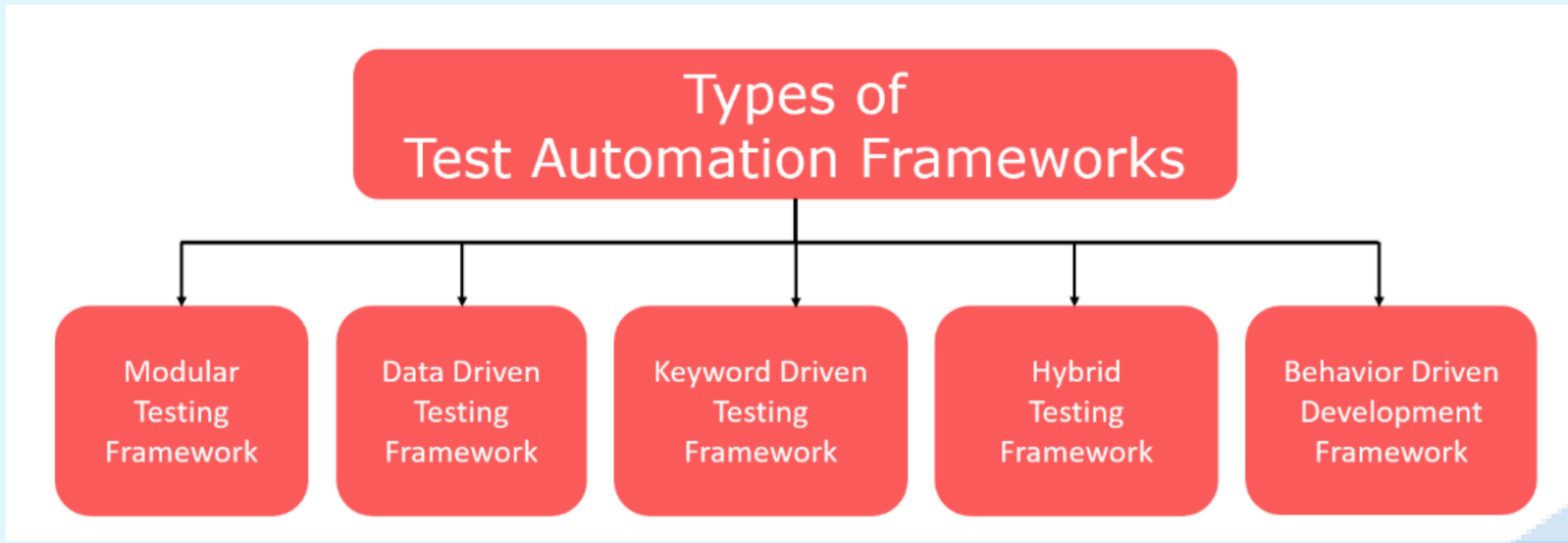

Sreenidhi Rajakrishnan
Automation Architect
@sreenidhirajakrishnan

Types of Test Automation Framework



Framework Type	Core Idea	Key Characteristics	Best Used When	Real-World Example
Linear (Record & Playback)	Tests are written sequentially without reuse	No modularity, no abstraction, low maintenance effort initially	Small POCs or quick demos	Recording login → checkout flow directly
Modular Framework	Application split into independent modules	Reusable functions, better maintainability	App has stable modules	Separate login, search, payment scripts
Library Architecture Framework	Common functions stored in libraries	High reusability, centralized utilities	Repeated actions across tests	login(), logout() in utility class
Data-Driven Framework	Test data separated from test logic	External data sources (Excel, CSV, JSON)	Same test with multiple datasets	Login test with 50 username/passwords
Keyword-Driven Framework	Tests driven by keywords instead of code	Business-readable keywords, abstraction layer	Non-technical users involved	Keywords like CLICK, ENTER, VERIFY
Hybrid Framework	Combination of multiple frameworks	Flexible, scalable, enterprise-ready	Large real-time projects	Data + Keyword + POM
Page Object Model (POM)	UI pages represented as objects	Clean separation of UI & test logic	UI automation at scale	LoginPage, HomePage classes
Behavior-Driven Development (BDD)	Tests written in business language	Gherkin syntax, collaboration focused	Agile teams, stakeholder visibility	Given-When-Then scenarios
TestNG / JUnit Framework	Annotation-based execution control	Parallel runs, grouping, reporting	Java test automation	@Test, @BeforeMethod
Playwright / Cypress Native Framework	Tool-native best practices	Auto-waits, fixtures, parallelism	Modern web apps	Playwright test fixtures
Robot Framework	Keyword-based, tabular syntax	High readability, library-driven	Teams preferring low-code	SeleniumLibrary keywords
API Automation Framework	Focused on backend testing	Request/response validation	Microservices & APIs	REST API regression suite
Microservices Test Framework	Service-level isolation testing	Contract & integration focus	Distributed architectures	Contract testing per service
CI/CD Integrated Framework	Built for pipelines	Headless runs, reports, triggers	DevOps-heavy teams	Jenkins + GitHub Actions
AI-Assisted Framework	Self-healing & intelligent execution	Locator healing, test optimization	High-change UI systems	AI-powered locator recovery

Extended Version for 2026