INSIDE STLC

BECOME A SOFTWARE TESTER ROADMAP 2026

The exact order I'd learn QA from zero - without wasting a year on the wrong things.

FUNDAMENTALS | TOOLS | PROOF OF WORK | AUTOMATION | JOB HUNT

Includes a 30-day starter plan, free tools to install, portfolio checklist and job-readiness guide.

By David Oxley
Founder, Inside STLC
25+ years in Software Testing & QA Leadership

Start in the Right Order

Two people can spend the same 12 months learning software testing and finish in completely different positions. The difference is often not talent or certification - it is sequence.

The entry point into QA has moved. Employers still need people who can investigate problems, communicate clearly and learn quickly, but today's junior tester is also expected to understand modern delivery, work with common tools and show evidence that they can actually test.

Do not build a 2019 skill list for a 2026 job market.

What you do NOT need before you begin

- A computer science degree.
- A coding background.
- Ten different testing tools.
- Months of certification study before you have tested anything.
- Advanced automation before you understand testing fundamentals.

What employers need to see

- You can think critically about a product.
- You can describe a problem clearly.
- You understand the basic language of software delivery.
- You can use common testing tools rather than merely list them.
- You can show proof that you have practised testing.

The 2026 Roadmap at a Glance

STEP	LEARN	PROVE
1	Testing fundamentals	Write useful tests and bug reports
2	Modern delivery language	Understand Agile, stories, sprints and CI/CD
3	Core tools	Use Jira, Postman and browser developer tools
4	Proof of work	Create a small QA portfolio
5	Automation basics	Read and build simple Playwright tests
6	Targeted applications	Apply with evidence, not a generic CV

Step 1 - Learn the Craft Before the Tools

Give this a focused weekend, not three months. You are building the mental model that every tool and technique will sit on.

- What software testing is - and what it is not.
- Test cases, test scenarios and expected results.
- Defects: severity, priority, reproducibility and evidence.
- Verification vs validation.
- Functional vs non-functional testing.
- Positive, negative and boundary thinking.
- Why testing is about risk and information, not proving software is perfect.

Mini-project: Pick a familiar website. Write 10 test scenarios and 3 high-quality bug reports or improvement observations.

Step 2 - Speak the Language of the Team

- Agile and iterative delivery.
- Sprint, backlog, refinement and retrospective.
- User story, acceptance criteria and requirement.
- Definition of Done.
- Basic SDLC and STLC concepts.
- What CI/CD means and why fast feedback matters.
- The roles of developer, product owner, business analyst and QA.

Step 3 - Learn a Small Core Toolset

Depth beats breadth. You are not trying to collect logos for your CV.

AREA	START HERE	WHAT TO PRACTISE
Test management / defects	Jira or similar	Stories, defects, workflow, evidence
API testing	Postman	GET/POST, status codes, JSON, auth basics
Browser investigation	Developer Tools	Network, console, elements, requests
Automation	Playwright	Simple readable tests when foundations are ready
Data	SQL basics	SELECT, WHERE, JOIN and validation thinking

Step 4 - Build Proof of Work

Reading is not experience. Your first portfolio does not need to be enormous; it needs to show how you think.

- A short test approach for one website or demo application.
- 15-25 well-written test scenarios or cases.
- 5 example defect reports with clear evidence.
- A Postman collection covering a small public/demo API.
- A simple SQL exercise or data-validation example.
- Later: a small Playwright suite with a handful of meaningful tests.
- A README explaining what you tested, the risks you identified and what you learned.

Step 5 - Add Automation at the Right Time

- Understand why repetitive, stable checks are candidates for automation.
- Learn to read a simple test script before worrying about framework architecture.
- Use Playwright as a modern starting point for browser automation.
- Learn basic TypeScript or JavaScript alongside it.
- Use AI to explain, scaffold and debug - but do not submit code you cannot explain.
- Keep exploratory and human judgement skills alongside automation.

The Three Mistakes That Waste a Year

MISTAKE	WHY IT HURTS	DO THIS INSTEAD
Starting with certification	Theory has little context before you have tested	Learn fundamentals, practise, then decide whether ISTQB s
No proof of work	Your CV makes claims but gives employers nothing to inspect	Build one small, polished QA portfolio
Applying too early and everywhere	Generic applications create rejection without useful feedback	Become technically job-ready, then target suitable roles care

Your First 30 Days

WEEK	FOCUS	OUTPUT
1	Testing fundamentals	10 scenarios + 3 example defects
2	Delivery language + tools	Use Jira-style workflow, DevTools and Postman
3	Proof of work	Package your testing into a simple portfolio
4	Job readiness	CV, LinkedIn, interview examples + first targeted applications

Consistency beats intensity. One useful piece of work every day is more valuable than a weekend of tutorials you never apply.

Free Starter Toolkit

PURPOSE	FREE / ACCESSIBLE STARTING POINT
Browser testing	Chrome, Edge or Firefox + built-in Developer Tools
API practice	Postman free tier + public/demo APIs
Automation	Playwright open-source framework
Code editor	Visual Studio Code
Version control	Git + a Git hosting account
SQL practice	SQLite or browser-based SQL practice environments
Documentation	Markdown / Google Docs / spreadsheets
Practice applications	Public demo sites designed for testing practice

Job-Ready Checklist

You do not need perfection. You need enough evidence to have a credible entry-level conversation.

[ ] I can explain software testing in my own words.

[ ] I can write clear test scenarios and defect reports.

[ ] I understand Agile delivery and common team terminology.

[ ] I can investigate a browser using Developer Tools.

[ ] I can send and validate basic API requests in Postman.

[ ] I understand basic SQL queries.

[ ] I have a small portfolio that shows real practice.

[ ] I can explain what I tested, why I tested it and what I found.

[ ] I understand the purpose of automation and can read or create a basic test.

[ ] I have prepared examples for common QA interview questions.

[ ] My CV focuses on evidence and transferable skills rather than a long tool list.

[ ] I am applying to roles that genuinely match my current level.

Your 12-Month Direction

The first 30 days get you moving. The next year should progressively turn fundamentals into employable depth.

MONTHS	PRIORITY
1-2	Fundamentals, delivery language, Jira/Postman/DevTools, first portfolio
3-4	SQL, stronger API testing, targeted applications, interview practice
5-6	Playwright + one language at beginner level; improve portfolio
7-9	Build realistic automation, CI/CD awareness, deeper exploratory and risk thinking
10-12	Develop domain knowledge, use AI productively, strengthen communication and target the next level

The principle to remember:

Technical skills get you noticed. Clear thinking, communication and evidence make you credible. Build the foundation first, then add tools in the order that supports the work.

Continue Your Journey

If you want a structured route beyond this free roadmap, Inside STLC Academy includes practical learning for software testing careers, QA interviews, AI, automation and Playwright.

INSIDE STLC ACADEMY
Structured learning for modern QA careers

www.insidestlcacademy.com

Free resources: **www.insidestlc.com/resources**

Inside STLC - Software Tester Roadmap 2026. Hiring requirements vary by employer, role and location. This roadmap is practical career guidance, not a guarantee of employment.