieces fit together. Next: the developer habits that keep it readable, and the peer-review focus that grades it.

Readable Code & Peer Review

The developer habits that keep the capstone readable, and the peer code-review focus your project is graded on.



Lesson goal

Finish the capstone with junior-developer habits that make a Wisej.NET app maintainable, and know what a peer code review will look for before you submit.

1. Keep the capstone readable

The final project isn't just about making the screen work — it should show habits that keep the app easy to maintain.

Habit	What it looks like in the capstone
Clear control names	Names like <code>btnCreateTicket</code> , <code>dgvTickets</code> , <code>lblStatus</code> , <code>txtTitle</code> , <code>cmbPriority</code> — not <code>button1</code> or <code>label2</code> .
Small event handlers	Button clicks call helper or service methods; they don't hide hundreds of lines of business logic.
Reusable validation	A <code>TicketValidator</code> or helper can be tested and reused by create/edit workflows.
Intentional refresh	After a save/delete, refresh the grid and status message deliberately — no confusing side effects.
Safe errors	Show friendly user messages; keep internal details in developer logs or notes.

2. Peer code-review focus

- **Naming:** do classes, controls and methods clearly describe their job?
- **Separation of concerns:** is ticket logic in a service/validator instead of scattered across UI events?
- **Navigation:** can the reviewer follow how each page opens?
- **Usability:** are buttons placed where users expect them, with clear labels and feedback?
- **Deployment:** are configuration, logging, secrets, themes and release notes checked before review?



Show the whole course

The capstone should demonstrate everything you've learned — navigation, CRUD, dialogs, validation, theming and deployment — not just one isolated feature.

3. Suggested final submission

Submit a working project folder or GitHub repository, plus screenshots of the main screens and a short deployment/readiness note. The Next Steps screen should describe what a production-ready version would still need.



You're ready for the lab

Next: watch the capstone walkthrough, pass the final knowledge check, then build the mini helpdesk — a navigation shell, ticket CRUD with dialogs and validation through a service, a polished theme, and architecture/review/deployment notes.