Software Testing and Quality Assurance Tutorial

Published On: July 12, 2024

Software Testing and Quality Assurance

Software testing and QA Architect tutorials provide basic and advanced software testing concepts. This is meant to teach beginners and experts how to test every piece of software before releasing it.

Software Testing & Quality Assurance Tutorial PDF

Introduction to Software Testing and Quality Assurance

Software testing is a procedure that determines whether the software is accurate by considering all of its characteristics (reliability, scalability, portability, reusability, and usability) and assessing how well its components are executed to uncover any errors, faults, or bugs.

Our Software Testing and QA Architect Tutorial covers the following:

- Core Java Concepts
- Manual Testing
- Automation Testing

Core Java Concepts

Classes, objects, inheritance, polymorphism, and encapsulation are examples of Java's OOP features, which facilitate modular and effective code development. To create software that is reliable, scalable, and manageable, one must comprehend OOPS in Java.

The following are the important Java skills for software testing and QA architects:

Java Collections

The one-stop shops for all data manipulation tasks, including storing, searching, sorting, inserting, deleting, and updating data, are Java collections. Java Collection replies as a single object and offers different interfaces and classes using the Java Collection Framework. Below are the Java collections interface methods:

- **Set Interface:** Elements that are redundant or duplicate cannot be stored in a set interface.
- **List Interface:** The Java util package is the source of the List interface.
- **Queue Interface:** Following the First In, First Out (FIFO) principle, the queue is a linear collection that provides data processing operations on the collection members.
- **Deque Interface:** The Java Collections Interface is the interface that a Deque interface inherits. Double-Ended Queue is referred to as DE-Que.
- **Map Interface:** Java Collection Interface is the interface that Map inherits. Duplicate elements cannot be stored on the map.
- **SortedSet Interface:** An ascending mapping is maintained using a sorted set interface. For naturally organized collecting, they are employed.
- **SortedMap Interface:** The element mappings are kept in ascending critical order via a SortedMap interface. SortedSet's map counterpart is called SortedMap.

Array

Mobile App Development Tutorial
Published On: September 25, 2024

Mobile App Development Tutorial Businesses looking to grow and scale can benefit greatly from mobile...

Microsoft Azure Tutorial for Beginners
Published On: September 24, 2024

Microsoft Azure Tutorial for Beginners We can use a wide range of services from Microsoft...

MEAN Stack Tutorial for Beginners
Published On: September 24, 2024

MEAN Stack Tutorial for Beginners 'M' refers to MongoDB, 'E' for Express, 'A' for Angular,...

An object called an array in Java is designed to carry a defined number of elements of similar data types.

Example: An array named “int” will only contain integers, while an array named “string” will only include strings. An array’s length is fixed at the time it is built. After creation, its length is determined.

Syntax

dataType[] arrayName;

arrayName = new dataType[arraySize];

Example

int [] arr;

arr = new int[5];

- “arr” is a group of five integers.
- int[] The symbol arr indicates that ‘arr’ is an integer-type array.
- [] is a symbol for an array.
- It is represented as an array of 5 numbers using new int[5].

Control Statements

The code is run through the Java compiler from top to bottom. The code’s statements are carried out in the order that they appear. However, Java has statements that can be used to control the flow of Java code.

We refer to these expressions as control flow statements. It is one of Java’s core characteristics that ensures a seamless program flow.

Three varieties of control flow statements are available in Java.

- Decision Making statements
    - if statements
    - switch statement

- Loop statements
    - do while loop
    - while loop
    - for loop
    - for-each loop
- Jump statements
    - break statement
    - continue statement

OOPs Principles

Programming languages that employ objects as a key source for implementing code are known as object-oriented programming (OOP) languages, or Java OOPs.

- Using real-world concepts like inheritance, hiding, polymorphism, etc. in programming is the goal of object-oriented programming.
- Ensuring that only the function that operates on the data may access it is the primary goal of object-oriented programming (OOPs).
- This ensures that no other portion of the code can access the data.

The following are OOPS concepts:

- **Class:** In Java, a class is a collection of objects that have similar traits, behaviors, and properties.
- **Object:** The instances of a class that are made in order to use its methods and properties are called objects.
- **Method and method passing:** We may reuse the code without having to type it again thanks to Java Methods.

In the context of computers, message passing refers to interprocess communication. It is a type of communication used in both parallel and object-oriented programming.

- Pillars of OOPs
    - **Abstraction:** In Java, abstraction is the practice of only displaying to the user the

functionality and features that are truly necessary. The user is not shown the non-essential implementation details.

- **Encapsulation:** A key idea in object-oriented programming (OOP) is encapsulation, which in Java refers to the grouping of data and methods that manipulate that data into a single unit known as a class.
- **Inheritance:** An essential component of OOP is inheritance. It is the Java method to permit a class to inherit the properties and methods of another class.
- **Polymorphism:** Polymorphism allows us to do a same action in a variety of ways.
  - **Compile-time polymorphism:** Another name for it is static polymorphism. Operator overloading or function overloading are the methods used to achieve this kind of polymorphism.
  - **Runtime polymorphism:** Another name for it is Dynamic Method Dispatch. It is a procedure wherein a runtime function call to the overridden method is handled. Method overriding is used to achieve this kind of polymorphism.

POJO

The acronym for Plain Old Java Object is POJO. It is a typical Java object that has no special limitations other than those set forth in the Java Language Specification.

It also doesn't need to be on a classpath. POJOs are used to improve a program's readability and reusability.

Characteristics of POJO

Extend prespecified classes:

Ex: public class GFG extends javax.servlet.http.HttpServlet { ... } is not a POJO

class.

Implement prespecified interfaces:

Ex: public class Bar implements javax.ejb.EntityBean { … } is not a POJO class.

Contain prespecified annotations:

Ex: @javax.persistence.Entity public class Baz { … } is not a POJO class.

Strings

Strings are an object type that may hold characters for values. In Java, each character is saved using UTF 16-bit encoding, which consists of 16 bits. In Java, a string functions just like an array of characters.

Example

String name = "SLA";

Software Testing and QA Architect Syllabus PDF

Understanding of Manual Testing

The term "manual testing" refers to a method of testing software where a QA examines it by hand to find errors.

To accomplish this, QAs adhere to a documented test plan that outlines a number of distinct test scenarios. Analyzing the online or mobile application's performance from the standpoint of the end user is a must for quality assurance.

Principles of Manual Testing

The following are the seven core principles of software testing.

- Testing demonstrates the existence of flaws rather than their lack.
- It is not possible to do exhaustive testing.

- Early testing saves money and time.
- Defects tend to group together.
- Continue going over your tests and making changes or additions to your scenarios.
- Context determines what is tested.
- Error-free logic is false

Test Case Creation

In software testing, a test case is a set of steps needed to confirm a certain feature or operation. The test case describes the procedures, information, preconditions, and postconditions required to validate a feature.

Writing Test Cases with the Intention of Testing Software

- to verify particular software features and functionalities.
- to assist testers with their daily practical work.
- to keep track of all the actions executed so that they can be reviewed if a bug arises.
- to give future testers and projects a plan so they won't have to start from scratch.
- to help in the early detection of design flaws and usability problems.
- to facilitate the short learning curve for new developers and testers, even if they join in the middle of a running project.

Standard Test Case Format

- Test Case ID
- Test Scenario
- Test Steps
- Prerequisites
- Test Data
- Expected/Intended Results
- Actual Results
- Test Status – Pass/Fail

When creating test cases, don't forget to incorporate:

- A fair explanation of the prerequisite
- An explanation of the testing procedure

- Any relevant files or attachments that testers will need
- Information about the testing configuration, such as the program version being tested, data points, operating system, hardware, security clearance, date, time, and requirements.
- Substitute for requirements, if any

SDLC and STLC

Software engineering includes two key components:

SDLC: The Software Development Life Cycle (SDLC) and the Software Testing Life Cycle (STLC). The six phases of the Software Development Life Cycle (SDLC) include the following:

- Requirement Analysis
- Design
- Development
- Testing
- Deployment
- Maintenance.

STLC: It is a part of SDLC. Its scope is narrowly focused, with particular attention to the SDLC testing phase. Through numerous testing procedures, the Software Testing Life Cycle (STLC) aims to guarantee the software's quality, functionality, and dependability.

There are five phases in STLC:

- **Requirement Analysis:** To determine testable aspects, feature requirements gathered throughout the SDLC process are assessed during this phase.
- **Test Planning:** A test plan document serves as an outline of the test strategy. This plan outlines the necessary instruments, the testing procedure, and roles and duties.
- **Test Case Development:** Every test case outlines the expected outcomes, execution

circumstances, test inputs, and processes. Test cases must to be clear, effective, and flexible.

- **Test Environment Setup:** Testing environments are deployed and setup at this phase. A range of testing tools, such as TestComplete, Selenium, Appium, or Katalon Studio, may be used in this step.
- **Test Execution:** In this stage, features are tested using the pre-defined test cases in the deployed environment. Test results are collected and compared to expectations in order to provide updates to development teams.
- **Test Cycle Closure:** A test result report is created during this final round of the STLC. The whole testing procedure should be outlined in this report, together with comparisons between the expected and actual outcomes.

Types of Testing

There are three primary manual testing types:

White Box Testing

White box testing is a type of software testing where an application's core operations and structure are tested. With access to the source code, the tester creates test cases that allow the software's correctness to be confirmed at the code level.

Advantages of White Box Testing:

- Thorough Testing
- Code Optimization
- Early Detection of Defects
- Integration with SDLC
- Detection of Complex Defects.

Black Box Testing

Black-box testing is a kind of software testing where the tester concentrates on validating the functionality according to the given specifications or requirements rather than worrying about the internal knowledge or implementation specifics of

the program.

Software Testing and QA Architect Interview Questions

Advantages of Black Box Testing:

- The tester does not require additional functional knowledge or programming abilities.
- When implementing the tests in the broader system, it works well.
- Tests are carried out with the user's or customer's perspective in mind.
- Test instances can be readily repeated.
- It is employed to identify any conflicts and ambiguities in the functional specifications.

Types of Black Box Testing:

- **Functional Testing:** A sort of software testing known as functional testing involves comparing the system's operation to the functional requirements and specifications.
    - **Unit Testing:** One way to test individual software modules or components is through unit testing.
    - **Integration Testing:** One way to verify how various software application units or components interact with one another is through integration testing.

Types of integration testing:

- **Incremental Testing:** One testing strategy that is frequently applied in the software industry is incremental testing, which is carried out at the testing stage of integration testing, which comes after unit testing.
- **Non-incemental Testing:** Big-bang testing, or non-incremental testing, is a software testing methodology in which all system modules are integrated at the same time and the system is tested as a whole.
- **System Testing:** System testing is a subset of

software testing that assesses the general performance and functionality of a whole software solution that is fully integrated.

- **Non-Functional Testing:** Software testing that is done to confirm that the program satisfies its non-functional criteria is known as non-functional testing. It checks to see if the system is acting in accordance with the requirements.
    - **Performance Testing:** Performance testing is a subset of software testing that assesses an application's or system's scalability and performance.
    - **Usability Testing:** Before being released onto the market, potential customers test the software through a variety of techniques known as usability testing.
    - **Compatability Testing:** The goal of compatibility testing is to ensure that generated software applications run properly across a range of hardware platforms, software, networks, and browsers.

Gray Box Testing

Combining elements of both Black Box and White Box software testing methodologies, Gray Box testing is a software testing approach.

Software Tester & QA architect Salary

Advantages of Gray Box Testing:

- Clarity of goals
- Done from a user perspective
- Programming expertise is not required
- Non-intrusive
- Improved quality of software.

Levels of Testing

The many stages or phases of software testing during its development cycle are referred to as testing levels, or levels of testing.

This concept's primary notion is that distinct

software functionality is tested at each stage, improving quality assurance and reducing the likelihood of problems.

Software testing typically occurs at four stages, which are listed below:

- **Unit Testing:** Unit testing is carried out at the code level, where each component is examined separately to verify its objectivity and assess its functioning.

First, list and describe the program's expectations, including:

- The application ought to accept two number values.
- The total of these two figures should be the result.
- It should also appropriately handle negative values.

- **Integration Testing:** Software testers can test group units incorporated into a system or subsystems using integration testing; this helps find any faults or problems resulting from coding errors or module integrations. Automation of integration testing is feasible.
- **System Testing:** System testing is carried out in an integrated environment that includes the entire application, and each component is evaluated in relation to particular business needs. Automation tools are available for use in system testing.

A tool for no-code test automation called Testsigma, for instance, can finish end-to-end flows for desktop, mobile, and online apps as well as APIs.

- **Acceptance Testing:** Functional and non-functional components of the system, including performance, security, usability, accessibility, compatibility, and dependability, are tested during acceptance testing.

Depending on how sophisticated the system is, automation tools or manual labor can be used.

Software Testing & QA architect Projects

Understanding of Automation Testing

Test automation is the process of automating the execution of tests in software development and quality assurance processes using software tools and scripts.

- Instead of requiring manual labor, it entails writing test cases or scripts that can be executed automatically.
- By decreasing human error, saving time, and enhancing test coverage, test automation seeks to improve testing efficacy and efficiency.

Testing Framework

Any effective automated testing procedure must include testing frameworks. They will give QA teams trying to optimize their agile procedures a better return on investment (ROI) and can cut down on testing and maintenance expenses.

They are necessary for several important reasons that make an automated testing procedure effective:

- Enhanced effectiveness of tests
- Reduced upkeep expenses
- Very little manual labor
- Maximum coverage of tests
- Code reuse

It's crucial to select the appropriate framework while creating a test strategy.

- Linear Automation Framework
- Modular Based Testing Framework
- Library Architecture Testing Framework

- Data-Driven Framework
- Keyword-Driven Framework
- Hybrid Testing Framework

Selenium

An open-source automated testing tool called Selenium is used to test web applications in a variety of browsers.

- Unfortunately, Selenium is limited to testing online applications; it cannot test desktop or mobile applications.
- Software and mobile applications can, however, also be tested using alternative technologies, such as Appium and HP's QTP.

Features of Selenium

- Because it was mostly created in JavaScript, Selenium is simple to use.
- It can test web applications across a variety of browsers, including Firefox, Chrome, Opera, and Safari.
- Numerous programming languages, including Java, Python, Perl, PHP, and Ruby, can be used to write tests.
- Because Selenium is platform-independent, it may be used with Macintosh, Linux, and Windows.
- Selenium can be combined with test management technologies such as JUnit and TestNG.

Locators

A locator helps you find items on a page. The Finding element methods receive this argument as input. For locator advice, such as which to use when and why to declare locators independently of the locating techniques, check out our recommended test procedures.

An overview of the top 8 locators in Selenium may be found here:

- By CSS ID: find_element_by_id

- By CSS class name: find_element_by_class_name
- By name attribute: find_element_by_name
- By DOM structure or Xpath: find_element_by_xpath
- by tagName: find_element_by_tag_name()
- By link text: find_element_by_link_text
- By partial link text: find_element_by_partial_link_text
- By HTML tag name: find_element_by_tag_name

Conclusion

We hope this software testing and QA architect tutorial will be helpful in understanding all about software testing. Gain expertise by enrolling in our **Software Testing and QA Architect training in Chennai.**

Share on your Social Media

Softlogic Academy

Softlogic Systems

KK Nagar [Corporate Office]

No.10, PT Rajan Salai, K.K. Nagar, Chennai – 600 078.
Landmark: Karnataka Bank Building
Phone: +91 86818 84318
Email: enquiry@softlogicsys.in
Map: Google Maps Link

Navigation

About Us
Blog Posts
Careers
Contact
Placement Training
Corporate Training
Hire With Us
Job Seekers
SLA's Recently Placed Students
Reviews
Sitemap

Important Links

OMR

No. E1-A10, RTS Food Street
92, Rajiv Gandhi Salai (OMR),
Navalur, Chennai - 600 130.
Landmark: Adj. to AGS Cinemas
Phone: +91 89256 88858
Email: info@softlogicsys.in
Map: Google Maps Link

Disclaimer

Privacy Policy

Terms and Conditions

Courses

- Python
- Software Testing
- Full Stack Developer
- Java
- Power BI
- Clinical SAS
- Data Science
- Embedded
- Cloud Computing
- Hardware and Networking
- VBA Macros
- Mobile App Development
- DevOps

Social Media Links

Review Sources

- Google
- Trustpilot
- Glassdoor
- Mouthshut
- Sulekha
- Justdial
- Ambitionbox
- Indeed
- Software Suggest
- Sitejabber