QA Automation Testing Course

A comprehensive, practice-ready program built around seven modules split into topics and subtopics. Every module includes examples, diagram descriptions so designers can add visuals later, practice exercises, and project ideas.

Module 1 - Introduction to QA & Automation Testing

1.1 What is Software Testing?

- **Definition:** Verification and validation activity that ensures software meets business and technical requirements before and after release.
- **Why testing is needed:** Prevents defects from reaching customers, protects brand reputation, and reduces support costs.
- **Cost of defects:** The later a bug is found in the lifecycle, the more expensive it is to fix.
- **SDLC vs STLC:** SDLC covers the entire software delivery process; STLC focuses specifically on testing phases inside the SDLC.

Example: A banking application released without adequate testing caused incorrect interest calculations that impacted 10,000 users. Early testing would have caught the regression before production.

1.2 Types of Testing

- **Functional testing:** Validates business flows such as login, checkout, and refunds.
- **Non-functional testing:** Examines performance, security, usability, and reliability.
- **Manual vs automation:** Manual explores, automation repeats predictable flows quickly.
- **Black-box vs white-box:** Black-box evaluates external behavior; white-box validates internal logic and code paths.

Diagram to add: "Testing Classification Tree" showing functional/non-functional split, and manual/automation overlays.

1.3 What is Automation Testing?

- **Why automation:** Accelerates regression testing, enables 24/7 unattended execution, and increases coverage.
- **When to automate:** High-volume, stable, repeatable scenarios (e.g., login validation for every release).
- **When not to automate:** Visual checks or rapidly changing UIs where maintenance outweighs benefit.
- **Automation testing lifecycle:** Tool selection -> framework setup -> test development -> execution -> reporting -> maintenance.

Example: Regression login scripts are ideal for automation; pixel-perfect visual checks are better suited for manual exploration.

1.4 Automation Tools Overview

- **Selenium:** UI automation for web applications.
- **Postman:** Manual and automated API checks.
- **RestAssured:** Code-first API testing for Java tech stacks.
- **Jenkins:** On-prem CI server to orchestrate automation runs.
- **GitHub Actions:** Cloud CI/CD service with workflow-as-code.
- **Other CI/CD tools:** Azure DevOps, GitLab CI, CircleCI.

1.5 Role of an Automation Tester

- Develop maintainable test scripts and utilities.
- Design automation frameworks (POM, data-driven, BDD).
- Integrate suites into CI/CD pipelines.
- Execute and monitor API and UI suites, analyze reports, and communicate risk.

Practice Questions

1. What is regression testing, and why does automation help with it?
2. Name five testing types and classify each as functional or non-functional.
3. Describe two scenarios where automation is not recommended.

Module 2 - Core Programming (Python/Java) plus SQL

2.1 Programming Basics

- Variables and constants, strongly typed vs dynamically typed.
- Primitive and reference data types.
- Arithmetic, logical, comparison, and assignment operators.
- Conditional statements (`if`, `elif`, `switch`).
- Looping constructs (`for`, `while`, nested loops, iterators).

Python example:

```
for i in range(5):
    print("Hello QA")
```

2.2 Object-Oriented Programming

- **Classes & objects:** Blueprint vs instance.
- **Inheritance:** Code reuse with `extends`/`super` or subclassing.
- **Polymorphism:** Compile-time (overloading) vs runtime (overriding).
- **Encapsulation:** Access modifiers (`private`, `protected`) and properties.

Example:

```
class Login:
    def __init__(self, user, pwd):
        self.user = user
        self.pwd = pwd
```

2.3 Exception Handling

- `try/except/finally` (Python) or `try/catch/finally` (Java).
- Creating and raising custom exceptions to wrap framework-specific errors.

2.4 File Handling

- Reading structured test data (CSV, JSON, YAML).
- Writing execution logs and screenshots to disk.

2.5 SQL for Testers

- CRUD statements: `SELECT`, `INSERT`, `UPDATE`, `DELETE`.
- Joins (INNER/LEFT/RIGHT) to merge tables.
- Constraints: primary keys, foreign keys, unique, not null, check.
- Writing verification queries against AUT databases.

Example query:

```
SELECT name, balance
FROM customers
WHERE balance < 0;
```

Practice Exercises

- Implement a `Calculator` class with add, subtract, multiply, divide methods and unit tests.
- Write SQL to fetch the top five highest salaries per department.
- Parse a CSV test data file and print unique test case IDs.

Module 3 - Selenium WebDriver for UI Automation

3.1 Introduction to Selenium

- Selenium suite components (IDE, Grid, WebDriver, RC legacy).
- WebDriver architecture: language bindings -> JSON Wire/W3C protocol -> browser drivers -> browsers.

Diagram to add: Selenium architecture flowchart showing command flow from tests to browsers via drivers.

3.2 Locators

- Basic locators: ID, Name, ClassName, LinkText, PartialLinkText, TagName.
- CSS Selectors: attribute, pseudo-classes, combinators.
- XPath: absolute vs relative, axes, functions (`contains`, `starts-with`).

Example:

```
driver.find_element(By.XPATH, "//input[@id='username']")
```

3.3 WebDriver Commands

- Element interactions: `click`, `send_keys`, `clear`, `submit`.
- Browser actions: `get`, `back`, `forward`, `refresh`, cookies handling.
- Wait strategies: implicit waits, explicit waits (WebDriverWait), fluent waits.

3.4 Handling UI Components

- Dropdowns via `Select` class, custom dropdown patterns.
- JavaScript alerts (accept, dismiss, send_keys).
- Frames and nested frames switching.
- HTML tables parsing and validations.
- Window/tab switching using handles.

3.5 Page Object Model (POM)

- Separation of test logic and page locators.
- Constructor injection for WebDriver instances.
- Centralized test data handling and reusable actions.

3.6 Test Frameworks (TestNG or PyTest)

- Annotations/decorators for setup/teardown and grouping.
- Assertion strategies and soft asserts.
- Parallel execution and HTML/XML report generation.

Practice UI Automation Task

Automate the Facebook login page using XPath and POM patterns. Include:
- Explicit waits for dynamic elements.
- Positive and negative credential datasets.
- Reporting of pass/fail state with screenshots on failure.

Module 4 - API Testing with Postman and RestAssured

4.1 What is an API?

- Client-server interaction through HTTP verbs.
- Request structure (URL, headers, body, auth).
- Response structure (status codes, headers, JSON/XML payloads).

4.2 Postman Basics

- Creating collections and environments.
- Using variables and pre-request scripts.
- Writing Tests tab assertions in JavaScript.

Example:

```
pm.test("Status is 200", function () {
    pm.response.to.have.status(200);
});
```

4.3 Automation with RestAssured (Java)

- Fluent API for GET/POST/PUT/DELETE.
- Serialization/deserialization of JSON payloads.
- Request/response logging and schema validation.

Example:

```
given()
    .baseUri("https://reqres.in")
.when()
    .get("/api/users")
.then()
    .statusCode(200);
```

Practice Assignment

Use Postman and RestAssured to test `https://reqres.in/api/users`:
- Validate pagination metadata.
- Create a new user and verify response body.
- Update the same user and assert `updatedAt` timestamp.

Module 5 - BDD with Cucumber

5.1 What is BDD?

- Encourages collaboration via business-readable scenarios.
- Gherkin syntax: `Feature`, `Scenario`, `Given-When-Then`, `And`, `But`.

Example Feature:

```
Scenario: Successful Login
  Given User is on login page
  When User enters valid credentials
  Then User should land on homepage
```

5.2 Cucumber Framework

- Organizing feature files, step definition glue code, and hooks.
- Sharing state via context objects or dependency injection.
- Generating HTML/JSON reports.

5.3 Integration with Selenium and API Testing

- Wiring WebDriver or API clients inside step definitions.
- Reusing page objects and service clients across steps.
- Tag-based execution for fast regression packs.

Practice BDD Assignment

Write a feature for "Search product on Amazon" covering:
- Scenario Outline with multiple search keywords.
- Steps for validating search suggestions and product results.
- Hooks to launch and close browsers.

Module 6 - CI/CD and Test Automation Integration

6.1 What is CI/CD?

- **Continuous Integration:** Frequent code merges, automated build/test, fast feedback.
- **Continuous Deployment/Delivery:** Automated promotion of tested builds to staging/production.

6.2 Jenkins

- Job configuration: freestyle vs pipeline.
- Build triggers (SCM polling, webhooks, schedules).
- Executing Selenium or API suites via shell/batch steps.
- Publishing JUnit/HTML reports and sending notifications.

6.3 Git and GitHub

- Branching strategies (feature, release, hotfix).
- Pull request workflows and code reviews.
- Resolving merge conflicts and rebasing best practices.

6.4 Running Selenium Tests in Pipelines

- Headless execution via Chrome/Firefox.
- Containerized runs (Docker image with browser + driver).
- Parallelization and artifact collection (screenshots, logs).

Practice Task

Integrate any Selenium smoke suite with Jenkins:
- Configure Git webhook trigger.
- Parameterize environment URLs.
- Schedule daily runs and collect HTML report artifacts.

Module 7 - End-to-End Automation Project and Certification

7.1 Project Requirements

Build a unified automation framework that includes:
- Selenium UI regression pack.
- API coverage for key services.
- Database verification layer.
- Logging, screenshots, and consolidated reporting.
- CI/CD integration with scheduled execution.

7.2 Example Capstone Project: E-commerce Platform

Automate the following flows:
1. User authentication with positive/negative scenarios.
2. Product search with filters and pagination.
3. Add-to-cart and cart validation.
4. Checkout including payment simulation.
5. API validation for order creation endpoint.
6. Database check to confirm order persistence.

7.3 Final Assessment

- 50 multiple-choice questions covering theory and tooling.
- Practical automation exam: build and execute a critical test script.
- Mock interview round focusing on framework choices and debugging.

Project Ideas

- Flight booking web + API suite.
- Banking customer onboarding with database reconciliation.
- Healthcare appointment scheduling with REST and UI checks.

Next Steps

- Review the content and request adjustments if needed.
- Once approved, the material is ready for PDF export with cover page, diagram placeholders, and certificate layout.