TR-05

Software Quality Assurance

COURSE BRIEF

SKILLS & TOOLS YOU WILL MASTER

Playwright + TypeScript | Selenium WebDriver | Postman + API Testing | Jira + TestRail | SQL + DB Testing | GitHub Actions

10 Reasons This Works.

Why students choose Saarathi - and how each card helps you ship a real career.

01

10 Max / Batch

Not 50. Not 30. Ten. Every student seen, heard, and unblocked - every class.

02

Sunday Open Classroom

Every Sunday the classroom is open. Come in, pair-program, get unstuck, or just focus.

03

Weekly Detailed Syllabus

Full syllabus from day one. Every week: what to read, what to build, what the outcome is.

04

Revision Every 5th Day

Every fifth class is a revision session. Nothing piles up or slips through.

05

Week 1: Mindset

The first week is about getting into the right headspace before touching a line of code.

06

3 Ways of Learning

Self-learning + collaboration + mentorship. All three built into every course.

07

1:1 Mock Interview

Real mock interview with your mentor in the final week. Actual hiring prep, not fluff.

08

CV + LinkedIn

Resume writing and LinkedIn optimisation. Leave with a profile that gets replies.

09

Real Projects

Every course ends with a deployed, publicly accessible project you can show employers.

10

Gate Assessment

Aptitude test before Day 1. Maps your strengths, weaknesses, learning style, and pace - then builds your personal plan.

saarathiacademy.com.np/software-quality-assurance-course-in-nepal

+977-9744442469 · +977-9761095364

Software Quality Assurance

Why This Course

This course trains the junior QA engineer profile employers actually hire: someone who can write structured test cases, manage defect lifecycles, test APIs (the behind-the-scenes connections apps use to talk to each other) with Postman, query databases with SQL, and use a Playwright automation suite (tests written as code that drive a real browser) wired into GitHub Actions CI/CD (pipelines that run those tests automatically on every code change) as a working QA tool. Automation is a built-in module of this QA course, not a separate track. Manual testing alone is not enough for modern entry-level QA roles, but jumping straight to automation without testing foundations produces engineers who write scripts without understanding what to test, and brittle bug reports no developer trusts. Pay scales with your skills, portfolio, and whether you work locally or for international/remote clients. We don't promise a salary.

This balanced 8-week program spends its first three weeks building the manual QA foundation a beginner needs before any automation: the testing mindset and why we test, SDLC and STLC, test case design, and the full bug lifecycle and developer-ready reporting with Agile QA. Week 4 adds just-enough HTTP, REST, Postman, and SQL literacy so you understand the request, response, and database state your automation later asserts against. Three focused weeks then teach Playwright automation with TypeScript, Page Object Model, fixtures, a hands-on Selenium WebDriver block, and GitHub Actions quality gates. AI runs through this course as **two separate skills, because employers treat them separately**. The first is AI as your tooling: Playwright bundles the **MCP** server and CLI as of 1.62, so from Week 5 you author tests with **Copilot**, **Claude Code** or **Cursor** driving a real browser, then run the full **Playwright Test Agents** loop (planner writes the plan, generator writes the tests, healer repairs them) and review every generated line. The second is **testing AI itself**, the skill Nepal's AI-shipping employers cannot currently hire for: chatbots and LLM features never answer the same way twice, so you learn gold sets, semantic and constraint-based assertions, LLM-as-a-judge scoring, the **RAG triad**, hallucination, bias and prompt-injection checks, and you ship an eval suite gated in CI with **promptfoo** or **DeepEval**. What does not change is the judgement of what to test and whether a result is right, which is the part employers still hire humans for. QA fundamentals practice questions run weekly from Week 1; the final week adds a timed mock test and portfolio assembly. Graduates leave with both a manual testing and bug-report portfolio and a Playwright automation framework running green in CI on GitHub: the two artefacts that cover junior QA and junior automation engineering roles.

Built against what Nepal actually advertises. Live Kathmandu QA vacancies ask for SDLC, STLC and defect lifecycle, Postman API testing, SQL database validation, a bug tracker **and** a test-management tool, functional, integration, regression and UAT execution, and a contribution to test automation. Mid-level posts add security testing of web applications. Nepal's tool stack

names **Selenium** first, then Cypress and Playwright, with **TestRail** or **Zephyr** for test management, **JMeter** next to **k6**, and **Jenkins** next to GitHub Actions. The biggest QA employers here are fintech (F1Soft, eSewa, Khalti, Fonepay), health and insurance outsourcing (Cotiviti, Deerwalk, Verisk), telecom (WorldLink) and product outsourcing (Leapfrog, Fusemachines). So this course teaches payment and transaction test scenarios as a first-class domain, ships a Selenium framework as a second public repo, runs a real test cycle in a test-management tool, and covers the OWASP Top 10 through a tester's lens.

And it takes the experience wall seriously. In Nepal, entry-level QA posts routinely say 6 months to 1 year, and some posts titled "Junior QA Engineer" ask for 2 years. A certificate does not clear that bar; verifiable public work does. Week 8 is built to manufacture the evidence a hiring manager can check in ten minutes: two public automation repos, a Jira defect history, a test-management run report, three real defects filed against real public software, a 20-company target list and five tailored applications sent before you graduate.

Who Is This Course For

- **Complete beginners** with basic digital literacy who want a full QA career foundation
- **Manual testers** transitioning into automation and wanting Playwright, API testing, and CI/CD on their profile
- **Fresh graduates** targeting junior QA Engineer, Software Tester, or QA Analyst roles
- **Career switchers** from non-technical backgrounds: business analysts, project coordinators, operations professionals
- **Developers** who want to understand quality from a tester's perspective and write better-tested software
- **Students** who want a portfolio-backed software quality assurance course in Kathmandu

Course Snapshot

PARAMETER	DETAILS
Course Code	TR-05
Duration	8 Weeks (2 Months)
Schedule	Monday to Friday, 2 Hours/Day
Total Hours	80 Hours
Batch Size	Maximum 10 Students

PARAMETER	DETAILS
Course Fee	NPR 28,200
Level	Beginner → Junior QA Engineer (manual + automation capable)
Outcome	Junior QA Engineer with Playwright **and** Selenium automation skills, plus API, SQL, security and payment-domain testing; automation is a built-in module of this QA course, not a separate track

Prerequisites

- No programming background required, TypeScript is taught from zero in Week 5 with a primer in Week 4 self-study
- A laptop with 8GB+ RAM
- Comfortable using a laptop and a browser, and willing to read an error message and search for the answer
- Saarathi Gate Assessment before Day 1, diagnostic, no pass or fail

Nothing to install beforehand. VS Code, Node.js, GitHub and Jira are set up together in the first class.

Assessments at a glance

Two exams bracket the course. The **Saarathi Gate Assessment** runs before Day 1 as a diagnostic with no pass or fail; it maps your learning style and pace. The **Saarathi Certification Exam** is sat at the end of the course, 2 hours and around 30 questions specific to this course in mixed formats (single choice, multi-select, true/false, numeric, ordering, matching, fill-in-the-blank, short or long answer, code where relevant, and short case studies). Pass mark is 60 percent and a pass is mandatory for the Saarathi Certificate of Completion. The exam is camera proctored: a working webcam must stay on for the full two hours, and stills are saved for your trainer to review. You can answer the questions in any order and come back to any of them, but every question needs an answer before you submit. Recommendation letters are discretionary, mentor-issued only to learners who pass the Certification Exam, and never guaranteed.

Your Learning Week

DAY	ACTIVITY
Mon–Fri	2-hour live class session (hands-on, project-driven) with mentor
Mon–Fri	2-hour self-study at home (assignments, practice, debugging)
Sunday	Open classroom for focused build time or rest
Day 5 (every week)	Within the 2-hour session, the last 1 hour is a built-in revision block for consolidation and doubt-clearing; new-topic load on Day 5 is intentionally lighter and hands-on tasks extend into the 2-hour self-study window

Every week's revision hour also runs 10 QA fundamentals practice questions mapped to the ISTQB CTFL chapters and **3 spoken interview drills**, questions asked verbatim, answered aloud, critiqued on structure. Interview practice runs from Week 1, not just in the final week, because "manual testing interview questions" is what every QA candidate in Nepal is frantically searching the night before, and doing it once is not preparation.

Real improvement happens when you build, break, test and re-submit between classes. The classroom gives structure; the hours between classes give depth.

Week-by-Week Curriculum

Phase 1 · QA Foundations: SDLC, Test Design & Manual Testing (Weeks 1–3, 30 Hours)

The genuine beginner base, given three unhurried weeks before any API or automation work: a third of the course. Week 1 is orientation, SDLC and testing theory; Week 2 is hands-on test design and manual platform testing; Week 3 is the bug lifecycle, Jira and Agile QA.

WEEK	FOCUS AREA	TOPICS
Week 1	QA Role, SDLC & Testing Fundamentals	- What a QA engineer does and why the role matters in a team - Telling apart the three words job posts mix up: QA (prevents defects), QC (checks the finished product) and Testing (the hands-on activity) - How software gets built stage by stage (SDLC phases) and where a tester fits in Waterfall (plan everything upfront) vs Agile (build in short cycles) - The **Software Testing Life Cycle** (STLC): the tester's own stages (analysis, planning, case design, setup, execution, closure) that

WEEK	FOCUS AREA	TOPICS
		job posts ask about - **Scrum** meetings, the most common **Agile** method: sprint planning, standups, sprint review, retrospective, what a tester says in each - The main testing types: functional (does it work), non-functional (is it fast and secure), regression (did old features break), smoke and sanity (quick health checks), exploratory (unscripted investigation), **UAT** (user acceptance testing) - Testing levels: unit (one piece of code), integration (pieces working together), system (the whole product), acceptance (ready for users) - Test strategy vs test plan vs test case: the big-picture approach, the per-release plan, the exact steps for one check - Risk-based testing and coverage: test the features most likely to break or hurt users first, then prove every requirement has a test with a requirement traceability matrix (**RTM**, maps requirements to test cases) - First look at **equivalence partitioning** (group similar inputs, test one per group) and **boundary value analysis** (attack the edges), drilled fully in Week 2 - **Git** & **GitHub** basics for QA, saving and sharing work (version control): clone, commit, push, pull requests - **What AI changed about this job, and what it did not.** Around 65% of QA professionals now use AI in testing. It scaffolds suites, drafts test matrices and heals broken locators. It does not decide what is worth testing, judge whether an answer is right, or defend a severity call. Running someone else's scripts is the exposed half of the job; designing tests, automating them and reviewing AI output is the half getting scarcer and better paid

| **Week 2** | Test Design Techniques & Manual Testing Mastery | - **Equivalence partitioning** hands-on: dividing real inputs into valid and invalid classes

- **Boundary value analysis** hands-on: testing the exact edges (minimum, maximum, just outside) on real forms

- **Decision table** testing: when conditions combine, laying out every combination and expected outcome in a table

- **State transition** testing: mapping legal and illegal moves between states, on login flows, status changes, workflow states

- Writing clear, reproducible test cases a stranger can re-run: precondition, steps, expected result

- **AI-drafted cases, human-pruned**: an LLM writes 20 candidates from a user story in seconds,

and your job is what it cannot do, cutting duplicates, catching the boundary it missed, rejecting cases that assert nothing. Graded on what you deleted and added, not on the prompt
- Exploratory testing vs scripted testing (pre-written steps), when each fits
- Organising exploratory work with charter-based session-based test management (**SBTM**): a written mission in a timeboxed session, then regression, smoke and sanity in sprints: choosing scope by impact analysis (what a change can break) and what earns a place in the smoke pack vs the wider regression pack
- Browser **DevTools** basics: **Chrome**'s inspector for network requests, responses and console errors while testing
- Test execution and result logging in a spreadsheet or Markdown report (**TestRail** at awareness level in Phase 4)
- Cross-browser testing: same behaviour in **Chrome**, **Firefox**, **Safari**, **Edge**
- Mobile testing: **Android** vs **iOS**, viewport testing (screen sizes), responsive design (layouts that adapt)
- Usability (easy to use) and accessibility (usable with disabilities) testing basics
- **Integration testing** (checking the seams where two features hand data over) and **UAT** support (running the session with a non-technical stakeholder), both named as day-job duties in Nepal job posts
- **Payment and transaction test scenarios**, the Nepal domain block: transfer limits, OTP entry and expiry, double-submit and idempotency (one tap, one debit), timeout mid-payment, failed and partial payments, refunds, and reconciling the screen against the database |

| **Week 3** | Bug Lifecycle, Jira & Agile QA | - Bug lifecycle stages, found to fixed: New, Assigned, In Progress, Fixed, Verified, Closed, Rejected, Reopened
- Severity (how badly it hurts users) vs Priority (how urgent the fix is), classifying defects correctly
- A good bug report: title, steps to reproduce, actual vs expected, severity, priority, environment (browser, device, build), attachments
- Writing developer-ready bug reports, actionable in minutes
- **Jira** (the issue tracker most teams use) for defect management: creating, linking, labelling, triaging (by urgency), tracking
- A real test cycle run end to end in a **test-management tool** (**TestRail**, **Zephyr** or **Qase** free tier): suites, runs, pass/fail/blocked results, a defect linked from a failed step, run report exported, then the same suite kept in the **Excel** format Kathmandu teams actually mail around
- Root cause analysis basics: separating symptoms from causes, digging to the real cause
- **AI in the paperwork half of QA**: turning a messy repro into a developer-ready bug report, summarising a failed run into a sprint update, clustering 50 failures into 3 root causes, drafting cases from acceptance criteria. The rule that survives a real team: AI drafts, a human verifies against the actual build, and the human's name is on it
- QA in sprint planning: breaking down user stories (short feature descriptions) for test scope
- Acceptance criteria review (rules a feature must meet): the right questions before sprint starts

- Test sign-off and release readiness: the go/no-go decision (ship or hold) backed by the QA metrics that inform it, pass rate, fail rate, defect density (bugs per area), test coverage, escape rate (bugs that reached users), defect age (time open)
- Where that decision sits in the **release management process** Nepal job posts name as a plus: release branches, staging sign-off, post-deploy smoke, what a rollback costs
- Writing QA status reports for managers and stakeholders in **Confluence**, the doc tool paired with Jira at most Kathmandu companies |

Phase 2 · API Testing & Backend Quality (Week 4, 10 Hours)

WEEK	FOCUS AREA	TOPICS
Week 4	REST API Testing with Postman & SQL for Testers	- **HTTP** fundamentals, the web's language: methods (its verbs), status codes (like 404), headers, request/response body - **REST API** concepts, the standard web API style: endpoints (the addresses you call), resources, authentication (API keys, Bearer tokens) - **Postman** setup (the standard API testing tool): workspace, collections (saved requests), environments, variables - Sending GET (read), POST (create), PUT (update), DELETE (remove) requests and inspecting responses - Negative testing, breaking the API on purpose: invalid inputs, missing auth, wrong methods, boundary values - Response body validation and the collection runner: automatically checking fields, types and values, then running a full test sequence end-to-end in one click (Postman's **Postbot** AI helper can draft assertions, reviewed like any AI output) - **SQL** for testers, asking the database questions: SELECT, WHERE, JOIN, COUNT, SUM - Writing queries to verify data state after API calls (did it really save) - INSERT, UPDATE and DELETE for test data setup and teardown (create, then clean up) - Running the same collection headless from the terminal with **Newman** (Postman's CLI), which is how it becomes a pipeline step in Week 7 - **Security testing for QA, the OWASP Top 10 through a tester's lens**, because Nepal mid-level posts ask for it: **SQL injection** and **XSS** on a real form, broken authentication and **IDOR** (changing an id in a URL to read someone else's record), missing authorisation on an endpoint the UI never calls, sensitive data leaking in a response or error trace, and missing rate limits on login and OTP. On a deliberately vulnerable practice app only, never a system you do not own - **Graded TypeScript primer (Day-5 self-study, checked before**

WEEK	FOCUS AREA	TOPICS
		Week 5): variables, functions, arrays, objects, async/await mini-exercises - **Capstone deliverable: Postman collection (20+ requests) + SQL verification queries + 3 written-up security findings**

Phase 3 · Test Automation with Playwright (Weeks 5–7, 30 Hours)

WEEK	FOCUS AREA	TOPICS
Week 5	TypeScript Basics & Playwright Foundations	- Why automate: what to automate vs keep manual, ROI thinking (is the test worth the time) - Your first real programming, **TypeScript**/**JavaScript** for testers: variables (named boxes), types, functions (reusable steps), arrays (lists), objects (labelled bundles) - Async/await and Promises: waiting for slow things (pages loading, servers replying) without freezing - **Node.js** project setup (JavaScript outside the browser), **npm** (installs packages), `tsconfig.json` basics - **Playwright** installation and first test project - Locator strategy, what to find on the page: `getByRole`, `getByText`, `getByTestId`, **CSS** selectors as a last resort - Actions, driving the browser: click, fill, check, select, keyboard input - Assertions, automatic pass/fail checks: `toBeVisible`, `toHaveText`, `toHaveURL`, `toBeEnabled`, `toHaveCount` - Auto-waiting: why arbitrary sleeps cause random failures, Playwright's smarter alternative - **Agentic test authoring from your first automation week**: Playwright bundles the **MCP** server and CLI (`npx playwright mcp`), so **Copilot**, **Claude Code** or **Cursor** can open a real browser and write a test against what is actually on the page instead of hallucinating selectors. Then the part that makes you employable: read every line, check the locator survives a re-render, confirm the assertion would fail if the feature broke, delete the noise. **You own every test with your name on the commit** - **Playwright** UI Mode (`--ui` watch runner) and trace viewer: stepping through and replaying a failed test like a video, timeline and **DOM** snapshots (frozen page copies), with the Copy-prompt button to hand errors to an LLM - Running the first end-to-end test suite green (every test passing)

WEEK	FOCUS AREA	TOPICS
		- **Graded coding strand 1 of 3 (self-study): 2-3 easy string/array problems**
Week 6	Page Object Model, Fixtures, Cross-Browser & Advanced Test Patterns	- **Page Object Model** (**POM**): each screen in its own typed page class so tests read like plain English and survive design changes - Structuring a maintainable framework anyone can navigate: tests/, pages/, fixtures/, data/ layout - Custom fixtures: write authentication (login) setup once, reuse across all tests - Test data builder pattern: typed test data via method chaining (linked calls, not long setup blocks) - Test tagging, running just one slice: @smoke, @regression, filtering with --grep - Cross-browser testing: **Chromium**, **Firefox**, **WebKit** (Safari's engine) projects in one config - Parallel execution (many tests at once) and test sharding (splitting across machines) for faster **CI** - Visual regression testing with `toHaveScreenshot` : a reference baseline screenshot plus pixel diff detection - Network interception with `page.route()` : mocked responses to test error states without the real backend - Auth state reuse: log in once with `globalSetup` , save storage state, skip login in every test - **Selenium WebDriver, your second framework, not a demo.** Nepal's job ads name Selenium first and Kathmandu hiring managers open your GitHub looking for a Selenium Page Object Model repo, so you rebuild the login and checkout flows in it: driver setup, locator strategies, **explicit waits** with `WebDriverWait` and why `Thread.sleep` causes the flakiness Playwright hides from you, a POM layer, **TestNG**-style structure named, shipped as **its own public repo**. Delivered in JavaScript/TypeScript with a side-by-side **Java** and **Python** reading card, so you can read the Selenium code an interviewer puts in front of you. Playwright stays the primary framework - **Graded coding strand 2 of 3 (self-study): 2-3 easy array/object problems**
Week 7	API Automation, CI/CD Quality Gates &	- **Playwright** `request()` for API test automation, no browser needed - Combining UI and API tests in one **Playwright** project - Database state validation via **SQL** in test teardown - **GitHub Actions**: `.github/workflows/tests.yml` running the

WEEK	FOCUS AREA	TOPICS
	Performance Testing	suite on pull requests, plus a `newman run` step so the Week 4 Postman collection gates merges too - **Jenkins** at read-and-explain depth, a real `Jenkinsfile` walked through stage by stage, because plenty of Kathmandu teams run Jenkins and will ask you about it - Installing and caching **Playwright** browsers in **CI** - Matrix sharding: splitting the suite across parallel CI jobs - Uploading failure artefacts: traces, screenshots, HTML report - Flaky test investigation: root-cause analysis vs blind retries - Required checks: blocking merges on a red pipeline - **k6** performance testing: one real load test against an API, reading p95 latency, throughput and error rate - **Apache JMeter**: the same endpoint load-tested with the tool Nepal job descriptions name first, thread groups, ramp-up, samplers, listeners, and the two result sets compared - **The Playwright Test Agents loop, run end to end on your own capstone**: planner explores the app and writes a Markdown plan into `specs/`, generator turns it into real test files in `tests/` from a `seed.spec.ts` context, healer runs the suite and repairs what broke. Chained they produce coverage overnight; ungoverned they produce 100 tests nobody trusts, so you review the diff like a code reviewer - **Testing AI features, the skill Nepal's AI employers cannot hire for.** An LLM never returns the same string twice, so `toHaveText` is useless. Instead: a **gold set** of curated input/expected pairs, **semantic similarity** and constraint-based assertions, **LLM-as-a-judge** scoring against a rubric you wrote, and for retrieval the **RAG triad** (faithfulness, answer relevance, context groundedness). Plus the failures that pull a fintech chatbot from production: hallucinated numbers, prompt injection, leaked system prompts, unsafe or biased output, and silent quality drift. One tool done properly: **promptfoo**, open source, declarative YAML so there is no new framework to learn, one line in GitHub Actions (**DeepEval** named as the Python alternative) - **Graded coding strand 3 of 3 (self-study): 2-3 mixed function/array problems; 6-9 easy problems total across Weeks 5-7** - **Capstone deliverable: Playwright framework on GitHub with CI green pipeline, the Selenium POM framework as a second public repo, and an eval suite for one AI feature gated in the same pipeline**

Phase 4 · Portfolio & Career Launch (Week 8, 10 Hours)

WEEK	FOCUS AREA	TOPICS
Week 8	Capstone Demo, Portfolio & Career Launch	- QA fundamentals consolidation: key concepts, glossary, test design vocabulary (weekly practice since Week 1) - Timed QA fundamentals practice test and weak-area review - Portfolio assembly: test plan, test case library (60+ cases), **Jira** bug history, test-management run report, **Postman** collection, **SQL** queries, security findings, **Playwright** suite, **Selenium** suite and **AI eval suite** - **Three real defects filed against real public software** (an open-source issue tracker, a public beta, or a programme that invites reports), because filed public defects are checkable in a way a certificate is not. Responsible disclosure rules covered first and not optional - **Positioning for the AI-era market**: employers reading junior applications respond to a public eval suite far more than a tool list, so yours gets a README stating what it measures and what it caught. Quantified over adjectives, "cut hallucinated responses on the FAQ bot from 18% to 3% on a 60-case gold set" beats "familiar with AI testing". Which artefact leads for a manual role, an automation role and an AI-testing role, since they are three different opening paragraphs - **The Nepal QA job campaign, run as work rather than advice**: where the jobs actually are (fintech, health and insurance outsourcing, telecom, product outsourcing, Baneshwor / Pulchowk / Tinkune startups), where they post (company career pages first, then merojob, KumariJob, JobAxle, LinkedIn Nepal), the honest market bands, and the experience wall head on. You leave with a 20-company target list and five tailored applications already sent - Strong portfolio README: the decisions you made and why - Resume for QA roles: manual + automation framing (guided self-study, mentor-reviewed) - **LinkedIn** optimisation: headline, About, QA skills (guided self-study, mentor-reviewed) - Mock interview 1: manual testing (test case design, bug reporting, **Agile** QA) - Mock interview 2: automation (**Playwright**, **Selenium**, **POM**, **CI/CD**, flaky tests), including the two questions every Nepal QA candidate gets and most fumble: "walk me through how you would test this screen" and "you have two days and forty test cases, what do you do" - QA career roadmap: junior to senior, advanced QA / SDET track,

WEEK	FOCUS AREA	TOPICS
		remote opportunities - Demo day: **Playwright** framework, **CI** pipeline, full portfolio

Skills You Will Gain

CATEGORY	TECHNOLOGIES
Languages	TypeScript, JavaScript, SQL
Manual Testing	Test case design, exploratory testing, regression, cross-browser, mobile
Bug Management	Jira, defect lifecycle, severity/priority, root cause analysis
API Testing	Postman, REST, authentication flows, negative testing, chaining
Automation	Playwright (depth), TypeScript, Page Object Model, fixtures, Selenium WebDriver (second framework, own public repo, Java/Python reading literacy)
Test Management	TestRail / Zephyr / Qase runs and reports, Excel test-case format, Confluence summaries
Security Testing	OWASP Top 10 for testers: SQL injection, XSS, broken auth, IDOR, missing authorisation, data exposure, rate limits
Domain Testing	Payment and transaction scenarios: limits, OTP, idempotency, timeouts, refunds, reconciliation
CI/CD	GitHub Actions, Newman in the pipeline, quality gates, artefact uploads, PR blocking, Jenkins pipeline literacy
Performance	k6 and JMeter (hands-on load tests, results compared)
Database	SQL SELECT/INSERT/UPDATE/DELETE for test data setup and teardown validation
Test Design	Equivalence partitioning, boundary value analysis, decision tables, state transitions
Reporting	Pass rate, defect density, coverage trends, sprint QA dashboards

CATEGORY	TECHNOLOGIES
Tools	Git, GitHub, VS Code, Jira, TestRail / Zephyr / Qase, Confluence, Postman, Newman, Chrome DevTools, JMeter, Jenkins (awareness), Playwright MCP + CLI, promptfoo
AI-Assisted Testing (AI as your tool)	Playwright MCP + CLI agentic authoring with Copilot / Claude Code / Cursor, the Playwright Test Agents loop (planner, generator, healer) run and reviewed line by line, AI-drafted test cases and bug reports pruned by hand, self-healing locators used with review discipline
Testing AI Systems (AI as the thing under test)	Gold sets, semantic and constraint-based assertions, LLM-as-a-judge, RAG triad (faithfulness, answer relevance, context groundedness), hallucination, bias and prompt-injection testing, an eval suite gated in CI with promptfoo

Topic Depth and Awareness

FOCUS AREA	LEARNING FOCUS
Test case design (EP, BVA, decision tables, state transitions)	Depth
Manual QA execution (regression, smoke, exploratory)	Depth
Bug reporting and Jira defect lifecycle	Depth
Agile QA participation and sprint ceremonies	Depth
API testing with Postman	Depth
SQL for test data and teardown validation	Depth
TypeScript for test authoring	Depth
Playwright (locators, POM, fixtures, config)	Depth
GitHub Actions CI quality gates	Depth
QA testing fundamentals	Depth
k6 performance testing	Depth (one hands-on load test)

FOCUS AREA	LEARNING FOCUS
JMeter load testing	Hands-on (one test plan, compared against k6)
Selenium WebDriver	Depth (second framework, own public repo, POM and explicit waits)
Security testing for QA (OWASP Top 10 lens)	Hands-on (findings written up as defects)
Test management (TestRail / Zephyr / Qase)	Hands-on (one full run cycle with report)
Integration testing and UAT support	Hands-on
Payment and transaction domain scenarios	Hands-on
Jenkins	Awareness (read a pipeline, explain each stage)
Confluence	Awareness
Agentic test authoring (Playwright MCP + CLI, Copilot / Claude Code / Cursor)	Depth
Playwright Test Agents loop (planner, generator, healer)	Depth (run end to end on your capstone, then reviewed)
Testing AI features (gold sets, LLM-as-a-judge, RAG triad, promptfoo, DeepEval)	Depth (you ship an eval suite gated in CI)
AI-drafted test cases and bug reports, human-pruned	Hands-on
Self-healing locators	Hands-on (with review discipline)
Mobile testing (responsive, viewport)	Depth
Visual regression testing	Depth
Cross-browser testing	Depth
Cypress	Awareness
Bruno (free Postman alternative)	Awareness

FOCUS AREA	LEARNING FOCUS
Mobile automation (Appium)	Awareness

Project Pool

Each student picks **one** capstone in Week 7. Pool rotates per batch:

#	PROJECT	FOCUS
1	E-commerce full QA portfolio	Manual test plan + 60 test cases + Jira bug history + Playwright UI suite
2	API regression suite with SQL teardown	Postman collection + Playwright request() + SQL assertions + GitHub Actions
3	Visual regression kit for a marketing site	Playwright screenshots, CI diffs, cross-browser coverage
4	CI-integrated automation framework with reporting	GitHub Actions, sharding, retries, pass-rate metrics
5	Bug-hunting capstone on a deliberately broken app	Exploratory testing + structured bug reports + automation for regression
6	Digital wallet / payment flow QA pack (Nepal domain)	Transfer limits, OTP, double-submit idempotency, timeout mid-payment, refund and reversal, SQL reconciliation, plus the OWASP-for-QA pass
7	Selenium-first regression framework	Selenium POM, explicit waits, data-driven runs, CI, with a written comparison against the Playwright suite
8	AI feature QA pack: testing a chatbot or LLM feature	Gold set, promptfoo or DeepEval suite, LLM-as-a-judge rubric, RAG triad scoring, hallucination and prompt-injection cases, CI quality gate, before/after quality number
9	Agent-generated suite with a human review report	Playwright Test Agents planner to generator to healer over a real app, plus a written audit of every generated test: kept, rewritten or deleted, with reasoning

Career Paths & Trajectory

ROLE	TIMELINE	FOCUS
Junior QA Engineer / Software Tester	0–2 yrs	Write test cases, execute regression suites, file bugs, participate in Agile sprints
Junior Test Automation Engineer	0–2 yrs	Write Playwright suites, maintain CI pipelines, automate critical flows
QA Automation Engineer / SDET I	2–4 yrs	Own feature automation, API coverage, test data, release quality checks
AI Quality / LLM Test Engineer	1–4 yrs	Own eval suites for AI features, gold sets and judge rubrics, hallucination and safety testing, quality gates on model and prompt changes. Newest lane, thinnest candidate pool
Senior QA Engineer / Test Lead	4–7 yrs	Test strategy, framework architecture, quality metrics, team mentoring
Quality Engineering Manager	7+ yrs	QA org leadership, release confidence, quality policy across products

FAQ

What does Software Quality Assurance cost and how do I pay?

NPR 28,200. Pay by cash at our Old Baneshwor campus, bank transfer, or Fonepay. Email info@saarathiacademy.com.np for the payment plan.

Does this course cover both manual and automation testing?

Yes, both properly, not as an afterthought on either side. The first three weeks build the manual QA foundation: SDLC and testing theory, test case design, integration and UAT work, manual platform testing, payment and transaction scenarios, and full bug lifecycle management with Jira, a test-management tool and Agile QA, a genuine third of the course before any automation. Week 4 then adds API testing with Postman, SQL database verification and security testing through a tester's lens. Weeks 5 to 7 teach Playwright automation with TypeScript, Page Object Model, cross-browser testing, a second Selenium framework, and CI/CD with GitHub Actions.

What do QA engineers actually earn in Nepal?

We publish the bands rather than promise a number. A QA internship or trainee stipend runs roughly NPR 10,000 to 25,000 a month. Fresher and junior roles sit around NPR 25,000 to 50,000. Mid-level engineers with a couple of years and real automation on their profile are in the NPR 40,000 to 75,000 range, and senior automation engineers earn well above that. Remote and international contract work pays on a different scale entirely and is a realistic target once you have shipped work you can show. What moves you up the band is not the certificate, it is the portfolio, the automation, and the domain you can speak to. We don't promise a salary.

Does QA require coding? Can I do this without a programming background?

Manual QA does not require coding, and the first four weeks of this course involve none. But the honest answer for 2026 is that manual-only limits how far you go and what you earn, which is why automation is a built-in module here rather than a separate course. You start programming in Week 5, from zero, with a primer in Week 4 self-study. Every student who has finished the manual weeks has been able to write Playwright tests; it is a smaller step than it looks from the outside.

Job ads say 1 to 2 years of experience. I have none. Is this course pointless?

No, but you should know exactly what you are up against, so we say it plainly. Nepal's entry-level QA posts routinely ask for 6 months to a year, and some posts titled "Junior QA Engineer" ask for 2 years. Many of those lines are soft filters. What gets you past them is evidence a hiring manager can check in ten minutes, so Week 8 is built to produce it: two public automation repos (Playwright and Selenium), a Jira defect history, a test-management run report, a security findings write-up, three real defects filed against real public software, and five tailored applications sent before you graduate. That is not the same as experience, and we won't pretend it is. It is the strongest substitute available to someone starting out, and it is what separates the candidates who get called from the ones who don't.

How much AI is actually in this course?

Two kinds, taught separately, because employers hire for them separately. **AI as your tool** runs from Week 1 to Week 8: AI-drafted test cases you prune by hand in Week 2, AI-cleaned bug reports and run summaries in Week 3, agentic authoring with Playwright MCP plus Copilot, Claude Code or Cursor from Week 5, self-healing locators in Week 6, and the full Playwright Test Agents loop (planner, generator, healer) run on your own capstone in Week 7. **AI as the thing under test** is a dedicated Week 7 block and a capstone option: gold sets, semantic and constraint-based assertions, LLM-as-a-judge scoring, the RAG triad, hallucination, bias and prompt-injection testing, and an eval suite gated in CI with promptfoo or DeepEval.

Why does testing AI features matter for a job in Nepal?

Because the products are already here and the testers are not. F1Soft is advertising an AI Automation Engineer, Fusemachines is an AI company, and effectively every wallet, bank and telco in Kathmandu is shipping a chatbot or an AI feature that somebody has to sign off. Almost nobody applying for junior QA roles here can write an eval suite, so a public one on your GitHub is a genuine differentiator rather than another line on a tool list.

The honest caveat on money: US postings requiring these skills show a median base of $119,300 against $80,000 for postings without them, but that sample is small (79 postings) and the gap mostly reflects scarcity of people who can do the work. Treat it as evidence the skill is rare, not as a number to quote in a Nepali salary negotiation.

Will AI take QA jobs? Is QA still a good career in 2026?

AI has changed what a QA engineer does, not whether one is needed. It writes test scaffolding fast and it heals brittle locators, which removes a chunk of the typing. It does not decide what is worth testing, judge whether a result is actually correct, weigh a release risk, or argue a severity call with a developer. Those are the parts employers hire for, and they are the parts this course drills hardest. The practical read: a tester who only executes scripts someone else wrote is exposed; a tester who designs tests, automates them, and reviews AI output is in a stronger position than they were three years ago. We teach the AI tooling directly (Copilot, Playwright Test Agents, MCP self-healing locators) so you use it rather than compete with it.

Is this ISTQB certified? Should I sit the ISTQB Foundation exam?

The course is not an ISTQB exam centre and we don't sell the exam. What we do is map the weekly QA fundamentals practice to the ISTQB CTFL syllabus chapters, so the same study builds toward Foundation Level if you decide to sit it. Whether it is worth the roughly $150 to $200 registration depends on where you are aiming: Nepal's outsourcing firms building for international clients recognise it and it can push a fresher to the upper end of the entry band, while smaller product teams and startups care far more about your GitHub. Most students are better served spending the first months on the portfolio and adding ISTQB later if the roles they want keep asking for it.

Do I need a Bachelor's degree?

Most Nepal QA postings ask for one in Computer Science or IT, and some employers hold that line. Plenty do not, especially product startups and remote clients, where the portfolio and a technical interview decide it. This course is open to you either way, and the career module covers which employers filter on the degree and which don't, so you spend your applications where they can land.

Why Selenium if Playwright is better?

Because you are job hunting in Nepal, not on the internet. Playwright is the better tool to learn on and it stays primary here. But Selenium is the name that appears first in Nepal job ads and recruiter shortlists, and Kathmandu hiring managers open your GitHub specifically looking for a Selenium Page Object Model framework. So you build one, as a second public repo, and you learn to read Selenium in Java and Python because that is the code in a Nepal technical interview. Learning Playwright first makes the Selenium framework take days instead of weeks.

Do I need prior experience?

No prior programming experience is required. TypeScript and JavaScript are introduced from zero in Week 5. You will learn everything needed to write Playwright tests before writing the first line of automation code. The Saarathi Gate Assessment runs before Day 1 to map your starting point, diagnostic, not pass-or-fail.

How long is the course?

8 weeks. A capstone project runs through the automation phase, with demo day and mock interviews in the final week.

What tools will I use?

Jira, TestRail / Zephyr / Qase, Postman, Newman, SQL, Playwright (with MCP and CLI), Selenium WebDriver, TypeScript, GitHub Actions, Chrome DevTools, k6 and JMeter, plus the AI stack: **Copilot / Claude Code / Cursor** for authoring, **Playwright Test Agents**, and **promptfoo** for testing AI features. Awareness of Confluence, Bruno and Jenkins because they appear in Nepal job descriptions. Every module ships a public, deployed artefact for your portfolio, no toy projects.

What roles does this prepare me for?

Junior QA Engineer, Junior Test Automation Engineer, Software Tester, QA Analyst, and increasingly AI Quality / LLM Test Engineer, the newest lane and the one with the thinnest candidate pool. The course is designed so you are competitive for both manual and automation QA roles at entry level. SDET and automation-engineer titles usually come after 2+ years of experience; this course gives you the foundation to grow into them.

How does the batch and mentorship work?

Maximum 10 students per batch, every student gets direct mentor time and project review. Hybrid by default, in-person at Old Baneshwor with online recordings and office hours. Sunday Open Classroom is free for enrolled students: pair-test, get unstuck, or just focus.

Enrollment

- **Next Batch:** see the live course page
- **Location:** Old Baneshwor Chowk, Kathmandu, Nepal
- **Mode:** Online + Offline
- **Contact:** 9744442469, 9761095364
- Limited to 10 seats per batch

Software Quality Assurance — full details, fees and next batch dates: saarathiacademy.com.np/software-quality-assurance-course-in-nepal

Version 1.0 · Updated July 2, 2026