



Advanced Test Automation Course  
Specially designed for working professionals



# Table of Contents

1. Core Java	3
2. TestNG	5
3. Web Automation using Selenium	5
4. Automation Frameworks	7
5. BDD using Cucumber	7
6. API Testing with Postman	8
7. API Test automation using RestAssured	9
8. Mobile Automation using Appium	10
9. Desktop Test Automation using WinAppDrive	10
10. Version Control, Git & Github	11
11. DevOps & CI/CD	12
12. Security Testing using Zap Proxy	12
13. Performance Testing using Jmeter	12
14. Cross-Browser Testing using BrowserStack	13
15. Bonus class - Web Automation using Playwright	13

# About Advanced Test Automation Course

Welcome to the Advanced Automation course, where we will provide you with the knowledge and skills necessary to become a highly effective Software Development Engineer in Test.

In this comprehensive training program, you will learn how to apply your programming and testing expertise to build and maintain robust, scalable, and efficient software testing frameworks. Our experienced trainers will guide you through the latest tools, technologies, and best practices in the field of software testing, including Core Java, TestNG, Web Automation using Selenium, Automation Frameworks, BDD using Cucumber, API Testing with Postman, API Test Automation using RestAssured, Mobile Automation using Appium, Version Control, Git & Github, DevOps & CI/CD, Security Testing using Zap proxy, Performance Testing using Jmeter, Cross-Browser Testing using BrowserStack.

By the end of the program, you will be equipped with the skills to design and implement effective test automation frameworks, automate testing for web and mobile applications, perform API testing, implement continuous integration and deployment, and execute security and performance testing. Additionally, you will be able to collaborate seamlessly with developers, testers, and stakeholders to ensure the quality and reliability of the software products.

Our training program is designed for both beginners and experienced professionals who want to enhance their skills and knowledge in the field of software testing. Our trainers have years of hands-on experience in the software testing industry and will provide you with practical training that is relevant to the industry. So, if you are looking to advance your career as an SDET or want to enter the field of software testing, this training program is the perfect opportunity for you. Enroll now and take the first step towards becoming an expert Software Development Engineer in Test.

## Course Content

### 1. Core Java

- a. Introduction and Installation
  - i. History of Java
  - ii. Installing the Java Development Kit
  - iii. The basics of the Java programming language
  - iv. Writing and running a simple Java program
- b. Structure of programming language
  - i. Datatypes
    1. Primitive - Int, Boolean, Char, float, long, double
    2. Non-Primitive - String, arrays
  - ii. Operators
  - iii. Keywords

- iv. Access modifiers
- c. Control Flow Statements
  - i. If – else
  - ii. If – else – if
  - iii. Nested if
  - iv. Switch case
- d. Looping in Java
  - i. For Loop
  - ii. While Loop
  - iii. Do – while Loop
  - iv. Continue statement
  - v. Break statement
  - vi. Inner Loop / Nested Loop
- e. Object-oriented Concepts
  - i. Classes and Objects
  - ii. Object creation
  - iii. Reference variable
  - iv. Global and local variables
  - v. Constructors
  - vi. Aggregation
  - vii. Composition
  - viii. Encapsulation
  - ix. Inheritance
    - 1. Single inheritance
    - 2. Multilevel inheritance
    - 3. Hierarchical inheritance
  - x. Polymorphism
    - 1. Method overloading
    - 2. Method overriding
  - xi. Abstraction
    - 1. Abstract class
    - 2. Interface
  - xii. Variables
    - 1. Local variable
    - 2. Instance variable
    - 3. Static / Global variable
  - xiii. Methods
    - 1. Declaration
    - 2. Parameterization
    - 3. Returning value
    - 4. Automatic promotion
    - 5. Method signature
- f. super keyword
- g. this keyword
- h. final keyword
- i. Typecasting

- j. Java packages
- k. Exception Handling
- l. Generics
- m. Collections & streams - ArrayList, Hashmaps, hashtables
- n. Multithreading
- o. File I/O
- p. Lambda Expressions
- q. Functional interfaces
- r. Method References
- s. Default Methods
- t. Optional Class
- u. Serialization
- v. Garbage Collector
- w. New Date/Time API

## 2. TestNG

- a. Introduction to TestNG
  - i. History and Overview of TestNG
  - ii. Installing TestNG in Eclipse or IntelliJ IDEA
  - iii. Writing and running a simple TestNG test
- b. TestNG Annotations
  - i. Understanding test annotations and attributes
  - ii. Configuring test behavior using annotations
  - iii. Using TestNG assertions
- c. TestNG Test Suites
  - i. Creating test suites in TestNG
  - ii. Grouping tests in test suites
  - iii. Running test suites using TestNG
- d. TestNG Parameterization
  - i. Parameterizing tests in TestNG
  - ii. Data Providers and Data Sources
  - iii. TestNG Factory annotation
- e. Data-Driven Testing in TestNG
  - i. Data-driven testing concepts and best practices
  - ii. Data-driven testing using TestNG
  - iii. Reading test data from Excel, CSV, or JSON files
- f. Parallel Test Execution in TestNG
  - i. Overview of parallel test execution in TestNG
  - ii. Configuring parallel test execution in TestNG
  - iii. Parallelism at the test and suite level
- g. TestNG Listeners
- h. Reports using TestNG
- i. Final Project
  - i. Students will complete a final project that demonstrates their understanding of TestNG for test automation. The project will involve writing a set of automated tests using

TestNG, and one of the integration tools covered in the course. The project will be graded based on functionality, design, and code quality.

### 3. Web Automation using Selenium

- a. Introduction to Test Automation
  - i. What is Automation Testing?
  - ii. When we switch to Automation Testing?
  - iii. Why Automation testing?/Advantages/Disadvantages
  - iv. Automation Testing Tools
- b. Selenium IDE
- c. Selenium WebDriver Overview
- d. Locating Elements
  - i. id
  - ii. name
  - iii. className
  - iv. tagName
  - v. link text
  - vi. partialLinkText
  - vii. CSS selector
  - viii. Xpath
    - 1. Absolute
    - 2. Relative X-path
      - a. Contains
      - b. Ancestor/descendants
      - c. And/OR
      - d. Parent
      - e. Starts With
      - f. Preceding/following
      - g. Siblings
- e. Handling Multiple Elements
- f. Navigation in selenium-
- g. Handling Edit-box-
- h. Handling Disabled Element
- i. Taking Screenshot
- j. Performing Scroll down Action
- k. Handling Drag and Drop
- l. Handling Keyboard and Mouse Actions
- m. Handling Mouse Hover
- n. Keyword Events using Action class
- o. Handling Popups (web-based and Window-based)
- p. Handling New Windows/New Tabs
- q. Scrolling on a web page using JavaScript Executor
- r. Types of Alerts
  - i. Handling Alerts
  - ii. Handling multiple windows & tabs

- iii. Verify Page title in Selenium WebDriver
- s. Handling links
- t. Handling Radio button & Check-box
- u. Handling WebTable
- v. Handling Drop Down using Select class
- w. Methods under Select class
- x. Resize operations
- y. Handling File Upload
- z. What is an IFrame
  - i. Identifying an IFrame
  - ii. Switching to a specific IFrame in Selenium WebDriver
- aa. Working with Excels
- bb. Handling Synchronisation issues by using implicitlyWait and Explicitly Wait

#### 4. Automation Frameworks

- a. Stages of Automation Framework Design
- b. Automation Approach
- c. How to scale Automation?
- d. Explanation of Hybrid Framework with a Combination of
  - i. Data-Driven
  - ii. Keyword-Driven
  - iii. Method-Driven
  - iv. Behavior-Driven
- e. Design Principles
  - i. SOLID
  - ii. DRY
  - iii. KISS
  - iv. YAGNI
- f. Design Patterns
  - i. Page Object Model Pattern
  - ii. Factory Design Pattern
  - iii. Facade Pattern
  - iv. Singleton Pattern
  - v. Fluent Page Object Model

#### 5. BDD using Cucumber

- a. Introduction to Cucumber and BDD
  - i. Overview of BDD and its benefits
  - ii. Introduction to Cucumber and its Architecture
  - iii. Understanding the Gherkin syntax
- b. Writing Feature Files

- i. Creating feature files for different scenarios
    - ii. Defining scenarios and steps in Gherkin syntax
    - iii. Best practices for writing feature files
  - c. Implementing Step Definitions
    - i. Creating step definitions in Java
    - ii. Mapping Gherkin syntax to Java code
    - iii. Implementing parameterization in step definitions
  - d. Running Cucumber Tests
    - i. Running Cucumber tests from the command line
    - ii. Generating HTML reports for Cucumber tests
    - iii. Debugging Cucumber tests
  - e. Integrating Cucumber with Selenium
    - i. Introduction to Selenium WebDriver
    - ii. Integrating Selenium with Cucumber for UI testing
    - iii. Implementing Page Object Model (POM) with Cucumber and Selenium
  - f. Integrating Cucumber with TestNG
    - i. Integrating TestNG with Cucumber for Test Automation
    - ii. Writing maintainable and scalable Cucumber tests
  - g. Advanced Cucumber Topics
    - i. Working with data tables in Cucumber
    - ii. Using hooks in Cucumber
    - iii. Cucumber Reports
    - iv. Integration with Allure reports
    - v. Integration with Extent reports
  - h. Final Project
    - i. Students will complete a final project that demonstrates their understanding of Cucumber in BDD. The project will involve writing a set of feature files, implementing step definitions, and running the tests using Cucumber. The project will be graded based on functionality, design, and code quality.

## 6. API Testing with Postman

- a. Introduction to API Testing with Postman
  - i. Overview of API testing and its benefits
  - ii. Introduction to Postman and its Architecture
  - iii. Understanding the Postman environment setup
- b. Setting Up Postman and Creating Requests
  - i. Installing and setting up Postman on Windows/Mac
  - ii. Creating and sending requests with Postman
  - iii. Understanding HTTP request methods, headers, and parameters
- c. Creating and Managing Collections
  - i. Creating collections in Postman
  - ii. Managing requests and responses within collections
  - iii. Using variables and environments in Postman
- d. API Testing with Postman
  - i. Writing API tests in Postman



- ii. Understanding the Postman testing framework
  - iii. Implementing assertions in API tests
- e. Advanced Postman Features
  - i. Using Postman for exploratory API testing
  - ii. Using Postman for performance testing
  - iii. Understanding Postman's built-in features for load testing
- f. Integrating Postman with Newman
  - i. Overview of the Newman command-line tool
  - ii. Using Newman to run Postman collections
  - iii. Integrating Newman with GitHub Actions for continuous integration and delivery
- g. Advanced Topics in API Testing
  - i. Understanding and handling authentication in APIs
  - ii. Working with APIs that require authorization tokens
  - iii. Using Postman for API Documentation
- h. Final Project
  - i. Students will complete a final project that demonstrates their understanding of API testing with Postman. The project will involve creating a set of API tests, integrating them with Newman, and running the tests using GitHub Actions. The project will be graded based on functionality, design, and code quality.

## 7. API Test automation using RestAssured

- a. Introduction to API Testing with RestAssured
  - i. What is Rest Assured?
  - ii. Introduction to Rest Assured and its Architecture
  - iii. Understanding Rest Assured Environment Setup
- b. HTTP Request and Response in RestAssured
  - i. Understanding HTTP methods, Headers, Parameters, Request Body & Response Body
  - ii. Understanding API Documentation
  - iii. Understanding Rest-Assured Testing Frameworks
- c. API Testing with RestAssured
  - i. API test to verify GET call using BDD format of Rest Assured
  - ii. API test to verify GET call using Request Specification & Response Specification in Rest Assured
  - iii. Integrating TestNG to run Rest Assured tests
  - iv. API test using Authentication Key to verify GET call
  - v. Automating POST call with the body as Java String
  - vi. Automating POST call with the body as JSON Object
  - vii. Automating PUT & DELETE call
  - viii. Handling Authentications in API using Token.
  - ix. End-to-end test case to verify Create, Update, and DELETE operations of API.
- d. Integrating RestAssured with TestNG & Cucumber
  - i. Integrating Rest Assured with Cucumber
  - ii. Automating GET, POST, PUT, and DELETE calls using Rest Assured with Cucumber
  - iii. Creating API Utils class to increase reusability and enhance the framework
- e. Final Project

- i. Students will complete a final project demonstrating their understanding of API test automation using RestAssured. The project will involve creating a set of API tests, integrating them with TestNG, and running the tests using GitHub Actions. The project will be graded based on functionality, design, and code quality.

## 8. Mobile Automation using Appium

- a. Introduction to Appium and Mobile Automation
  - i. Overview of mobile automation and its benefits
  - ii. Introduction to Appium and its Architecture
  - iii. Understanding the Appium environment setup
- b. Appium Setup and Configuration
  - i. Setting up Appium on Windows, Mac, and Linux
  - ii. Understanding the Appium server and client
  - iii. Creating an Appium project in Eclipse
- c. Creating Appium Tests
  - i. Creating test scripts in Java
  - ii. Understanding the Appium API
  - iii. Implementing implicit and explicit waits
- d. Mobile Automation with Appium
  - i. Automating mobile applications using Appium
  - ii. Understanding the Appium driver and locator strategies
  - iii. Working with mobile elements
- e. Integrating Appium with TestNG
  - i. Integrating TestNG with Appium for mobile testing
  - ii. Writing maintainable and scalable Appium tests using TestNG
- f. Advanced Appium Topics
  - i. Working with mobile gestures in Appium
  - ii. Using the Appium inspector
  - iii. Integrating Appium with Cucumber + TestNG framework
- g. Final Project
  - i. Students will complete a final project demonstrating their understanding of Appium in mobile automation. The project will involve creating a set of test scripts for a mobile application, integrating them with TestNG and Cucumber, and running the tests using Appium. The project will be graded based on functionality, design, and code quality.

## 9. Desktop Test Automation using WinAppDriver

- a. Introduction to WinAppDriver and Desktop Automation
  - i. Overview of desktop app automation and its benefits
  - ii. Overview of WinAppDriver: What it is, how it works, and its architecture
- b. WinAppDriver Setup and Configuration
  - i. Setting up WinAppDriver on Windows
  - ii. Enabling Developer Mode

- iii. Identifying the Desktop App ID
- c. Element Identification
  - i. Identifying UI elements: Using UIA Verify, Inspect tool, and WinAppDriver's built-in inspector
  - ii. Identifying UI elements: Using Inspect.exe
- d. WinAppDriver API and interactions
  - i. Different methods in WinAppDriver API
  - ii. Interacting with UI elements: Clicking, typing, selecting, and more.
  - iii. Handling different types of controls: buttons, text boxes, dropdowns, etc
- e. Integrating WinAppDriver with Cucumber, TestNG, and Page Object Model
  - i. Writing your first WinAppDriver script: Launching a sample desktop application.
  - ii. Designing and implementing the Page Object Model (POM) pattern.
  - iii. Writing BDD tests using Cucumber to run WinAppDriver tests
- f. Advanced WinAppDriver Topics
  - i. Switching Windows
  - ii. Wait and synchronizations
  - iii. Handling dynamic UI elements using XPath and other strategies.
  - iv. Keyboard actions and shortcuts automation.
  - v. Capturing and validating screenshots during test execution.
- g. Final Project
  - i. Students will complete a final project demonstrating their understanding of WinAppDriver desktop automation. The project will involve creating a set of test scripts for a desktop application, integrating them with TestNG and Cucumber, and running the tests using WinAppDriver. The project will be graded based on functionality, design, and code quality.

## 10. Version Control, Git & Github

- a. What is VCS?
- b. Different types of VCS present
- c. What is Git? & Why choose Git?
- d. What is GitHub?
- e. Difference between Git & GitHub
- f. Difference between Local and Remote Branches
- g. Installation of Git
- h. How to create a GitHub account
- i. GitHub dashboard overview
- j. Creating and adding a public SSH key
- k. How to create a GitHub repository and different modules to look after while creating a repo.(public/private, readme.md, gitignore, protecting the branch)
- l. Git Commands:
  - i. How to clone the repo?
  - ii. How to create a branch & checkout?

- iii. How to pull the latest code to the created branch?
- iv. How to commit and push the changes?
- v. How to delete the branch?
- vi. How to reverse the commit?
- vii. Other basic git commands
- m. What is a pull request?
- n. How to create a pull request?
- o. What are merge conflicts?
- p. How to resolve the merge conflicts?

## 11. DevOps & CI/CD

- a. What is DevOps?
- b. Why is it necessary for testers?
- c. What is CI/CD?
- d. Benefits of CI/CD
- e. Different tools used for CI/CD
- f. The CI/CD process flow
- g. Introduction of GitHub Actions for CI/CD
- h. What is workflow?
- i. YAML language introduction and structure
- j. Overview of GitHub marketplace
- k. How to create a workflow file?
- l. What are the different components of CI/CD workflow?
- m. Difference between self-hosted and GitHub cloud-hosted runner
- n. Create a complete CI/CD workflow file for your repository

## 12. Security Testing using Zap Proxy

- a. What is Security?
- b. What is Security Testing?
- c. Importance of Security Testing.
- d. Tools used for Security Testing.
- e. Introduction to Passive and Active Scans
- f. Introduction to OWASP ZAP tool
- g. Installation of tool
- h. Dashboard Overview
- i. How to create a session for the passive scan?
- j. Parameters to consider for active scan
- k. Thresholds and Scan policy change
- l. How to run the active scan?
- m. How to generate the HTML Report?
- n. How to read the HTML Report?

### 13. Performance Testing using Jmeter

- a. What is performance testing?
- b. Why do we need performance testing?
- c. What are the different types of performance testing?
- d. Introduction of Jmeter for performance testing
- e. Installation of Jmeter
- f. Dashboard Overview
- g. Explanation of different components of JMeter script (thread group, thread count, assertions, etc.)
- h. How to capture the requests?
- i. Introduction of Blazemeter for request recording and installation
- j. How to add assertions in the script?
- k. How to add variables in the script?
- l. How to add a result tree in the script?
- m. How to generate an HTML report?
- n. How to read the HTML report?

### 14. Cross-Browser Testing using BrowserStack

- a. Introduction to BrowserStack
  - i. Testing web applications on multiple devices and browsers
  - ii. Understanding cross-browser compatibility issues
  - iii. Using BrowserStack for Responsive Testing
- b. Integrating BrowserStack with Selenium
  - i. Using Selenium to run tests on BrowserStack
  - ii. Creating Selenium tests for BrowserStack
- c. Integrating BrowserStack with Appium
  - i. Using Appium to run tests on BrowserStack
  - ii. Creating Appium tests for BrowserStack
- d. Debugging and Reporting with BrowserStack
  - i. Understanding and using BrowserStack debugging features
  - ii. Using BrowserStack for issue reporting and management
  - iii. Best practices for issue tracking and resolution
- e. Advanced Topics in BrowserStack Testing
  - i. App Live & App Automate
  - ii. The device with SIM
  - iii. Image Injection
  - iv. Audio Injection
  - v. File Injection
  - vi. Biometric Injection

### 15. Bonus class - Web Automation using Playwright

- a. Introduction to playwright
- b. Installation of playwright
- c. Playwright Selectors
  - i. Developer Tool
  - ii. Test Selector, CSS Selector
  - iii. :nth-match and :has-text
  - iv. Layout Selectors
  - v. XPath Selectors
  - vi. Chaining Selectors
  - vii. Playwright Selectors Summary
- d. Brower Context
  - i. Running test cases and in different models of browser
  - ii. Different browser option
- e. Waits
  - i. Wait for the selector and the page loads
  - ii. Build a Custom wait
- f. Page Object Model and Project Structure
  - i. Page ObjectModel
  - ii. Project Structure
  - iii. POM helper methods
- g. Data-Driven Testing
  - i. Makar - parametrize
  - ii. Data for negative testing
  - iii. Stack parametrize for easier maintenance and better test data coverage
- h. CI/CD Integration
  - i. Set up GitHub Action
  - ii. Edit yml file and run testing CI/CD pipeline
  - iii. yml file reference link
  - iv. Use login password via GitHub secrets
  - v. Toggle password between local and remote runs
  - vi. Store password in env file
- i. Final Project
  - i. Students will undertake a final project to showcase their proficiency in Playwright for web application automation. The project will encompass the creation of a suite of test scripts for a web application, the incorporation of these scripts with testing frameworks such as TestNG and Cucumber, and the execution of tests using Playwright. The evaluation of the project will consider factors such as functionality, design, and code quality.