TYMIQ

WinForms. Firts-week action plan

WinForms. First-week actions that reduce risk fast

Vendor and dependency audit (Day 1–2)

- Inventory all third-party UI controls and components.
- Check vendor support for **net8.0-windows / net9.0-windows.**
- Flag abandoned or version-locked controls immediately.
- Identify COM, ActiveX, native DLL, and printing dependencies.
- Confirm installer and deployment tooling compatibility.

Output A hard “supported / risky / blocked” list. No assumptions.

Identify extraction seams (Day 2–3)

- Locate business logic inside:
  - Form constructors.
  - Click and change event handlers.
  - BackgroundWorker and thread callbacks.
- Identify data access code coupled to UI state.
- Mark candidates for:
  - Domain services.
  - Application services.
  - Integration adapters.

Rule If logic cannot run without a Form instance, it is a migration risk.

Define the short-term target (Day 4–5)

- Decide explicitly:
  - Keep WinForms UI and modernize backend, or
  - Treat WinForms as a temporary shell.
- Lock the TFM decision (Windows-only is not optional here).
- Set UI behavior as a **compatibility constraint**, not a design goal.

Output A written decision that stops scope drift.

WebForms. First-week actions that prevent rewrite chaos

URL and routing preservation checklist (Day 1)

- Export all live URLs and routes.
- Identify:
  - Deep links used by users.
  - URLs consumed by integrations.
  - SEO-sensitive or bookmarked paths.
- Map current URLs to future ownership:
  - Legacy WebForms.
  - New ASP.NET Core app.

Rule No page rewrite without a URL decision.

Authentication and session audit (Day 1–2)

- Document current auth model:
  - Forms auth.
  - Windows auth.
  - Custom session logic.
- Identify where auth logic is enforced:
  - Global.asax.
  - Page events.
  - Custom HTTP modules.
- Decide early:
  - Centralize auth first, or
  - Keep legacy auth temporarily behind a gateway.

Output One source of truth for identity and session handling.

Business logic extraction seams (Day 3–4)

- Identify logic embedded in:
  - Page_Load and event handlers.
  - User controls (.ascx).
  - ViewState-dependent flows.
- Extract:
  - Data access.
  - Validation rules.
  - Integration calls.
- Expose logic through services or APIs before UI rewrite.

Rule Do not rewrite UI that still owns business rules.

Define the strangler boundary (Day 5)

- Decide which features move first:
  - Reporting.
  - Admin screens.
  - Read-only workflows.
- Configure parallel hosting:
  - Legacy WebForms stays live.
  - New ASP.NET Core app introduced alongside it.
- Establish rollback paths per feature.

Output A controlled coexistence plan, not a full rewrite fantasy.