Cloud Contraptions
TRAINING & CONSULTING

To discuss this course and customizations:
Call: 434-509-5680 or Email: sales@cloudcontraptions.com

Test-Driven Development with .NET MAUI and Blazor Hybrid

Class Duration

35 hours of live training delivered over 5 days.

Student Prerequisites

- Professional C# programming experience
- Working knowledge of Blazor and Razor components, and MAUI Blazor Hybrid fundamentals (see *Build Cross-Platform Apps with .NET MAUI and Blazor Hybrid*)
- Familiarity with Git
- No prior testing experience required

Target Audience

Blazor and web developers building MAUI Blazor Hybrid apps who want the confidence of a test-first workflow. Ideal for teams sharing one Razor component library across native apps and the web, where a regression ships to every platform at once — and for organizations that require automated test coverage in their delivery pipeline. This is the test-driven companion to our *Build Cross-Platform Apps with .NET MAUI and Blazor Hybrid* course.

Description

Shared UI multiplies the value of every test: when one Razor component library ships to iOS, Android, Windows, Mac, and the web, a single well-written test protects five platforms — and a single regression breaks them all. This five-day course teaches test-first development for that world on .NET 10. Day one grounds students in TDD itself — red-green-refactor, xUnit organization, fast feedback — and the hybrid testability architecture: logic in services and components, platform access behind interfaces, and the dependency injection seams that make BlazorWebView-hosted UI testable. Day two is component TDD with bUnit: driving Razor components from failing tests, asserting on rendered markup, testing parameters, events, cascading values, and component state without ever launching an app. Day three tackles the hybrid-specific seams: mocking platform services (geolocation, camera, connectivity, permissions) injected into components, testing JS

To discuss this course and customizations:
Call: 434-509-5680 or Email: sales@cloudcontraptions.com

interop boundaries with faked IJSRuntime, HybridWebView messaging under test, and per-target render logic verified across native and web configurations. Day four covers the rest of the pyramid: TDD for SQLite repositories and offline sync, service-layer integration tests, authentication flows with faked MSAL, and end-to-end automation — Playwright against the web target, Appium against the native shell — with clear guidance on what belongs at each level. Day five puts it in the pipeline: CI workflows that run bUnit and integration suites on every commit and platform builds with E2E smoke tests on a cadence, coverage and mutation testing, and AI-assisted test generation within the TDD loop.

Learning Outcomes

- Practice red-green-refactor fluently and organize xUnit test suites that stay fast and maintainable.
- Architect hybrid apps for testability: logic in services, platform access behind interfaces, DI as the testing seam.
- Drive Razor components test-first with bUnit: rendering, parameters, events, and state.
- Test components that depend on cascading values, layouts, and routing.
- Mock platform services — geolocation, camera, connectivity, permissions — injected into components.
- Test JS interop boundaries with faked IJSRuntime, and HybridWebView messaging contracts.
- Verify per-target rendering logic across native and web configurations.
- Apply TDD to local data: SQLite repositories, migrations, and offline sync logic.
- Write integration tests for service layers and authentication flows with faked MSAL.
- Automate end-to-end tests: Playwright for the web target, Appium for the native shell.
- Build CI pipelines that run component and integration suites per commit and E2E suites on a cadence.
- Use coverage and mutation testing to keep the suite honest, and AI assistants to extend it without losing test-first discipline.

Cloud Contraptions
TRAINING & CONSULTING

To discuss this course and customizations:
Call: 434-509-5680 or Email: sales@cloudcontraptions.com

Training Materials

Comprehensive courseware is distributed online at the start of class. All students receive a downloadable MP4 recording of the training.

Software Requirements

.NET 10 SDK, Visual Studio 2026 (Windows) or VS Code with the .NET MAUI extension (Windows or Mac), Android SDK with an emulator, Appium, Playwright, and Git. A Mac with Xcode is required only for the iOS labs; all other labs run on Windows or Mac.

Training Topics

TDD Foundations

- Red-green-refactor: the discipline and the payoff
- Test organization with xUnit: fixtures, theories, and naming
- The test pyramid for shared-UI hybrid apps
- Why shared components raise the stakes — and the value — of every test

Architecting Hybrid Apps for Testability

- Logic in services and components, not host pages
- Platform services behind interfaces; DI in MauiProgram as the seam
- Razor class libraries structured for test coverage
- Humble objects at the WebView and platform boundaries

Component TDD with bUnit

- Driving components from failing tests: render, assert, refactor
- Parameters, EventCallbacks, and two-way binding under test
- Cascading values, layouts, and routing dependencies
- Component state, lifecycle, and async rendering
- Markup assertions that don’t break on every refactor

Mocking the Hybrid Seams

- Platform services in components: geolocation, camera, connectivity, share
- Permission flows under test: granted, denied, degraded
- Faking IJSRuntime: testing JS interop contracts
- HybridWebView messaging: bidirectional contracts as test cases

- Per-target render logic: native vs. web configurations

TDD for Local Data and Sync

- SQLite repositories test-first
- In-memory and on-disk test strategies
- Migrations under test
- Offline-first sync logic: conflict cases as test cases

Service and Auth Integration Testing

- Service-layer tests with real HTTP via test servers
- Resilience behavior under test: retries, timeouts, offline
- Authentication flows with faked MSAL; token storage contracts

End-to-End Automation

- Playwright against the web target: fast, reliable browser E2E
- Appium against the native shell: selectors, waits, and stability
- Sharing E2E scenarios across targets
- What belongs in E2E tests — and what does not

CI/CD for Tested Hybrid Apps

- Per-commit pipelines: bUnit and integration suites
- Platform builds with E2E smoke tests on a cadence
- Coverage reporting and mutation testing as honesty checks
- Test results as release gates

AI-Assisted Testing

- Generating bUnit and integration tests with AI assistants — within the TDD loop
- Reviewing AI-generated tests for real assertions
- Coverage-gap analysis with AI tools