Your Continuous Testing Cheat Sheet for Mobile App Quality

Shift Gears • Shift Left

This cheat sheet outlines the key steps to build a hands-free, continuous mobile application testing project within your CI pipeline. The aim is to help you validate and perform quality checks on hardened applications.

What You Need to Start Mobile App Testing:

1. **Application Under Test**
   - A **mobile application,** ideally hardened using Digital.ai Application Security
2. **Test Infrastructure**
   - Access to **mobile devices** (either local or remote)
   - Clearly **defined device and OS coverage requirements**
   - A **test orchestration or grid execution platform** (e.g., Digital.ai Testing)
3. **Test Development Tools**
   - Appropriate **mobile test development tools,** such as Appium Studio, Appium, XCTest, or Espresso
   - A suite of **test cases** that cover functional, performance, accessibility, and basic security validation
4. **DevOps & CI/CD Toolchain**
   - A **code repository** (e.g., Git) to manage and version control test scripts
5. **Build Automation Tools**
   - A build automation tool (e.g., Maven or Gradle) to compile and package apps
   - A **CI tool** (e.g., Jenkins) to run tests automatically as part of your development pipeline

This cheat sheet utilizes Appium Desktop, Eclipse, Git, Jenkins, Jira, and Digital.ai Testing.

Steps:

1. Develop Tests in Your Environment

STEP 1

1. Connect your test development environment to a physical mobile device or emulator.
2. Record user flows manually or edit your existing test cases to cover specific functionalities.
3. Execute tests locally within the environment to ensure they function as expected and reflect desired behavior.
4. Export the generated test code and assets for your Integrated Development Environment (IDE) to further develop them.

2. Set Up Your Automation Project (e.g., Eclipse/IntelliJ)

STEP 2

1. Initiate a new Java class or project within your IDE as your test automation entry point.
2. Initialize a Git repository for version control and clone it in your local workspace.
3. Create a Java Gradle project and add essential automation framework dependencies to your build file.
4. Build a foundational test framework structure that includes page object models, utility classes, and reporting.
5. Configure your test project to with a remote test execution grid using URL and access credentials for scalable execution.
6. Integrate your exported test cases into the project, defining target platforms and devices needed for execution.
7. Customize test execution settings including defining which suites to run, parallelization and test data management.
8. Add a unique build ID or versioning key to your tests that runs traceability and reporting.
9. Validate project setup by executing tests locally within your IDE and test functionality before CI integration.

3. Integrate with Your CI Tool (e.g., Jenkins)

STEP 3

1. Establish the connection between your Git repository and CI tool to enable automated code polling and build triggering.
2. Create a new CI job that will orchestrate the build, test execution, and reporting.
3. Define necessary environment variables for the build within your CI configuration.
4. Configure parameterized builds for flexible execution, like running specific suites or targeting different environments.
5. Execute the CI job manually from Jenkins to execute tests and review initial reports.

4. Automate Continuous Testing Execution

STEP 4

1. Configure automated job execution triggers (e.g., new code commits, scheduled cron jobs, webhooks) within your CI pipeline.
2. Rigorously verify that triggers function and that test reports are automatically generated, presented, and analyzed, **including security posture and hardening effectiveness.**

5. Monitor KPIs for Continuous Improvement

STEP 5

1. Create a process for continuous monitoring and analysis of KPIs. Continuously track metrics like feedback time, quality trends, and **Security KPIs to drive** ongoing optimization and continuous improvement.

The Digital.ai Difference

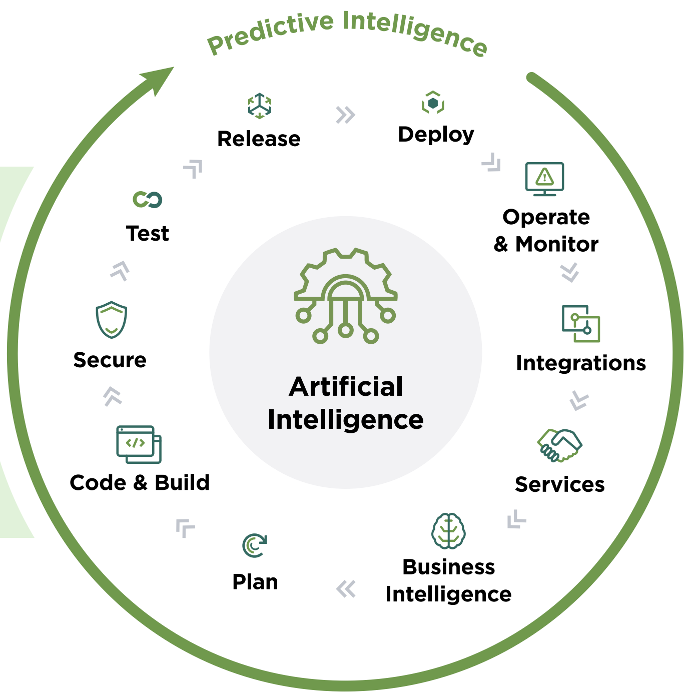
One platform that automates release pipelines and integrates complex toolchains.
Unify app delivery, integrate existing tools & scale across any environment.

Automated mobile app testing and security designed to scale.
Provide secure, high-quality apps through better automated testing & app protection techniques.

Built-in AI, intelligence, compliance, and governance across all software delivery workflows.
Centralize data, optimize processes, and gain augmented insights for faster, safer software delivery.

Digital.ai AI-powered DevSecOps platform

Helping you harmonize software delivery



About Digital.ai

Digital.ai is the only AI-powered software delivery platform purpose-built for the enterprise, enabling the world's largest organizations to build, test, secure, and deliver high-quality software. By unifying AI-driven insights, automation, and security across the software development lifecycle, Digital.ai empowers enterprises to deliver innovation with confidence. Trusted by global 5,000 enterprises, Digital.ai is redefining how enterprises build better software in an AI-driven world.

Additional information about Digital.ai can be found at digital.ai/ and on Twitter, LinkedIn and YouTube.

Learn more at Digital.ai

digital.ai™