Saarathi Academy

TR-05 Course Brief

Version 1.0.0 · Kartik 2083

Software Quality Assurance

Duration	Live classes	Schedule	Batch size
8 weeks	**80 hours**	**Mon to Fri, 2 hrs/day**	**10 students max**

Tools you will use: Jira + TestRail · Postman + SQL · Playwright + TypeScript · Selenium WebDriver · GitHub Actions + k6 · promptfoo AI evals

10 students a batch. Exceptional mentors who guide you personally, and a **syllabus built from real job posts**, cuz we never skip the "hard parts" 😉

The class

10 per batch

Ten students. Never more.

A hard cap, not a target. Your mentor knows your name and your work.

The mentor

4–5 yrs in the job

Someone who does the job.

Working professionals, not full-time teachers. They read your work line by line.

The syllabus

1/3 is fundamentals

The hard parts, not just the basics.

Teaches everything the job needs and never skips the hard parts most courses skip, built from real job descriptions. Check the syllabus before you enroll.

Built into every course

- **Gate assessment** — A short test before Day 1 finds your strengths and pace, then sets your plan.
- **Revision every 5th class** — Every fifth class goes back over the week, so nothing piles up.
- **Certification exam** — Pass the exam at the end of the course to earn your certificate.
- **A real, live project** — Every course ends with a deployed project you can show employers.
- **1:1 mock interview** — A real interview practice with your mentor in the final week.
- **CV and LinkedIn help** — We help you rewrite your CV and LinkedIn before you apply.

saarathiacademy.com.np

Old Baneshwor Chowk, Kathmandu

+977-9744442469 +977-9761095364 +977-9768618434

Software Quality Assurance

Why This Course

You train for the junior QA engineer job that employers hire for. QA (quality assurance) means checking that software works before users get it. You learn to write test cases and bug reports, and to check APIs and databases. You also automate tests with Playwright and run them on every code change (CI). Automation is part of the course, not an extra track.

- **You start with manual testing.** Weeks 1 to 3 build your testing basics. In Week 4 you add Postman for API testing and SQL for checking the database.
- **Then you automate your tests.** JavaScript self-study starts in Week 2. Weeks 5 to 7 cover Playwright with TypeScript, Selenium in TypeScript, a first Cypress test, GitHub Actions and JMeter.
- **You test on a real phone.** You test on your own Android phone, and your mentor demos Appium.
- **You use AI to write tests, and see how AI features are tested.** From Week 6 an AI tool writes tests you review. In Week 7 you run one small eval of an AI feature with promptfoo.
- **You learn what Nepal job ads ask for.** This includes Selenium, Postman, a test-management tool, payment testing and the OWASP Top 10 security risks.
- **You leave with proof of your work.** In Week 8 you put your projects online so employers can see them. You also show your bug history and send real job applications.

Pay depends on your skills, portfolio and clients. We do not promise a salary.

Who Is This Course For

- **Complete beginners** who can use a computer and want to start a full career in QA.
- **Manual testers** who want to move into automation and add Playwright, API testing and CI/CD to their CV.
- **Fresh graduates** aiming for junior QA Engineer, Software Tester or QA Analyst jobs.
- **Career switchers** from non-technical jobs, such as business analysts, project coordinators and operations staff.
- **Developers** who want to see quality from a tester's side and write better-tested software.
- **Students** who want a QA course in Kathmandu that leaves them with a real portfolio.

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
Course Fee	NPR 29,200
Level	Beginner → Junior QA Engineer (manual + automation capable)
Outcome	Junior QA Engineer with Playwright **and** Selenium automation skills, plus API, SQL, security, mobile and payment-domain testing; automation is a built-in module of this QA course, not a separate track

Prerequisites

- You do not need any programming background. JavaScript self-study starts in Week 2 (30 minutes a day), a TypeScript primer is checked on Week 4 Day 5, and Week 5 opens with a live JavaScript clinic.
- You need your own laptop with at least 8GB RAM. The academy does not provide devices.
- An Android phone helps for the real-phone hour in Week 7. If you do not have one, you pair with a classmate.
- You should be comfortable with a laptop and a browser. You should also be willing to read an error message and search for the answer.
- You take the Saarathi Gate Assessment before Day 1. It only shows where you are starting, with no pass or fail.
- You install nothing beforehand. We set up VS Code, Node.js, GitHub and Jira in the first class, and Java (JDK 17 or later) for JMeter in Week 7.

Costs. You do not need to pay for anything. Free plans cover the course: Jira and Confluence, Qase, Postman (single user) or Bruno, GitHub, GitHub Copilot Free (or free with GitHub Education), the Gemini API free tier, and the open-source Playwright, Selenium, Cypress, JMeter

and promptfoo. Paid and optional: TestRail (about USD 37 per user a month after a 30-day trial), Zephyr and Xray, Claude Code and Cursor Pro. The ISTQB exam is optional; prices vary by provider, roughly USD 75 to 230.

Assessments at a glance

- **Gate Assessment.** You take it before Day 1. It is not pass or fail. It shows us how you learn and how fast.
- **Certification Exam.** You take it online at home on your own laptop in the final week, outside class hours. It lasts 2 hours and has around 30 questions of different types, including code and short real-world cases.
- **Pass mark.** You need 70 percent overall and at least 50 percent in every area. You must pass to get the Certificate of Completion.
- **Attendance.** You need at least 70% attendance to sit the exam.
- **Camera.** Your camera stays on for the full 2 hours.
- **Three strikes.** Three strikes end the exam early: leaving the exam tab, someone else in the camera, or a phone in view.
- **Recorded, not a strike.** Leaving fullscreen, pasting text and looking away are recorded for the Saarathi team, but they do not end the exam.
- **Checked by a person.** A person on the Saarathi team checks the camera pictures before a pass counts.
- **Answer order.** You can answer in any order, but every question needs an answer before you submit.
- **Recommendation letters.** We may give one to learners who pass, but it is never guaranteed.

Projects

Your mentor gives you 2 projects during the course, and they need to be approved before you get your certificate. The Career Launchpad stays locked until then.

Your Learning Week

DAY	ACTIVITY
Mon–Fri	2-hour live class with your mentor, with hands-on practice
Mon–Fri	2 hours of self-study at home: tasks, practice and fixing errors
Sunday	Open classroom. Come in to build in a quiet space, or rest.

DAY	ACTIVITY
Day 5 (every week)	In the last hour of the 2-hour class, you revise the week and clear your doubts. Day 5 has fewer new topics, and hands-on tasks continue in your self-study time.

Each revision hour adds 10 practice questions on QA basics, based on the ISTQB CTFL chapters (a well-known testing certificate). You also do **3 spoken interview drills**. Interview practice starts in Week 1.

Most real progress happens between classes, when you build, break, test and submit again.

Week-by-Week Curriculum

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

Three weeks of testing basics before any API or automation work. Week 1 is orientation, SDLC and testing theory; Week 2 is test design and manual testing; Week 3 is the bug lifecycle, Jira, Qase 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 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

WEEK	FOCUS AREA	TOPICS
		(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** (20 minutes): most QA teams now try AI tools and few use them at scale (World Quality Report 2025-26). AI saves typing: it scaffolds a suite, drafts a test matrix and repairs a broken locator. It does not decide what is worth testing, judge whether a result is right, weigh release risk, or defend a severity call to a developer. Also a first look at the new thing you will be asked to test: products with chatbots that never answer the same way twice, covered in Week 7

| **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 (the test-management tool comes hands-on in Week 3)
- Cross-browser testing: same behaviour in **Chrome**, **Firefox** and **Edge** (Safari's engine is checked through Playwright later)
- Mobile testing in emulation: **Android** vs **iOS**, viewport testing (screen sizes), responsive design (layouts that adapt); real-phone testing comes in Week 7
- 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
- **JavaScript self-study starts:** 30 minutes a day at home, no class hours |

| **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
- **AI in the paperwork half of QA**, one drafting drill: AI turns a messy repro into a clean bug report, and you see where else it helps (summarising a failed run into a sprint update, grouping many failures by cause, drafting cases from acceptance criteria). The rule that survives a real team: AI drafts, a human checks every step against the actual build, and the human's name is on it
- **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 **Qase** (free plan): suites, runs, pass/fail/blocked results, a defect linked from a failed step, run report exported. Your mentor shows **TestRail** on its trial; **Zephyr** and **Xray** at awareness. Then the same suite kept in the **Excel** format many Kathmandu teams mail around
- Root cause analysis basics: separating symptoms from causes, digging to the real cause
- 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): collections (saved requests), environments, variables. The free plan is single-user, so you export the collection into your Git repo for mentor review (**Bruno** works the same way) - 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, plus a one-line **JSON Schema** check, then running a full test sequence end-to-end in one click (Postman's AI, **Postbot** / Agent Mode on limited free credits, can draft assertions, reviewed like any AI output) - **SQL** for testers, asking the database questions: SELECT, WHERE, JOIN, COUNT, SUM, GROUP BY, with extra guided practice - 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 (the newer **Postman CLI** also works) - **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, checked on Day 5: variables,**

WEEK	FOCUS AREA	TOPICS
		functions, arrays, objects, async/await mini-exercises, built on your JavaScript self-study since Week 2

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

WEEK	FOCUS AREA	TOPICS
Week 5	TypeScript Basics & Playwright Foundations	- **Live JavaScript clinic** (1 hour): the JavaScript you practised since Week 2, worked through with your mentor - 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 - **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) - Running the first end-to-end test suite green (every test passing) - No AI agents this week: you learn the code first - **Graded coding strand 1 of 3 (self-study): 2-3 easy string/array problems** - **Public defects strand (self-study, Weeks 5 to 7):** file three genuine defects against real public software that invites reports
Week 6	AI-Assisted Test Writing, Page Object Model, Fixtures,	- **AI-assisted test writing (Day 1)**: the **Playwright MCP** server (Playwright 1.62 or later) lets **GitHub Copilot Free** (or Copilot through GitHub Education; Claude Code and Cursor optional, paid) open a real browser and write a test against what is on the page. Then you read every line, check the locator survives a re-render, confirm the assertion would fail if the feature broke, and delete the

WEEK	FOCUS AREA	TOPICS
	Selenium & Cypress	noise. **You own every test with your name on the commit** - **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 and auth state reuse: log in once with `globalSetup`, save storage state, reuse across all tests - 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 workers and sharding at awareness) - Network interception with `page.route()`: mocked responses to test error states without the real backend; visual regression with `toHaveScreenshot` as a short demo - **Selenium WebDriver in TypeScript**, the automation tool Nepal posts name most, so you build with it and do not just hear about it: **Selenium Manager** handles the browser driver, locator strategies, **explicit waits** with `driver.wait(until.elementLocated(...))` and why implicit waits and `driver.sleep` cause flaky tests, a POM layer, **Mocha** as the runner, rebuilding the login and checkout flows - A one-page **reading card** with the same login test in **Python** (pytest) and **Java** (TestNG, `WebDriverWait`, `ExpectedConditions`, why `Thread.sleep` is flaky), plus a short **REST Assured** API test, so you can read the code an interviewer shows you - **A first Cypress test**: one Playwright test ported to **Cypress** in the same repo (30 minutes) - **Graded coding strand 2 of 3 (self-study): 2-3 easy array/object problems**
Week 7	API Automation, CI/CD, Load Testing & Real-Phone Testing	- **Playwright** `request()` for API test automation, no browser needed - Combining UI and API tests in one **Playwright** project - **GitHub Actions**: `.github/workflows/tests.yml` running the 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** - Uploading failure artefacts: traces, screenshots, HTML report - Flaky test investigation: root-cause analysis vs blind retries

WEEK	FOCUS AREA	TOPICS
		- Required checks: blocking merges on a red pipeline - **Apache JMeter** load testing, the load tool Nepal posts name most: thread groups, ramp-up, samplers, listeners, reading p95 latency, throughput and error rate; **k6** as a 20-minute demo - **Mobile testing on a real phone** (1 hour): USB debugging on your Android phone, Chrome remote debugging from your laptop, installing a test APK, and testing interruptions (incoming call, network drop, rotation, app sent to background); your mentor then demos **Appium** - **AI in the loop**: a 30-minute demo of the **Playwright Test Agents** (planner, generator, healer), then one local **promptfoo** eval: a small **gold set** (inputs with the answer you expect) run against the **Gemini API** free tier with test data only, and why exact-match checks like `toHaveText` fail on AI output, since a chatbot never returns the same string twice. promptfoo is open source and owned by OpenAI since 2026. LLM-as-a-judge (a second model grades the answer against a rubric you wrote), the RAG triad (is the answer supported, relevant and based on the right documents), the failures that get a chatbot pulled (made-up numbers, prompt injection, leaked instructions), **DeepEval** and gating evals in CI at awareness - **Graded coding strand 3 of 3 (self-study): 2-3 mixed function/array problems**

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

WEEK	FOCUS AREA	TOPICS
Week 8	Project Showcase, 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 cases, **Jira** bug history, **Qase** run report, **Postman** collection, **SQL** queries, security findings, real-phone bug log, **Playwright** suite, **Selenium** suite and your promptfoo eval, with a short README for each: the decisions you made and why, and for the eval what it measures and what it caught - Your **three public defects** from the Weeks 5 to 7 self-study strand checked and linked, because filed public defects can be checked in a way a certificate cannot. Responsible disclosure rules come first and are not optional: report through the channel the owner provides, and never test a system you were not invited to test - **The Nepal QA market and career roadmap**: where QA jobs sit

WEEK	FOCUS AREA	TOPICS
		(fintech, health and insurance outsourcing, telecom, product companies, Kathmandu startups), the experience wall head on (few posts take freshers, so internships and trainee posts are the realistic first target), and the path from junior to senior, the **SDET** track, the optional **ISTQB CTFL** exam and remote opportunities - **Day 3 · CV, job applications and LinkedIn:** an ATS-friendly resume for QA roles with manual + automation framing, built around your projects with real numbers from them. Concrete results over adjectives, and which project leads for a manual role or an automation role. Where QA jobs are posted (company career pages first, then merojob, KumariJob, JobAxle, LinkedIn Nepal), reading a job post, tailoring the CV, a short cover note and an application tracker; you leave with a 20-company target list and five tailored applications already sent. **LinkedIn** headline, About, QA skills, your best project featured and a post about it - **Day 4 · Mock interview:** one day, everyone takes part. Each student gets a short round with the mentor (about 10 to 15 minutes): a timed testing task on a screen you have not seen, AI tools allowed; a few manual and automation questions, including the two every Nepal QA candidate gets: "walk me through how you would test this screen" and "you have two days and forty test cases, what do you do"; and a **bug-report defence**, where your mentor plays a developer disputing one of your bugs ("can't reproduce", "works on my machine") and you defend it with reproduction steps, evidence and severity reasoning. Completed, with written feedback and one thing to fix; no pass or fail - **Day 5 · Project showcase and presentation:** each student presents their projects (**Playwright** framework, **CI** pipeline, full portfolio) to the batch and mentor: the problem, a live demo, what was hard, what they would do next, then questions and feedback

Skills You Will Gain

CATEGORY	TECHNOLOGIES
Languages	TypeScript, JavaScript, SQL (Python and Java reading only)
Manual Testing	Test case design, exploratory testing, regression, cross-browser, mobile emulation and real-phone testing
Bug Management	Jira, defect lifecycle, severity/priority, root cause analysis

CATEGORY	TECHNOLOGIES
API Testing	Postman, REST, authentication flows, negative testing, chaining
Automation	Playwright (depth), TypeScript, Page Object Model, fixtures, Selenium WebDriver in TypeScript (with a Python and Java reading card), a first Cypress test
Test Management	Qase runs and reports (TestRail shown by the mentor), 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	JMeter (hands-on load test), k6 (demo)
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, escape rate
Tools	Git, GitHub, VS Code, Jira, Qase, Confluence, Postman or Bruno, Newman, Chrome DevTools, JMeter, Jenkins (awareness), Playwright MCP, GitHub Copilot, promptfoo
AI-Assisted Testing (AI as your tool)	Playwright MCP test writing with GitHub Copilot, reviewed line by line; AI-drafted test cases and bug reports pruned by hand; Playwright Test Agents (demo)
Testing AI Systems (AI as the thing under test)	One local promptfoo eval with a gold set; LLM-as-a-judge, the RAG triad and CI-gated evals at awareness

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
JMeter load testing	Hands-on (one test plan)
k6 performance testing	Awareness (20-minute demo)
Selenium WebDriver (TypeScript)	Hands-on (POM and explicit waits; Python and Java reading card)
Security testing for QA (OWASP Top 10 lens)	Hands-on (findings written up as defects)
Test management (Qase; TestRail shown by the mentor)	Hands-on (one full run cycle with report)
Integration testing and UAT support	Hands-on
Payment and transaction domain scenarios	Hands-on

FOCUS AREA	LEARNING FOCUS
Jenkins	Awareness (read a pipeline, explain each stage)
Confluence	Awareness
AI-assisted test writing (Playwright MCP + GitHub Copilot)	Hands-on
Playwright Test Agents (planner, generator, healer)	Awareness (30-minute demo)
Testing AI features (promptfoo)	Hands-on (one local eval with a gold set)
LLM-as-a-judge, RAG triad, DeepEval, evals gated in CI	Awareness
AI-drafted test cases and bug reports, human-pruned	Hands-on
Mobile testing (emulation and a real Android phone)	Hands-on
Visual regression testing	Awareness (demo)
Cross-browser testing	Depth
Cypress	Hands-on (one test)
Bruno (free Postman alternative)	Awareness
Mobile automation (Appium)	Awareness (mentor demo)

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	0–2 yrs	Write Playwright suites, maintain CI pipelines, automate critical flows

ROLE	TIMELINE	FOCUS
Engineer		
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. A newer role; this course gives you the first step only
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 29,200. You can pay in cash at our Old Baneshwor campus, by bank transfer, or with Fonepay. Email info@saarathiacademy.com.np to ask about a payment plan.

Does this course cover both manual and automation testing?

Yes, and both are taught properly.

- **Weeks 1 to 3** cover testing theory, writing test cases, user acceptance testing (UAT) and payment scenarios. You also track bugs in Jira and a test-management tool.
- **Week 4** covers API testing with Postman, checking data with SQL, and security testing.
- **Weeks 5 to 7** cover Playwright with TypeScript, the Page Object Model, Selenium, a first Cypress test, GitHub Actions CI/CD, JMeter and testing on a real phone.

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

Manual QA needs no coding, and Weeks 1 to 4 have none in class. But manual testing alone limits how far you can grow, so automation is part of the course. You start JavaScript at home in Week 2 (30 minutes a day), a primer is checked in Week 4, and Week 5 opens with a live JavaScript clinic.

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

No, but be realistic. Few Nepal QA posts take freshers, so internships and trainee posts are the best first target. By Week 8 you have proof that a hiring manager can check in ten minutes:

- Public automation work in Playwright and Selenium.
- A history of bugs you filed in Jira, and a report from a test-management tool.
- A write-up of security problems you found, and three real bugs filed against public software.
- A list of 20 target companies, and five tailored job applications sent before you finish.

This is not work experience, and we will not pretend it is. But it is the strongest proof you can show when starting out.

How much AI is actually in this course?

We teach two kinds, and we teach them separately:

- **AI as your tool.** AI drafts test cases and cleans up bug reports. From Week 6 you write tests with Playwright MCP and GitHub Copilot, and review every line. You also see a demo of the Playwright Test Agents.
- **AI as the thing you test (part of Week 7).** You run one small eval of an AI feature with promptfoo and learn why normal checks fail on AI answers. The deeper methods are covered at awareness level.

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

AI changed what QA engineers do, not whether companies need them. AI writes basic test code and can repair broken locators (the way a test finds a button on the page). It does not decide what to test or judge whether a result is right. We teach the AI tools directly, so you use them instead of competing with them.

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

We are not an ISTQB exam centre, and we do not sell the exam. Our weekly practice follows the ISTQB CTFL chapters, so you can sit the Foundation Level exam later if you want. It is optional.

- There is no ISTQB board in Nepal. You sit it online with an approved provider.
- Prices vary by provider, roughly USD 75 to 230.
- It has 40 questions in 60 minutes (75 if English is not your first language), and you need 26 correct to pass.
- Outsourcing companies value it. Startups care more about your GitHub.
- Most students do better if they build the portfolio first.

Do I need a Bachelor's degree?

Most QA job ads in Nepal ask for a CS or IT degree, and some employers insist on it. Startups and remote clients often do not. The course is open to you either way, and the career module shows which employers filter by degree.

Why Selenium if Playwright is better?

Playwright is our main tool because it is the better tool to learn on. But Selenium is the automation tool Nepal job ads name most. So you also build a Selenium Page Object Model framework in the same TypeScript, and a reading card shows you the same test in Java and Python.

Do I need prior experience?

No programming experience is needed. JavaScript self-study starts in Week 2 and TypeScript is taught in class from Week 5. The Gate Assessment before Day 1 shows where you are starting, and there is no pass or fail.

How long is the course?

8 weeks. The last three days of the final week are your CV and job applications day, a mock interview and the project showcase.

What tools will I use?

You use **Jira, Qase, Postman, Newman, SQL, Playwright (with MCP), Selenium WebDriver, Cypress, TypeScript, GitHub Actions, Chrome DevTools, JMeter, your Android phone, GitHub Copilot and promptfoo**. You see TestRail, Appium, k6, the Playwright Test Agents, Confluence, Bruno and Jenkins as demos or awareness. Every tool you need has a free plan.

What roles does this prepare me for?

Junior QA Engineer, Junior Test Automation Engineer, Software Tester and QA Analyst. AI Quality roles are newer, and this course gives you a first step toward them. SDET titles (testers who write a lot of code) usually come after 2 or more years. This course is the foundation for them.

How does the batch and mentorship work?

Each batch has at most 10 students, and each one gets direct mentor time and project reviews. Classes are in person at Old Baneshwor, or live online. Recordings and office hours are online. The Sunday Open Classroom is free for enrolled students.

Enrollment

- **Next Batch:** see the live course page
- **Location:** Old Baneshwor Chowk, Kathmandu, Nepal
- **Mode:** Online + Offline
- **Contact:** 9744442469, 9761095364, 9768618434
- Each batch has only 10 seats.

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

Version 1.0.0 · Updated 14 Kartik 2083