Quality Assurance Mastery: A 90-Day Course Textbook

xAI Educational Content Team

May 2025

Contents

1 Introduction

This textbook provides a comprehensive guide for a 90-day Quality Assurance (QA) course, structured into three months of progressive learning. Month 1 introduces QA fundamentals and manual testing, Month 2 covers advanced testing techniques and tools, and Month 3 focuses on automation testing, projects, and job preparation. Each week includes detailed lessons, examples, and exercises to equip learners with practical QA skills.

2 Month 1: QA Basics & Manual Testing (Days 1–30)

2.1 Week 1: Introduction to QA & SDLC

2.1.1 Day 1: What is Quality Assurance? Role of a QA

Quality Assurance (QA) ensures that software meets specified requirements and delivers a defect-free user experience. A QA professional verifies functionality, performance, and usability through systematic testing.

Key Responsibilities of a QA:

- Designing and executing test cases.
- Identifying and reporting defects.
- Collaborating with developers and stakeholders.
- Ensuring product quality aligns with user expectations.

Example: A QA tester verifies that a banking app allows secure fund transfers without errors.

Exercise:

1. List three industries where QA is critical.
2. Describe a scenario where a QA could prevent a major software issue.

2.1.2 Day 2: Software Development Life Cycle (SDLC) Overview

The SDLC outlines phases of software development: Planning, Requirements Analysis, Design, Implementation, Testing, Deployment, and Maintenance. QA is primarily involved in the Testing phase but influences requirements and design.

Example: During the Testing phase, a QA ensures a mobile app's login feature works across devices.

Exercise:

1. List the SDLC phases and QA's role in two of them.
2. Explain how QA contributes to the Requirements Analysis phase.

2.1.3 Day 3: Software Testing Life Cycle (STLC)

The STLC defines testing phases: Requirement Analysis, Test Planning, Test Case Development, Test Environment Setup, Test Execution, and Test Closure.

Example: In Test Case Development, a QA writes test cases to verify an e-commerce checkout process.

Exercise:

1. Outline the STLC phases.
2. Describe one activity in the Test Execution phase.

2.1.4 Day 4: Types of Testing

Testing is categorized as:

- **Manual vs. Automation:** Manual involves human execution; automation uses scripts.
- **Functional vs. Non-Functional:** Functional tests features (e.g., login); non-functional tests performance (e.g., load time).

Example: Functional: Verify a search button returns results. Non-Functional: Ensure the search loads in under 2 seconds.

Exercise:

1. Classify three testing scenarios as Functional or Non-Functional.
2. Explain one benefit of manual testing over automation.

2.1.5 Day 5: QA Deliverables

QA deliverables include:

- **Test Plan:** Outlines testing scope and strategy.
- **Test Cases:** Define steps to verify functionality.
- **Bug Reports:** Document defects.

Example Test Case:

```
ID: TC001
Description: Verify user login with valid credentials.
Steps: 1. Enter username. 2. Enter password. 3. Click Login.
Expected Result: User is logged in.
```

Exercise:

1. Write a sample test case for a registration form.
2. List three components of a test plan.

2.2 Week 2: Manual Testing Basics

2.2.1 Day 6: Requirement Analysis for QA

QAs review requirements to identify testable features and potential ambiguities.

Example: For a shopping app, a QA identifies that “add to cart” must support multiple items.

Exercise:

1. Review a sample requirement and list three testable features.
2. Identify one ambiguous requirement and suggest clarification.

2.2.2 Day 7: Writing Test Cases – Best Practices

Test cases should be clear, concise, and cover all scenarios. Use a standard template: ID, Description, Steps, Expected Result.

Example:

```
ID: TC002
Description: Verify password validation.
Steps: 1. Enter password < 6 characters. 2. Click Submit.
Expected Result: Error message displayed.
```

Exercise:

1. Write two test cases for a payment gateway.
2. List three best practices for test case writing.

2.2.3 Day 8: Test Case Design Techniques

- **Boundary Value Analysis (BVA):** Tests edge cases (e.g., min/max values).
- **Equivalence Partitioning (EP):** Divides inputs into valid/invalid groups.

Example: For an age field (18–60):

- BVA: Test 17, 18, 60, 61.
- EP: Test one value in 18–60, one <18, one >60.

Exercise:

1. Design test cases for a field accepting 1–100 using BVA.
2. Apply EP to a username field (4–20 characters).

2.2.4 Day 9: Test Execution & Defect Reporting

Test execution involves running test cases and comparing actual vs. expected results. Defects are logged with details like steps to reproduce and severity.

Example Bug Report:

```
ID: BUG001
Summary: Login fails with valid credentials.
Steps to Reproduce: 1. Enter valid username/password. 2. Click Login.
Actual Result: Error message displayed.
Expected Result: User logged in.
```

Exercise:

1. Execute a sample test case and document results.
2. Write a bug report for a failed test case.

2.2.5 Day 10: Mini Project – Write Test Cases for a Sample App

Students write test cases for a sample app, such as a to-do list application.

Exercise:

1. Write five test cases covering key features.
2. Include at least one BVA and one EP test case.

2.3 Week 3: Defect Life Cycle & Test Management

2.3.1 Day 11: Defect Life Cycle

The defect life cycle includes: New, Open, Assigned, Fixed, Retest, Closed.

Example: A bug (login failure) is marked "New," assigned to a developer, fixed, retested, and closed.

Exercise:

1. Diagram the defect life cycle with descriptions.
2. Describe what happens in the "Retest" stage.

2.3.2 Day 12: Severity vs Priority in Defect Reporting

- **Severity:** Impact of the defect (e.g., Critical, Minor).
- **Priority:** Urgency of fixing (e.g., High, Low).

Example: A crash (High Severity, High Priority) vs. a typo (Low Severity, Low Priority).

Exercise:

1. Classify three sample defects by severity and priority.
2. Explain why a low-severity defect might have high priority.

2.3.3 Day 13: Test Plan Creation

A test plan includes scope, objectives, resources, schedule, and deliverables.

Example Test Plan Section:

```
Scope: Test login, registration, and checkout features.
Objectives: Ensure 100% functional coverage.
Resources: 2 QAs, TestRail, Chrome browser.
```

Exercise:

1. Write a test plan section for a mobile app.
2. List three risks to include in a test plan.

2.3.4 Day 14: Hands-on with Test Case Templates & Bug Reporting

Students use templates to write test cases and bug reports.

Exercise:

1. Create a test case template and write two test cases.
2. Write a bug report using a standard template.

2.3.5 Day 15: Mini Challenge – Simulate Testing Scenario

Students simulate testing a feature, logging defects, and updating a test plan.

Exercise:

1. Execute three test cases and log one defect.
2. Update a test plan based on test results.

2.4 Week 4: SDLC Models & Communication

2.4.1 Day 16: Waterfall vs Agile vs V-Model

- **Waterfall:** Sequential, testing after development.
- **Agile:** Iterative, testing in sprints.
- **V-Model:** Testing paired with each development phase.

Example: In Agile, QAs test user stories each sprint; in Waterfall, testing occurs after coding.

Exercise:

1. Compare QA roles in Waterfall vs. Agile.
2. List one advantage of the V-Model for QA.

2.4.2 Day 17: Introduction to Agile for QA – Scrum Basics

Scrum includes sprints, stand-ups, and roles (Scrum Master, Product Owner, Team).

Example: A QA attends daily stand-ups to report testing progress.

Exercise:

1. List three Scrum ceremonies.
2. Describe the QA's role in sprint planning.

2.4.3 Day 18: QA in Agile Projects

QAs write test cases for user stories, participate in stand-ups, and test incrementally.

Exercise:

1. Write a test case for an Agile user story.
2. Explain how QAs contribute to retrospectives.

2.4.4 Day 19: Communication with Developers & Product Owners

QAs communicate defects clearly and align with product owners on requirements.

Example: A QA explains a bug to a developer with steps to reproduce.

Exercise:

1. Write an email reporting a bug to a developer.
2. Role-play a discussion with a product owner.

2.4.5 Day 20: Review + Manual Testing Practice Session

Students practice test case execution and communication in group activities.

Exercise:

1. Execute a set of test cases and log defects.
2. Present test results to peers.

3 Month 2: Advanced Testing Techniques & Tools (Days 31–60)

3.1 Week 5: Advanced Testing Concepts

3.1.1 Day 21: Smoke, Sanity, Regression Testing

- **Smoke Testing:** Verifies major features work.
- **Sanity Testing:** Checks specific fixes.
- **Regression Testing:** Ensures new changes don't break existing functionality.

Example: Smoke test: Verify login works. Regression test: Re-test login after a new feature is added.

Exercise:

1. List three features for a smoke test in a banking app.
2. Design a regression test case.

3.1.2 Day 22: Integration Testing, System Testing

- **Integration Testing:** Tests interactions between modules.
- **System Testing:** Tests the entire system end-to-end.

Example: Integration: Test API connection between payment and order modules.

Exercise:

1. Write an integration test case for a shopping cart.
2. Describe a system testing scenario.

3.1.3 Day 23: UAT, Alpha & Beta Testing

- **UAT:** Users validate the system meets needs.
- **Alpha/Beta Testing:** Early testing in controlled (Alpha) or real-world (Beta) environments.

Exercise:

1. Plan a UAT session for a travel app.
2. List two differences between Alpha and Beta testing.

3.1.4 Day 24: Exploratory Testing & Ad-hoc Testing

- **Exploratory Testing:** Tests without predefined cases, guided by intuition.
- **Ad-hoc Testing:** Informal testing to find defects.

Exercise:

1. Perform exploratory testing on a sample app and log findings.
2. Explain when ad-hoc testing is useful.

3.1.5 Day 25: Mini Project – Regression Suite for a Web App

Students create a regression test suite for a web app.

Exercise:

1. Write five regression test cases.
2. Execute the suite and report results.

3.2 Week 6: Test Design & Static Testing

3.2.1 Day 26: Static Testing – Reviews, Walkthroughs

Static testing involves reviewing documents (e.g., requirements, code) to find issues early.

Example: A QA reviews a requirements document to identify ambiguities.

Exercise:

1. Review a sample requirements document and list three issues.
2. Describe the difference between a review and a walkthrough.

3.2.2 Day 27: Decision Table Testing & State Transition Testing

- **Decision Table Testing:** Maps conditions to outcomes.
- **State Transition Testing:** Tests system state changes.

Example: Decision Table for login: Conditions (Valid/Invalid credentials) □ Outcomes (Success/Fail).

Exercise:

1. Create a decision table for a discount system.
2. Draw a state transition diagram for a ticket booking system.

3.2.3 Day 28: Use Case Testing & User Story Mapping

- **Use Case Testing:** Tests based on use cases.
- **User Story Mapping:** Visualizes user journeys.

Exercise:

1. Write a test case based on a use case.
2. Create a user story map for a fitness app.

3.2.4 Day 29: Risk-Based Testing & Traceability Matrix

- **Risk-Based Testing:** Focuses on high-risk areas.
- **Traceability Matrix:** Links requirements to test cases.

Example Traceability Matrix:

Requirement ID	Description	Test Case ID
R1	User login	TC001

Exercise:

1. Identify three risks for a payment system and prioritize tests.
2. Create a traceability matrix for five requirements.

3.2.5 Day 30: Test Design Techniques Project

Students apply test design techniques to a project.

Exercise:

1. Create test cases using decision tables and state transitions.
2. Include a traceability matrix.

3.3 Week 7: Test Management Tools

3.3.1 Day 31: JIRA for Test Case & Bug Management

JIRA tracks test cases, bugs, and workflows.

Example: A QA creates a JIRA ticket for a bug with steps to reproduce.

Exercise:

1. Create a JIRA test case ticket.
2. Log a bug in JIRA with all required fields.

3.3.2 Day 32: Introduction to TestRail / Zephyr / qTest

These tools manage test cases, executions, and reports.

Exercise:

1. List three features of TestRail.
2. Compare TestRail and Zephyr for QA tasks.

3.3.3 Day 33: Hands-on with Test Execution & Reporting

Students execute test cases and generate reports using a test management tool.

Exercise:

1. Execute five test cases in TestRail and log results.
2. Generate a test execution report.

3.3.4 Day 34: Defect Tracking Best Practices

Best practices include clear descriptions, screenshots, and prioritization.

Exercise:

1. Write a defect report following best practices.
2. Review a peer's defect report and suggest improvements.

3.3.5 Day 35: Mini Project – Manage a Test Cycle using JIRA

Students manage a test cycle in JIRA, including test cases and defects.

Exercise:

1. Create a test cycle with five test cases in JIRA.
2. Log two defects and track their lifecycle.

3.4 Week 8: Performance & Security Testing Basics

3.4.1 Day 36: Introduction to Performance Testing

Performance testing includes Load (handling normal traffic) and Stress (testing limits).

Example: Load test: Ensure an app handles 1,000 users. Stress test: Test until it crashes.

Exercise:

1. Define a load test scenario for an e-commerce site.
2. Explain the purpose of stress testing.

3.4.2 Day 37: Tools Overview – JMeter Basics

JMeter simulates user loads to test performance.

Example: A JMeter script tests a website's response time under 500 users.

Exercise:

1. Create a basic JMeter test plan for a login page.
2. List three JMeter features.

3.4.3 Day 38: Introduction to Security Testing Concepts

Security testing identifies vulnerabilities like SQL injection or weak authentication.

Exercise:

1. List three common security vulnerabilities.
2. Describe how QA can test for weak passwords.

3.4.4 Day 39: Common Vulnerabilities (OWASP Top 10)

The OWASP Top 10 lists critical vulnerabilities, e.g., Cross-Site Scripting (XSS).

Exercise:

1. Explain two OWASP Top 10 vulnerabilities.
2. Write a test case to check for XSS.

3.4.5 Day 40: Review + Advanced Concepts Practice

Students practice performance and security testing concepts.

Exercise:

1. Design a performance test plan.
2. Write two security test cases.

4 Month 3: Automation, Tools & Job Prep (Days 61–90)

4.1 Week 9: Automation Testing Basics

4.1.1 Day 41: What is Automation Testing?

Automation testing uses scripts to execute tests, saving time for repetitive tasks.

Example: Automating login tests for multiple browsers.

Exercise:

1. List three benefits of automation testing.
2. Identify one test unsuitable for automation.

4.1.2 Day 42: Introduction to Selenium WebDriver

Selenium WebDriver automates browser interactions.

Exercise:

1. List three Selenium WebDriver components.
2. Explain how WebDriver interacts with browsers.

4.1.3 Day 43: Setting up Selenium with Java/Python

Students set up a Selenium environment with Java or Python.

Example Setup (Python):

```
from selenium import webdriver
driver = webdriver.Chrome()
driver.get(”https://example.com”)
```

Exercise:

1. Set up a Selenium environment and run a sample script.
2. List three dependencies for Selenium with Python.

4.1.4 Day 44: Writing Basic Selenium Test Scripts

Students write scripts to automate simple tasks like form submissions.

Example Script:

```
from selenium import webdriver
driver = webdriver.Chrome()
driver.get(”https://example.com/login”)
driver.find_element_by_id(”username”).send_keys(”user”)
driver.find_element_by_id(”password”).send_keys(”pass”)
driver.find_element_by_id(”login”).click()
```

Exercise:

1. Write a Selenium script to automate a search.
2. Add assertions to verify results.

4.1.5 Day 45: Mini Project – Automate Login Test

Students automate a login test for a sample web app.

Exercise:

1. Write a Selenium script for login with valid/invalid credentials.
2. Execute the script and report results.

4.2 Week 10: Automation Frameworks & CI/CD

4.2.1 Day 46: TestNG/JUnit Framework Basics

TestNG (Java) and JUnit (Java) organize and run automated tests.

Example TestNG:

```
@Test
public void testLogin() {
    // Selenium code
}
```

Exercise:

1. Write a TestNG test case for a login feature.
2. List three TestNG annotations.

4.2.2 Day 47: Page Object Model (POM) in Automation

POM organizes code by separating page logic from tests.

Example POM:

```
public class LoginPage {
    WebDriver driver;
    By username = By.id(”username”);
    public void enterUsername(String user) {
        driver.find_element(username).send_keys(user);
    }
}
```

Exercise:

1. Create a POM class for a login page.
2. Write a test using the POM class.

4.2.3 Day 48: Introduction to CI/CD Tools – Jenkins Basics

Jenkins automates test execution in a CI/CD pipeline.

Exercise:

1. List three benefits of Jenkins for QA.
2. Describe a Jenkins pipeline for running Selenium tests.

4.2.4 Day 49: Running Automated Tests in CI/CD Pipeline

Students configure Jenkins to run Selenium tests.

Exercise:

1. Set up a Jenkins job for a Selenium script.
2. Execute the job and review logs.

4.2.5 Day 50: Automation Project – E-commerce App Tests

Students automate tests for an e-commerce app using POM and TestNG/JUnit.

Exercise:

1. Write three automated test cases.
2. Run tests in a Jenkins pipeline.

4.3 Week 11: Capstone Project

4.3.1 Day 51–55: Capstone Project – Manual + Automation Testing of a Web App

Students perform manual and automated testing for a web app, documenting test cases, defects, and automation scripts.

Exercise:

1. Write a test plan and ten manual test cases.
2. Automate three test cases using Selenium and POM.
3. Log defects in JIRA and generate a test report.

4.4 Week 12: Wrap-up & Job Prep

4.4.1 Day 56: Common QA Interview Questions

Common questions include: “How do you prioritize test cases?” and “Explain a complex bug you found.”

Exercise:

1. Prepare answers for three QA interview questions.
2. Practice with a peer.

4.4.2 Day 57: Resume Preparation & GitHub Portfolio Tips

A QA resume highlights testing skills, tools, and projects.

Exercise:

1. Draft a resume section for a QA role.
2. Create a GitHub repository with test scripts.

4.4.3 Day 58: Test Case & Bug Report Portfolio Setup

A portfolio includes test cases, bug reports, and automation scripts.

Exercise:

1. Compile a portfolio with five test cases and two bug reports.
2. Add one automation script to the portfolio.

4.4.4 Day 59: Mock Interview – Manual + Automation Focus

Students participate in mock interviews covering manual and automation testing.

Exercise:

1. Complete a mock interview.
2. Incorporate feedback for improvement.

4.4.5 Day 60: Final Project Presentation + Feedback

Students present their capstone project and receive feedback.

Exercise:

1. Create a presentation for the capstone project.
2. Deliver it to peers and incorporate feedback.