

Software Quality Assurance

Duration

8 weeks

Live classes

80 hours

Schedule

Mon to Fri, 2 hrs/day

Batch size

10 students max

Tools you will use

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

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.

A real, live project

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

Revision every 5th class

Every fifth class goes back over the week, so nothing piles up.

1:1 mock interview

A real interview practice with your mentor in the final week.

Certification exam

Pass the exam at the end of the course to earn your certificate.

CV and LinkedIn help

We help you rewrite your CV and LinkedIn before you apply.

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.** Weeks 5 to 7 cover Playwright with TypeScript, Selenium and GitHub Actions.
- **You use AI in two ways.** It helps you write tests, and you also learn to test apps that have AI features.
- **You learn what Nepal job ads ask for.** This includes Selenium, TestRail or Zephyr, 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

PARAMETER	DETAILS
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 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. We teach TypeScript from zero in Week 5, with a short primer to study at home in Week 4.
- You need a laptop with at least 8GB RAM.
- 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.

Assessments at a glance

- **Gate Assessment:** You take it before Day 1. It shows your learning style and pace, and there is no pass or fail.
- **Certification Exam:** It comes at the end of the course. It lasts 2 hours and has about 30 questions on this course. Question types include single choice, multiple choice, true or false, numbers, ordering, matching, fill in the blank, short or long answers, code where needed, and short case studies.
- **Pass mark:** You need 70 percent. You must pass to get the Saarathi Certificate of Completion.
- **Exam rules:** The exam is watched (proctored) and runs in fullscreen. Switching tabs or apps and pasting are recorded, and three strikes end the exam.

- **Question order:** You can answer in any order and go back to any question. Every question needs an answer before you submit.
- **Recommendation letters:** We give these at our choice, only to learners who pass. They are never guaranteed.

Your Learning Week

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

Your mentor gives you 2 projects during the course. They need to be approved before you get your certificate and before the Career Launchpad opens.

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)

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	- What a QA engineer does and why the role matters in a team - Telling apart the three words job posts mix up: QA (prevents

WEEK	FOCUS AREA	TOPICS
	Fundamentals	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 (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

WEEK	FOCUS AREA	TOPICS
		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 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), <code>tsconfig.json</code> basics - Playwright installation and first test project - Locator strategy, what to find on the page: <code>getByRole</code> , <code>getByText</code> , <code>getByTestId</code> , CSS selectors as a last resort - Actions, driving the browser: click, fill, check, select, keyboard input - Assertions, automatic pass/fail checks: <code>toBeVisible</code> , <code>toHaveText</code> , <code>toHaveURL</code> , <code>toBeEnabled</code> , <code>toHaveCount</code> - 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 (<code>npx playwright mcp</code>), 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 (<code>--ui</code> watch runner) and trace viewer:

WEEK	FOCUS AREA	TOPICS
		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) - 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	FOCUS AREA	TOPICS
Week 7	API Automation, CI/CD Quality Gates & Performance Testing	- Playwright <code>request()</code> 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: <code>.github/workflows/tests.yml</code> running the suite on pull requests, plus a <code>newman run</code> step so the Week 4 Postman collection gates merges too - Jenkins at read-and-explain depth, a real <code>Jenkinsfile</code> 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 <code>specs/</code>, generator turns it into real test files in <code>tests/</code> from a <code>seed.spec.ts</code> 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 <code>toHaveText</code> 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

WEEK	FOCUS AREA	TOPICS
		- 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 - A separate timed live testing task, AI tools allowed: given a feature you have not seen before, design and run test cases against it under a clock, screen shared, narrating every decision out loud, the same way it works on the job - 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-

WEEK	FOCUS AREA	TOPICS
		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" - Bug-report defence: your mentor plays a developer disputing one of your filed bugs, and you defend the call with reproduction steps, evidence and severity reasoning - QA career roadmap: junior to senior, advanced QA / SDET track, remote opportunities - Demo day and portfolio walkthrough: Playwright framework, CI pipeline, full portfolio, delivered to a panel the way a hiring manager reviews one

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

CATEGORY	TECHNOLOGIES
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
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

FOCUS AREA	LEARNING FOCUS
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)
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)

FOCUS AREA	LEARNING FOCUS
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
Mobile automation (Appium)	Awareness

Project Pool

Each student picks **one** capstone (final project) in Week 7. The list changes from batch to 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

#	PROJECT	FOCUS
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. This is the newest QA job, and very few people can do it yet
Senior QA Engineer / Test	4–7 yrs	Test strategy, framework architecture, quality metrics, team mentoring

ROLE	TIMELINE	FOCUS
Lead		
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, a Selenium framework and GitHub Actions CI/CD.

What do QA engineers actually earn in Nepal?

Pay rises from an intern stipend to junior, mid and senior levels. Remote jobs pay on a different scale. Your portfolio, automation skill and industry knowledge move you up, not the certificate. We do not promise a salary.

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

Manual QA needs no coding, and Weeks 1 to 4 have none. But manual testing alone limits how far you can grow, so automation is part of the course. You start coding from zero in Week 5, with a short primer to study at home in Week 4.

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

No. Many entry-level ads ask for 6 months to 2 years, but employers often do not hold to it strictly. In Week 8 you build proof that a hiring manager can check in ten minutes:

- Two public automation projects, one in Playwright and one in 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 (Weeks 1 to 8).** AI drafts test cases and cleans up bug reports. You use Playwright MCP with Copilot, Claude Code or Cursor, self-healing locators (tests that fix themselves when a page changes) and the Playwright Test Agents loop.
- **AI as the thing you test (a Week 7 block and a capstone option).** You learn gold sets, LLM-as-a-judge and the RAG triad. You check AI features for made-up answers, bias and prompt injection, and run an eval suite in CI with promptfoo or DeepEval.

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

The AI products are already here, but the testers are not. FISOFT is hiring an AI Automation Engineer, and most wallets, banks and telecom companies are launching chatbots. Few junior QA applicants can write an eval suite (a set of tests for an AI feature), so a public one stands out.

US job ads pay more for this skill, but there are only a few of them. Treat this as a sign the skill is rare, not as a Nepali salary figure.

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 repairs 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.

- The exam costs roughly \$150 to \$200.
- 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 Nepal job ads name Selenium first. So you also build a Selenium Page Object Model project as a second public repo, and learn to read Selenium code in Java and Python.

Do I need prior experience?

No programming experience is needed. TypeScript and JavaScript start from zero in 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. You build a capstone project during the automation weeks. Demo day and mock interviews happen in the final week.

What tools will I use?

You use **Jira, TestRail / Zephyr / Qase, Postman, Newman, SQL, Playwright (with MCP and CLI), Selenium WebDriver, TypeScript, GitHub Actions, Chrome DevTools, k6 and JMeter**. You also use **Copilot / Claude Code / Cursor, Playwright Test Agents** and **promptfoo**. You see Confluence, Bruno and Jenkins too, because Nepal job ads name them. Every module ends with a portfolio piece that is public and live online.

What roles does this prepare me for?

Junior QA Engineer, Junior Test Automation Engineer, Software Tester, QA Analyst, and AI Quality / LLM Test Engineer. The AI Quality job is the newest kind, with the fewest candidates. 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. 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
 - 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 2.0 · Updated 11 Ashwin 2083

