Cloud Contraptions
TRAINING & CONSULTING

To discuss this course and customizations:
Call: +1 434-509-6890 or Email: sales@cloudcontraptions.com

Testing .NET MAUI Apps: XAML and Blazor Hybrid

Class Duration

21 hours of live training delivered over 3 days.

Student Prerequisites

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

Target Audience

C# developers shipping .NET MAUI apps written in XAML, Blazor Hybrid, or both, who want automated tests at every level. Typical apps include field tools, data-collection apps, and line-of-business clients.

Description

This three-day course teaches test-driven development for .NET MAUI apps written in XAML, Blazor Hybrid, or both. Students first make apps testable with thin views, MVVM view models, and platform services behind interfaces. They unit test view models, navigation, and data layers with xUnit v3 and NSubstitute, and Razor components with bUnit. Platform test projects then run MSTest tests inside an app on emulators, simulators, and devices. The course ends with Appium and Playwright UI tests, CI pipelines, coverage, and mutation testing.

Learning Outcomes

- Practice red-green-refactor on MAUI solutions
- Organize xUnit v3 test suites for MAUI projects
- Structure MAUI apps for testability with MVVM and dependency injection

Copyright 2026 Cloud Contraptions LLC – All Rights Reserved

- Unit test view models, commands, and validation without launching the app
- Test Shell navigation logic behind an abstraction
- Fake permissions, geolocation, connectivity, and media APIs
- Test local data, offline sync, and service layers
- Drive Blazor Hybrid components test-first with bUnit
- Verify JS interop and HybridWebView messaging contracts
- Run platform tests on emulators, simulators, and devices with `dotnet test`
- Automate UI tests with Appium and Playwright
- Build CI pipelines that test every target
- Check suite quality with coverage and mutation testing

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 (LTS) and .NET 11 SDK with the .NET MAUI workload (the Android and Apple test project templates require .NET 11), 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 a current Xcode release is required only for iOS and Mac Catalyst testing; everything else runs on Windows or Mac.

Training Topics

Testing Foundations for MAUI

- The red-green-refactor cycle
- The test pyramid for mobile apps
- Host unit tests, platform tests, and UI tests
- xUnit v3 on Microsoft.Testing.Platform
- Fixtures, theories, and naming
- Test tooling in Visual Studio and VS Code
- Why mobile suites get slow and flaky

Architecting MAUI for Testability

- Dependency injection in MauiProgram

- Thin views and testable view models
- Multi-targeting shared code for host tests
- Platform services behind interfaces
- Humble objects at the platform boundary
- Handlers and in-process tests

View Models and MVVM Under Test

- Driving view models from failing tests
- Observable properties and relay commands
- CanExecute logic
- Async command states and cancellation
- Validation with ObservableValidator
- Converters as pure functions
- Messaging between view models

Navigation Under Test

- Shell navigation behind an abstraction
- Testing route logic and route templates
- Testing returned results
- Navigation guards and login redirects
- Modal flows and deep links
- Thin navigation adapters

Mocking the Platform

- Mocks, stubs, and fakes with NSubstitute
- Permission flows through `IPermissions`
- Granted, denied, and degraded paths
- Faking geolocation, sensors, and connectivity
- MediaPicker behind an interface
- Capture, cancellation, and storage errors
- Faking time with TimeProvider
- Avoiding over-mocking

Data and Services Under Test

- SQLite repositories test-first
- In-memory vs. on-disk SQLite tests
- Preferences and SecureStorage abstractions
- Offline sync conflicts as test cases

Copyright 2026 Cloud Contraptions LLC – All Rights Reserved

- Connectivity transitions as test cases
- Service tests with in-process test servers
- Testing retries, timeouts, and circuit breakers
- Authentication flows with faked MSAL

Blazor Hybrid Components with bUnit

- Driving Razor components from failing tests
- Parameters, EventCallbacks, and two-way binding
- Cascading values
- Faked platform services in components
- Routing and NavigationManager fakes
- Back-button behavior
- Forms and validation with bUnit
- Per-target render logic
- Markup assertions that survive refactoring

JS Interop and HybridWebView Contracts

- Faking IJSRuntime
- Testing JS interop call contracts
- HybridWebView messaging contracts
- Message serialization and versioning
- Error paths in messaging
- Keeping interop thin

Platform Tests on Devices

- Host tests vs. platform test projects
- The `androidtest` and `iostest` templates
- The `maccatalysttest`, `macostest`, and `tvostest` templates
- MSTest platform tests with `dotnet test`
- Running on emulators, simulators, and devices
- Device selection and test filtering
- Testing platform APIs and handler behavior
- Release-build checks for trimming and NativeAOT

UI Automation

- Appium and its .NET client
- Appium drivers for Android, iOS, Windows, and Mac
- AutomationId and selectors

Copyright 2026 Cloud Contraptions LLC – All Rights Reserved

To discuss this course and customizations:
Call: +1 434-509-6890 or Email: sales@cloudcontraptions.com

- Waits and test stability
- Page objects for native screens
- Playwright for the shared web target
- Accessibility checks in UI runs
- What belongs in UI tests

CI for Tested MAUI Apps

- Per-commit host and bUnit suites
- Scheduled device and UI smoke tests
- Emulators and simulators in CI
- macOS build agents
- Cloud device farms
- Test results as release gates

Coverage, Mutation, and AI-Assisted Testing

- What coverage tells you and what it hides
- Mutation testing
- AI-generated tests inside the TDD loop
- Reviewing AI-generated tests

Copyright 2026 Cloud Contraptions LLC – All Rights Reserved