Faysal Aziz

Robot Framework and AI for QA: Hands-on iPhone Testing

Build readable web, API and mobile tests with Cursor, then apply them to physical iPhones.

FA-0829 · 2 days

Software Development · Agentic & Generative AI

Develop maintainable Robot Framework tests with AI assistance, meaningful assertions and guided iPhone automation using Appium and XCUITest.

Introduction

Effective automation starts with clear test intent, meaningful checks and reliable evidence when something fails. This practical workshop helps QA professionals develop Robot Framework tests they can understand, execute, troubleshoot and maintain. AI assistance supports the work, while participants retain responsibility for reviewing generated code and verifying test results.

The programme begins with how AI works, its risks and limitations, and the importance of providing relevant context and clear prompts. Participants then apply these principles through Cursor while developing reusable automation for web applications, APIs and mobile apps. The final section focuses on iPhone testing using MacBooks, physical iPhones and the Appium XCUITest toolchain.

The training emphasises practical engineering judgement, maintainable test design and skills applicable to everyday QA work.

Guided individual practice and collaborative activities build a focused set of Robot Framework tests against prepared applications. Participants use their existing Python, command-line and client-server knowledge to practise reusable keywords, test execution and failure diagnosis.

Learning outcomes

By the end of the course, participants will be able to:

- Explain how AI assistants work and recognise their limitations and risks.

- Provide relevant context and clear prompts for AI-assisted test development.
- Translate manual test cases into readable Robot Framework tests with meaningful assertions.
- Organise reusable keywords, resource files, configuration and test data.
- Develop and execute web, API and mobile tests using the appropriate Robot Framework libraries.
- Interpret reports, logs and failure evidence to troubleshoot test failures.
- Use Cursor to generate, explain, refactor and debug tests while verifying its output.
- Configure and execute guided iPhone tests using AppiumLibrary and the XCUITest driver.

Prerequisites

- Working QA experience: independently write test cases, execute tests and report defects.
- Good working knowledge of Python: write and run functions, import modules, manage packages and virtual environments, and interpret tracebacks.
- Good working knowledge of the iPhone ecosystem, including app installation, permissions and device settings.
- Comfort with POSIX systems and CLI tools: navigate files, manage permissions and environment variables, and run commands.
- Working understanding of client-server architecture, endpoints, ports and remote connections.
- Experience connecting to remote environments, transferring files and executing commands independently.
- A MacBook with compatible macOS and Xcode installed, plus permission to install compatible Python, Robot Framework, Node.js, Appium, XCUITest driver, Git and Cursor versions.
- A physical iPhone with a suitable data cable and permission to pair with the MacBook, enable Developer Mode and UI Automation, and trust the required development signing identity.
- Valid signing and provisioning access for the practice iPhone application and WebDriverAgent.
- Cursor access with AI features enabled under the organisation’s approved settings.
- Access to suitable practice web applications, APIs, an installable iPhone test build and authorised test accounts.
- Reliable internet access to the approved development tools and services.
- Authorised SSH access where remote practice environments are used.

Training outline

1. 1. AI Fundamentals, Risks, Context and Prompting

1. How AI assistants and language models work
2. AI limitations, hallucinations, privacy and security risks

3. Providing relevant project context and supporting evidence
4. Clear prompts, acceptance criteria and output verification

2. 2. Robot Framework Foundations and Test Execution

1. Project environment and dependency verification
2. Test structure, syntax, variables and library imports
3. Keywords, arguments, return values and assertions
4. Command-line execution, tags, reports and logs

3. 3. Writing Reusable and Maintainable Tests

1. Translating manual test intent into readable automation
2. User keywords, resource files and shared configuration
3. Setup, teardown, independent test data and data-driven templates
4. Refactoring duplication and maintaining tests after application changes

4. 4. AI-Assisted Automation with Cursor

1. Repository context and project rules
2. Generating tests from test cases and acceptance criteria
3. Explaining and refactoring Robot Framework and Python code
4. Diagnosing failures, reviewing changes and verifying assertions

5. 5. Web Application Testing with SeleniumLibrary

1. Browser sessions, element inspection and stable locators
2. User interactions and meaningful assertions
3. Explicit waits, synchronisation and flaky-test prevention
4. Hands-on web test development, execution and troubleshooting

6. 6. API Testing with RequestsLibrary

1. HTTP requests, headers, authentication and JSON payloads
2. Status codes, response content and negative-path assertions
3. Hands-on API test development, execution and test-data cleanup

7. 7. Mobile Application Testing with AppiumLibrary

1. Appium client-server architecture, drivers and sessions
2. Device capabilities, element inspection and mobile locators
3. Mobile interactions, waits and assertions
4. Mobile test execution and failure evidence

8. 8. iPhone Testing

1. MacBook, Xcode, iPhone pairing and Developer Mode checks
2. XCUITest driver, WebDriverAgent, signing and provisioning
3. iPhone application configuration and element inspection
4. Hands-on iPhone test execution, assertions and troubleshooting

Disclaimer

The course outline is a guide. Content, sequence and activities may be adjusted to participant readiness, available systems, technical constraints and the agreed learning objectives.

© 2026 Faysal Aziz · Kuala Lumpur