Sreenidhi Rajakrishnan
Automation Architect
@sreenidhirajakrishnan

Test Automation Framework

Section Name	Purpose	What It Includes	Real-Time Scenario (How to Explain in Interview)
Test Cases / Test Scripts	Execute automated scenarios	Test flows, validations, assertions	"We automated critical user journeys like login, search, and checkout as individual test scripts."
Framework Core / Base Layer	Common setup & teardown	Driver/session init, global hooks	"We initialized browser/session once in a base class and reused it across all tests."
Page Object / Screen Object Layer	Separate UI from test logic	Locators, page actions	"Whenever UI changes happened, we updated locators only in page classes, not tests."
Keyword / Action Layer	Reusability & readability	Login, navigation, common actions	"Actions like login and logout were written once and reused across multiple test cases."
Test Data Management	Externalize test data	JSON/CSV/Excel, data factories	"We used different test data sets for QA and UAT without changing test logic."
Configuration Management	Centralize execution control	URLs, browser, env flags	"Same test suite ran in QA and UAT by changing config values, not code."
Utilities / Helpers	Support common functions	Waits, file ops, date utils	"Custom wait utilities reduced flaky failures caused by dynamic elements."
Assertions / Validation Layer	Verify expected results	UI/API assertions	"We validated both UI messages and backend responses to ensure end-to-end correctness."
Error Handling & Recovery	Handle failures gracefully	Try-catch, retries	"If a test failed due to a temporary issue, retry logic handled it before marking it failed."
Logging Mechanism	Track execution flow	Step logs, error logs	"Logs helped us quickly identify whether a failure was due to test issue or application issue."
Reporting Module	Share execution results	HTML reports, screenshots	"After each run, stakeholders reviewed pass/fail status from automated reports."
Execution Controller / Runner	Control test execution	Suites, tags, parallel runs	"We ran only smoke tests during deployments and full regression during releases."
CI/CD Integration	Enable pipeline execution	Jenkins/GitHub Actions	"Automation ran automatically on every build, reducing manual effort."
Version Control	Team collaboration	Git repo, branches	"Multiple testers contributed safely using feature branches and pull requests."
Environment Management	Multi-env support	QA/UAT/Prod configs	"Environment-specific configs avoided hardcoding URLs and credentials."
Security & Secrets Management	Protect sensitive data	Env variables, encrypted secrets	"Credentials were stored securely and never committed to the repository."
Test Hooks / Lifecycle Management	Pre/Post execution activities	Before/After hooks	"Before hooks handled setup, after hooks captured screenshots on failure."
Retry & Stability Controls	Reduce flaky tests	Retry logic, smart waits	"Retries helped handle transient UI or network issues in CI runs."
Test Selection & Tagging	Run relevant tests	Smoke, regression tags	"We tagged tests so teams could quickly run only critical cases."
Artifacts Management	Store execution outputs	Screenshots, logs, videos	"Artifacts helped developers analyze failures without rerunning tests."
Documentation	Easy onboarding	Setup guide, README	"New team members could start automation quickly using framework documentation."
Maintenance & Governance	Long-term framework health	Refactoring, reviews	"Regular cleanup and reviews kept the framework stable as the app evolved."