

.NET APPLICATION TAKEOVER CHECKLIST

Use this checklist when you inherit, take over, or evaluate an existing .NET application. Work through each section in order. Items you can't answer yet are gaps to close in your first 30 days.

shannonmcvay.me/Services - Edit and share freely

01

First Contact

- ☐ Obtain source code repository access (Git, TFS, SVN, or ZIP archive)
- ☐ Obtain production deployment credentials
server login, FTP, Azure portal, or CI/CD pipeline
- ☐ Obtain database connection strings for production, staging, and dev environments
- ☐ Locate existing documentation
README, wiki, deployment notes, API docs
- ☐ Identify the original developer(s) — are they reachable for questions?
- ☐ Identify the deployment history — when was it last deployed, and by whom?

02

Technology Inventory

- ☐ Identify .NET version
.NET Framework 4.x vs .NET Core / .NET 5+/6/8/9/10
- ☐ Identify web framework
Web Forms, MVC, Razor Pages, Blazor, Web API only
- ☐ Identify ORM / data access layer
raw ADO.NET, Dapper, EF6, EF Core, stored procedures
- ☐ List NuGet packages and flag any that are deprecated or unmaintained
- ☐ Check for hardcoded connection strings or secrets in source code
- ☐ Note client-side framework
jQuery, Angular, React, Knockout, plain JS
- ☐ Identify authentication mechanism
Forms Auth, ASP.NET Identity, Windows Auth, JWT, OAuth/OIDC
- ☐ Check for third-party integrations
payment processors, email services, reporting tools, external APIs

- ☐ Where is it hosted?
on-premise IIS, Azure App Service, AWS, shared hosting, VPS
- ☐ Is there a CI/CD pipeline?
Azure DevOps, GitHub Actions, Octopus Deploy, or manual FTP
- ☐ Is SSL configured and is the certificate managed?
check auto-renew vs manual renewal
- ☐ Check application pool identity and permissions on the server
- ☐ Locate all configuration files
web.config, appsettings.json, environment-specific transforms
- ☐ Identify where secrets are stored
web.config, App Settings, Key Vault, environment variables
- ☐ Check for scheduled tasks or Windows Services that are part of the system

- ☐ Identify database server version and engine
SQL Server, PostgreSQL, MySQL, SQLite
- ☐ Locate all databases the application touches
- ☐ Determine if database schema is managed by migrations or maintained manually
- ☐ Check for undocumented stored procedures, views, or triggers central to business logic
- ☐ Verify a recent backup exists and has been tested for restore
- ☐ Review database permissions
does the app connect as SA or a scoped user?
- ☐ Note any linked servers or external data dependencies

- ☐ Does the project build cleanly from a fresh clone with no manual steps?
- ☐ Are there compiler warnings or suppressed warnings that should be reviewed?
- ☐ Is there test coverage?
unit tests, integration tests, or none
- ☐ Check error handling — does the app swallow exceptions silently?
- ☐ Check logging — is there structured logging or catch-and-ignore patterns?
- ☐ Review the largest or oldest controllers / classes for hidden complexity
- ☐ Note any TODO, HACK, or FIXME comments in the critical code path

Security Quick-Check

- ☐ Are connection strings and API keys excluded from source control?
- ☐ Is HTTPS enforced site-wide?
HSTS header, redirect middleware
- ☐ Does the app use [ValidateAntiForgeryToken] or equivalent on POST actions?
- ☐ Are user-supplied values HTML-encoded in views?
Razor @Model escapes by default
- ☐ Is the authentication cookie set to HttpOnly and Secure?
- ☐ Are there exposed admin endpoints with no authorization?
- ☐ Run: dotnet list package --vulnerable — are any NuGet packages flagged?

Business Continuity

- ☐ Can you deploy a fix to production in under two hours in an emergency?
- ☐ Is there a documented rollback plan if a deployment fails?
- ☐ Who else in the organization knows enough about this system to assist in a crisis?
- ☐ Is there a staging or QA environment that mirrors production?
- ☐ Are any licenses (tools, third-party libraries, hosting plans) expiring soon?

30-Day Stabilization Priorities

- ☐ Address any items from Section 06 that represent immediate security risk
- ☐ Move hardcoded secrets to environment variables or Azure Key Vault
- ☐ Ensure CI/CD can produce a deployable build from source alone — no manual steps
- ☐ Write a one-page system summary covering tech stack, deployment, and key contacts
- ☐ Establish a recurring error-log review cadence
weekly minimum
- ☐ Schedule a follow-up assessment with stakeholders at the 30-day mark