Created by Lamhot Siagian

111 Automation Test Framework – Interview Prep Cheatsheet

Framework Architecture Design (10)

Question	Hint answer
What is a modular test framework?	Decompose into reusable modules (pages, APIs, utils) to isolate change and maximize reuse.
POM vs Screenplay pattern?	POM encapsulates pages; Screenplay models actors/tasks/abilities for composability.
Data-driven vs Keyword-driven?	Data-driven varies inputs; keyword-driven expresses steps as high-level actions.
Hybrid architecture benefits?	Combines POM + data/keyword + service layers for flexibility & maintainability.
Layered framework structure?	Test layer → domain/service/page layers → drivers/infra → utils/config.
SOLID in test design?	SRP for class roles; DIP for abstractions; favors mocks & swappable deps.
How to decouple UI from tests?	Use page/service abstractions; avoid direct locators in tests; centralize waits.
Designing for maintainability?	Encapsulation, naming, small classes, DRY, code review & linters.
When to introduce a DSL?	Readable test intent; domain verbs; reduces glue code duplication.
Service layer in UI frameworks?	Encapsulate business workflows; stabilizes tests vs UI churn.

Test Lifecycle Management (10)

Question	Hint answer
Global vs per-test setup?	Global for heavy init (drivers, containers); per-test for isolation & data.
Hooks in TestNG/JUnit/PyTest?	@BeforeSuite/@AfterSuite, @BeforeMethod, @BeforeEach; fixtures/scope in PyTest.
Idempotent teardown?	Always clean even on failure; guard nulls; use finally/TWR.
Environment bootstrap?	Load config, secrets, drivers, base URLs; verify health/ping deps.
Managing test states?	Use factories/fixtures; avoid cross-test coupling; reset external state.
Retry policy placement?	Framework-level listener/interceptor to centralize retries.
Resource lifetimes?	Scope drivers/clients per thread/test; ensure close/shutdown hooks.
Conditional skips?	Use tags/assumptions for env-specific or known issues; document rationale.
Flaky test quarantine?	Tag and exclude; track owner and fix-by date; CI report visibility.
Clock & timezone handling?	Freeze with test clocks; standardize UTC; inject time providers.

Test Data Management (10)

Question	Hint answer
Centralized data strategy?	Factories/builders + templates; separate seed from scenario data.
Parameterized tests?	DataProviders/CSV/JSON/DB; cover boundaries & negative cases.
Synthetic vs anonymized data?	Synthetic for safety; anonymize prod data via masking/tokenization.
Data isolation per test?	Namespacing, unique IDs, teardown cleanup, ephemeral schemas.
Managing referential integrity?	Use fixtures that create related entities atomically.
TDM tools/APIs integration?	Self-service datasets, on-demand provisioning, TTL cleanup.
Schema version drift?	Migrations for test DB; contract checks; backward compatible seeds.
Large data volumes?	Seed via bulk loaders; snapshot/restore; avoid UI seeding.
Event/analytics data testing?	Contracts for events; schema validation; idempotent producers.
Secrets in data files?	Never commit; use vault/CI secrets; runtime injection.

CI/CD Pipeline Integration (10)

Question	Hint answer
Choosing CI system?	Jenkins/GitHub Actions/GitLab/Azure—consider hosting, plugins, runners
Pipeline stages for tests?	Lint/unit → API/component → UI smoke (PR) → full regression (nightly).
Parallelization in CI?	Matrix + shards; split by duration; aggregate reports.
Environment variables & secrets?	Store in CI vault; inject at runtime; avoid printing.
Artifacts management?	Publish JUnit/Allure/HTML, traces/videos/logs with retention.
Infra-as-Code for test env?	Docker/K8s/Terraform; ephemeral per-PR envs with TTL.
Quality gates?	Fail on pass% threshold, flake rate, coverage, security scans.
Caching to speed builds?	Dependencies, browsers, Docker layers; cache keys by lockfiles.
Notifications & ownership?	ChatOps, codeowners, oncall; link artifacts in alerts.
Version pinning?	Lock runners/browsers/drivers for reproducibility; renovate updates.

Reporting & Logging (10)

Question	Hint answer
Choosing a reporting stack?	Allure/Extent/custom HTML; align with stakeholders & CI integration.
Minimum report contents?	Test status, steps, screenshots, logs, traces, env & commit.
Screenshot/trace policy?	On failure/retry; link from report to CI artifacts.
Structured logging?	JSON logs with runId, shardId, testId; parsable by ELK/Grafana.
Log levels strategy?	TRACE/DEBUG for dev, INFO normal, WARN/ERROR for issues; no secrets.
Flake analytics?	Track failure signatures, retries, stability trends; auto-quarantine.
Mapping to requirements?	Link tests to stories/RTM; add tags/IDs for traceability.
Custom dashboards?	Grafana/Kibana: pass%, p95 runtime, cost/test, top flaky.
Exporting results?	JUnit XML for CI; REST endpoints for test management tools.
PII & log hygiene?	Redact sensitive data; rotate logs; retention policy.

Exception Handling & Retry Logic (10)

Question	Hint answer
Global exception strategy?	Central listener/interceptor; categorize (env, data, app) for actions.
Retry on what conditions?	Only transient issues (network/timeouts); cap attempts; backoff.
Fail-fast vs retry-first?	Fail-fast for deterministic failures; retry-first for known flaky ops.
Capturing failure evidence?	On first failure: screenshot, HTML/DOM dump, network trace.
Timeout management?	Explicit waits, per-step timeouts; avoid global sleeps.
Graceful resource cleanup?	finally/TWR; close drivers; detach listeners; delete temp data.
Error classification?	Intermittent vs permanent; auto-route to owner; add labels.
Circuit breaker for flaky areas?	Short-circuit long flows; mark @wip; raise visibility.
Test failure triage?	Deduplicate by signature; recent changes first; bisect if needed.
Audit & traceability?	Record run IDs, screenshots/trace links, environment metadata.

Cross-Platform & Cross-Browser (10)

Question	Hint answer
Selenium vs Playwright?	WebDriver protocol & ecosystem vs auto-waits, tracing, browser control.
Appium vs XCUITest/Espresso?	Cross-platform server vs platform-native fast frameworks.
Capabilities/config strategy?	Central caps files; env variables; matrix by OS/browser/device.
Dealing with flaky gestures?	Stability waits, retries, idling resources, coordinate precision.
Grid vs cloud labs?	Self-hosted control vs scale & device matrix from vendors.
Visual testing?	Baseline diffs, tolerance, per-DPI assets; ignore dynamic regions.
Network throttling & geolocation?	Simulate bandwidth/latency; location/timezone checks.
Accessibility across platforms?	ARIA roles, labels, contrast, voiceover/talkback checks.
Browser/OS version policy?	Top market share + LTS; sunset old versions with data.
Certificate/SSL issues?	Trust stores for devices; flags in drivers; use test CAs.

Configuration Management (10)

Question	Hint answer
Config file formats?	.properties, .yaml, .json; choose based on nesting & tooling.
Environment switching?	Profiles (dev/qa/stage/prod) via flags; central base URLs/secrets.
Secrets management?	Vault/KMS/CI secrets; never commit; short-lived tokens.
Runtime overrides?	Env vars/CLI args; precedence: CLI > env > file defaults.
Feature flags in tests?	Toggle risky features; target cohorts; kill-switch for rollback.
Driver/binary versions?	Pin versions; automated updates with Renovate; compatibility matrix.
Regionalization configs?	Locale/timezone, currency/formatting; screenshot per locale.
Config validation?	Schema validation & failsafe defaults; startup sanity checks.
Multi-tenant configs?	Tenant-specific endpoints/keys; isolate data; tagging.
Config drift detection?	Checksums & CI tests; IaC source of truth; alert on changes.

Reusable Utilities & Helpers (10)

Question	Hint answer
Wait utilities?	Explicit waits, conditions; avoid Thread.sleep; polling with timeouts.
File helpers?	Temp dirs, downloads/uploads, CSV/JSON readers, cleanups.
Assertion wrappers?	Custom matchers; soft asserts; helpful failure messages.
Date/time helpers?	Clock injection; format/parse; time zone safe.
Random & unique data?	Prefix with run/shard; UUIDs; deterministic seeds when needed.
Network/API helpers?	HTTP clients, retries, schema validation, contract checks.
Serialization helpers?	JSON/YAML mappers; POJO/DTO builders; schema evolution.
UI interaction helpers?	Stable selectors, scrolling, iFrame/window switches.
Security test helpers?	Token generators, role switching, CSRF/headers utilities.
CLI/DB helpers?	Shell exec with capture; DB connections/pools with cleanup.

API & Backend Test Integration (10)

Question	Hint answer
When to choose API tests?	Faster and stable for business logic; reduce E2E load.
Combining UI + API?	Setup via APIs, verify via UI; or validate API side effects after UI.
REST Assured best practices?	Reusable request/response specs; schema validation; auth helpers.
Contract testing?	OpenAPI/JSON Schema; consumer-driven contracts (Pact).
Mocking 3rd parties?	Service virtualization/record-replay; chaos settings for latency/errors.
Data integrity checks?	DB assertions, events verification; idempotency & dedupe tokens.
Security in API tests?	OAuth2/JWT, scopes, rate limits, input validation, headers.
gRPC/GraphQL testing?	Client stubs, schema/SDL checks, query/mutation coverage.
Performance smoke?	p95 latency guards; small k6/Gatling smoke in CI.
Test data via APIs?	Backdoor endpoints for seeds; admin APIs; feature flags.

Parallel Execution & Scalability (10)

Question	Hint answer
Thread-safe driver/session?	ThreadLocal drivers; session manager; one session per thread.
Sharding strategy?	Split by duration/history; balanced shards; deterministic seeds.
Resource contention?	Limit workers vs CPU/mem; queue; isolate test data.
Dockerized grids?	Selenium/Playwright grid with autoscaling nodes; healthchecks.
CI concurrency control?	Job queues; max-parallel; back-pressure to labs.
Retry at scale?	Retry-once with tracing; quarantine flaky tests; owner notifications.
Cost controls?	Use spot/preemptible, schedule heavy runs off-peak, artifact TTL.
Caching at scale?	Browser/deps cache, Docker layers, warm pools of devices.
Metrics to track?	Pass%, p95 runtime, flake rate, cost/test, capacity usage.
Geo/timezone matrices?	Region egress, timezone assertion, locale screenshots.

Version Control & Collaboration (11)

Question	Hint answer
Branching model?	feature → dev → main; release/hotfix branches; enforce PR checks.
Code review essentials?	Architecture, testability, naming, docs, performance & security.
Commit hygiene?	Small atomic commits; conventional messages; link issues.
PR templates & checklists?	Ensure tests, docs, screenshots, risk/rollbacks noted.
Selective execution via tags?	@smoke @regression @wip; CI filters; critical-path pipelines.
Git hooks & linters?	Pre-commit format/lint/tests; pre-push sanity; enforce style.
Monorepo vs polyrepo?	Monorepo: unified tooling; Polyrepo: isolation; choose by org scale.
Managing test ownership?	CODEOWNERS, module OWNERS, dashboards with owners.
Knowledge sharing?	ADR docs, READMEs, playbooks, brown-bags, internal wikis.
Versioning of test assets?	Pin baselines, schema versions, API contracts, fixtures.
Security & access control?	Least-privilege on repos/runners/secrets; audit logs.

Lamhot Siagian