

GENERIC PRODUCT TESTING

- ❖ **Product testing** is a research methodology allowing businesses to collect qualitative and quantitative information about consumer's potential consumption/usage behavior, preferences, and reactions on a product.
- ❖ A process, to evaluate the functionality of a software application with an intent to find whether the developed software met the specified requirements or not and to identify the defects to ensure that the product is defect-free in order to produce the quality product.

INDUSTRY AND ORGANISATION

HORIZONTALS AND VERTICALS

In the IT industry, we call the technical skills as horizontals and domain knowledge (about the client's business) as the verticals. Consider you are a Java developer and you are working on a ticket reservation system for your client, then your horizontal is Java delivery and your vertical is Travel.

- ❖ Horizontal SaaS tends to be the more mature model on the market, the model having been around for well over a decade. This type of SaaS focuses on a category of software, catering to a business need.
- ❖ In contrast to the horizontal, vertical SaaS produce software that is targeted to a very particular industry. This is a more recent trend in SaaS so not as mature as the horizontal market. In fact, there are several commentators who feel that there may be more opportunities for new vertical SaaS due to them being a more recent phenomenon.

WHAT IS TESTING IN ORGANISATIONAL PERSPECTIVE ?

Testing is a crucial part of any SDLC and it occupies roughly 30-40% of the development lifecycle. Still around 20 years ago, testing was not organized as it is today where dedicated teams are allocated for this job specifically.

TYPES OF TESTING

MANUAL TESTING

Manual Testing is a software testing process in which test cases are executed manually according to the end user's perspective. It ensures whether the application is working, as mentioned in the requirement document or not. Test cases are planned and implemented to complete almost 100 percent of the software application. Test case reports are also generated manually. Manual Testing is one of the most fundamental testing processes. It can find both visible and hidden defects of the software. The difference between expected output and output, given by the software, is defined as a defect.

Manual Testing methods are-

- ❖ **Black Box Testing** is a software testing method in which the internal structure/ design/ implementation of the item being tested is not known to the tester. No knowledge of implementation is needed.
- ❖ **Grey Box Testing** is a software testing technique performed with limited information about the internal functionality of the system.
- ❖ Grey Box testers have access to the detailed design documents along with information about requirements.
- ❖ **White Box Testing** is a software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester.
- ❖ Knowledge of implementation is required.

PROS

- Applications with short life cycles.
- Applications that have GUIs that constantly changes
- It requires less time and expense to begin productive manual testing.
- Automation cannot replace human intuition, inference, and inductive reasoning.
- We can easily update our test case according to project movement.

CONS

- Load testing and performance testing is not possible in manual testing.
- Running tests manually is a very time-consuming job.
- Regression Test cases are time-consuming if it is manual testing.

AUTOMATION TESTING

Automation Testing is a Software testing that is done through an automation tool. Basically less time is needed in exploratory tests and more time is needed in maintaining test scripts while increasing overall test coverage. Test automation is used to automate repetitive tasks and other testing tasks that are difficult to perform manually.

For automation engineers, robust support for scripting languages, integrations with CI systems, and the ability to scale tests easily is very important.

Automation Testing methods are-

- ❖ **Functional Testing** is a type of testing which verifies that each **function** of the software application operates in conformance with the requirement specification. This testing mainly involves black box testing, and it is not concerned about the source code of the application.

- ❖ **Non-Functional Testing** is a type of testing to check non-functional aspects (performance, usability, reliability, etc.) of a software application. It is explicitly designed to test the readiness of a system as per nonfunctional parameters which are never addressed by functional testing.

Some Automation Testing tools are - ***Selenium, JMeter, Loadrunner, Postman***

PROS

- Automation testing is essential for CI/CD.
- Automating Smoke, Sanity and Regression Tests frees us the time of testers, so they can focus more on exploratory testing of new features.
- Automated checks can take a while to script. However, when we execute them they are generally fast and can go through various steps much quicker than a human.
- Automated checks are a great way of confirming that the application still functions properly after changes made to it.

CONS

- A false sense of quality in terms of UI or System level.
- Automated checks can fail due to many factors. If automated checks keep failing due to issues other than genuine bugs, they can raise false alarms.
- Automated checks are short-lived. If the regression packs are not kept up to date, you start seeing all kinds of failures.

SOFTWARE DEVELOPMENT LIFE CYCLE

Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high-quality software. The SDLC aims to produce software that meets or exceeds customer expectations, reaches completion within times and cost estimates.

PHASES OF SDLC

□ PLANNING AND REQUIREMENT ANALYSIS

Requirement analysis is the most important and fundamental stage in SDLC. It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry. This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational and technical areas.

Planning for the quality assurance requirements and identification of the risks associated with the project is also done in the planning stage. The outcome of the technical feasibility study is to define the various technical approaches that can be followed to implement the project successfully with minimum risks.

□ DEFINING REQUIREMENTS

Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysts. This is done through an **SRS (Software Requirement Specification)** document which consists of all the product requirements to be designed and developed during the project life cycle.

□ DESIGNING THE PRODUCT ARCHITECTURE

SRS is the reference for product architects to come out with the best architecture for the product to be developed. Based on the requirements specified in SRS, usually, more than one design approach for the product architecture is proposed and documented in a **DDS - Design Document Specification**.

This DDS is reviewed by all the important stakeholders and based on various parameters as risk assessment, product robustness, design modularity, budget and time constraints, the best design approach is selected for the product.

A design approach clearly defines all the architectural modules of the product along with its communication and data flow representation with the external and third party modules (if any). The internal design of all the modules of the proposed architecture should be clearly defined with the minutest of the details in DDS.

□ BUILDING OR DEVELOPING THE PRODUCT

In this stage of SDLC, the actual development starts and the product is built. The programming code is generated as per DDS during this stage. If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle.

Developers must follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers, etc. are used to generate the code. Different high-level programming languages are used for coding. The programming language is chosen with respect to the type of software being developed.

□ TESTING THE PRODUCT

This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC. However, this stage refers to the testing only stage of the product where product defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.

□ DEPLOYMENT IN THE MARKET AND MAINTENANCE

Once the product is tested and ready to be deployed it is released formally in the appropriate market. Sometimes product deployment happens in stages as per the business strategy of that organization. The product may first be released in a limited segment and tested in the real business environment (**UAT- User Acceptance Testing**).

Then based on the feedback, the product may be released as it is or with suggested enhancements in the targeting market segment. After the product is released in the market, its maintenance is done for the existing customer base.

SDLC MODELS

There are various software development life cycle models defined and designed which are followed during the software development process. These models are also referred to as “**Software Development Process Models**”. Each process model follows a Series of steps unique to its type to ensure success in the process of software development.

Following are the most important and popular SDLC models followed in the industry –

- Waterfall Model
- V-Model
- Spiral Model
- Agile Model
- DevOps

WATERFALL MODEL

Waterfall approach was the first SDLC Model to be used widely in Software Engineering to ensure success of the project. In this approach, the whole process of software development is divided into separate phases. In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

All these phases are cascaded to each other in which progress is seen as flowing steadily downwards (like a waterfall) through the phases. The next phase is started only after the defined set of goals is achieved for the previous phase and it is signed off, hence the name "Waterfall Model". In this model, phases do not overlap.

PROS

- Simple and easy to understand and use.
- Clearly defined stages.
- Phases are processed and completed one at a time.
- Well understood milestones.
- Process and results are well documented.

CONS

- No working software is produced until late during the life cycle.
- Not a good model for complex and object-oriented projects.

- It is difficult to measure progress within stages.
- Cannot accommodate changing requirements.
-

V – MODEL

The V-model is an SDLC model where the execution of processes happens in a sequential manner in a V-shape. It is also known as the ***Verification and Validation model***. The V-Model is an extension of the waterfall model and is based on the association of a testing phase for each corresponding development stage. This means that for every single phase in the development cycle, there is a directly associated testing phase. This is a highly-disciplined model and the next phase starts only after completion of the previous phase.

Requirements have to be very clear before the project starts because it is usually expensive to go back and make changes. This model is used in the medical development field, as it is strictly a disciplined domain.

PROS

- This is a highly-disciplined model and Phases are completed one at a time.
- Works well for smaller projects where requirements are very well understood.
- Simple and easy to understand and use.
- Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.

CONS

- High risk and uncertainty.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing.

AGILE MODEL

Agile SDLC model is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product. Agile Methods break the product into small incremental builds. These builds are provided in iterations. Each iteration typically lasts from about one to three weeks. Every iteration involves cross-functional teams working simultaneously on various areas like –

- Planning

- Requirements Analysis
- Design
- Coding
- Unit Testing and Acceptance Testing

The most important features of Agile Model is:

- ❖ **Scrum** refers to a framework that makes for effective collaborations among teams that are working on complex products. It is a collection of meetings, roles and tools that work together to help teams to better structure and manage their workload.
- ❖ **Sprint** is a repeatable fixed time-box during which a "Finished" product of the highest possible value is created. Sprint lies at the core of the Sprint agile methodology and can be thought of as an event that wraps all other Scrum events.

PROS

- Is a very realistic approach to software development.
- Delivers early partial working solutions.
- Good model for environments that change steadily.
- Little or no planning required.
- Gives flexibility to developers.

CONS

- More risk of sustainability, maintainability and extensibility.
- An overall plan, an agile leader and agile PM practice is a must without which it will not work.
- Depends heavily on customer interaction, so if the customer is not clear, the team can be driven in the wrong direction.
- Transfer of technology to new team members may be quite challenging due to lack of documentation.

SPIRAL MODEL

The spiral model combines the idea of iterative development with the systematic, controlled aspects of the waterfall model. This Spiral model is a combination of iterative development process model and sequential linear development model i.e. the waterfall model with a very high emphasis on risk analysis. It allows incremental releases of the product or incremental refinement through each iteration around the spiral.

PROS

- Requirements can be captured more accurately.
- Users see the system early.
- Development can be divided into smaller parts and the risky parts can be developed earlier which helps in better risk management.
- Allows extensive use of prototypes.

CONS

- Management is more complex.
- Not suitable for small or low-risk projects and could be expensive for small projects.
- Process is complex
- Spiral may go on indefinitely.
- Large numbers of intermediate stages require excessive documentation.

DEVOPS

DevOps is a set of practices that combines software development and IT operations. It aims to shorten the systems development life cycle and provide continuous delivery with high software quality. DevOps is complementary with Agile software development; several DevOps aspects came from Agile methodology.

PROS

- Great approach for quick development and deployment of applications.
- Responds faster to the market changes to improve business growth.
- Escalate business profits by decreasing software delivery time and transportation costs.
- Improves customer experience and satisfaction.
- Reduce the implementation time of new services.
- Increase the productivity of the business and IT teams.

CONS

- DevOps professional or expert developers are less available.
- Adopting new DevOps technology into the industries is hard to cope in a short span of time.
- Developing with DevOps Is expensive. Skilled consultants would charge more.
- Lack of DevOps knowledge could be a problem in the continuous integration of automation projects.

SOFTWARE TESTING LIFE CYCLE

Software Testing Life Cycle (STLC) is a sequence of different activities performed by the testing team to ensure the quality of the software or the product. STLC is an integral part of the Software Development Life Cycle (**SDLC**). But, STLC deals only with the testing phases. STLC starts as soon as requirements are defined or **SRD (Software Requirement Document)** is shared by stakeholders.

STLC provides a step-by-step process to ensure quality software. In the early stage of STLC, while the software or the product is developing, the tester can analyze and define the scope of testing, entry and exit criteria and also the Test Cases. It helps to reduce the test cycle time along with better quality. As soon as the development phase is over, the testers are ready with test cases and start with execution. This helps to find bugs in the initial phase.

PHASES OF STLC

❑ REQUIREMENT ANALYSIS

During this phase, feature requirements collected in the SDLC process are evaluated to identify testable aspects. If necessary, testing teams may need to consult with stakeholders to clarify requirements. These requirements can either be functional or non-functional, defining what a feature can do or its characteristics respectively.

❑ TEST PLANNING

During this phase, the test strategy is outlined in a test plan document. This strategy includes tools needed, testing steps, and roles and responsibilities. Part of determining this strategy is a risk and cost analysis and an estimated timeline for testing. The Test Strategy Document is also created.

❑ TEST CASE DEVELOPMENT

During this phase, test cases are created. Each case defines test inputs, procedures, execution conditions, and anticipated results. Test cases should be transparent, efficient, and adaptable. Once all test cases are created, test coverage should be 100%. Any necessary automation scripts are also created during this phase.

❑ TEST ENVIRONMENT SETUP

During this phase, testing environments are configured and deployed. This phase may include a variety of testing tools, including TestComplete, Selenium, Appium, or Katalon Studio. Sometimes, this phase also includes setting up test servers. Once environments are deployed, smoke tests are performed to ensure that environments are working as expected with all intended functionality.

❑ TEST EXECUTION PHASE

During this phase, features are tested in the deployed environment, using the established test cases. Expected test results are compared to actual and results are gathered to report back to development teams.

□ TEST CYCLE CLOSURE

This is the last phase of the STLC, during which a test result report is prepared. This report should summarize the entire testing process and provide comparisons between expected results and actual. These comparisons include objectives met, time taken, total costs, test coverage, and any defects found.

REQUIREMENT TRACEABILITY MATRIX

Requirement Traceability Matrix (RTM) is a document that maps and traces user requirements with test cases. It captures all requirements proposed by the client and requirement traceability in a single document, delivered at the conclusion of the Software development life cycle. The main purpose of the Requirement Traceability Matrix is to validate that all requirements are checked via test cases such that no functionality is unchecked during Software testing.

DEFECT LIFE CYCLE

Defect Life Cycle, also known as Bug Life Cycle, is the journey of a defect, the cycle which a defect goes through during its lifetime. It varies from organization to organization and also from project to project, as it is governed by the software testing process and also depends upon the tools used. The different states in a defect life cycle are given below,

- **New** – Potential defect that is raised and yet to be validated.
- **Assigned** – Assigned against a development team to be addressed.
- **Active** – The Defect is being addressed by the developer and investigation is under progress. At this stage, there are two possible outcomes – Deferred or Rejected.
- **Test / Fixed / Ready for Retest** – The Defect is fixed and ready for testing.
- **Closed** – The final state of the defect that can be closed after the QA retesting or can be closed if the defect is duplicate or considered as NOT a defect.

SEVERITY & PRIORITY IN TESTING

□ SEVERITY

Severity is defined as the degree of impact a Defect has on the development or operation of a component application being tested. A higher effect on the system functionality will lead to

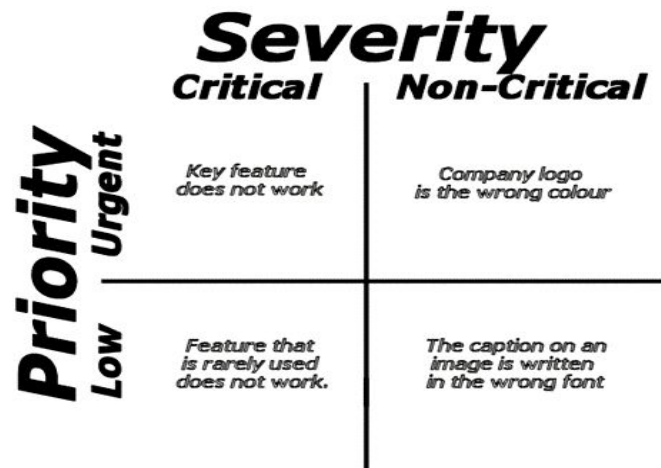
the assignment of higher severity to the bug. Quality Assurance engineer usually determines the severity level of the defect.

Severity is of 5 types: **Critical**, **Major**, **Moderate**, **Minor**, and **Cosmetic**.

□ PRIORITY

Priority is defined as the order in which a defect should be fixed. The higher the priority the sooner the defect should be resolved. Defects that leave the software system unusable are given higher priority over defects that cause a small functionality of the software to fail.

Priority is of 3 types: **Low**, **Medium**, and **High**.



TEST PLAN

A **Test Plan** is a detailed document that describes the test strategy, objectives, schedule, estimation, deliverables and resources required to perform testing for a software product. Test Plan helps us determine the effort needed to validate the quality of the application under test. The test plan serves as a blueprint to conduct software testing activities as a defined process which is minutely monitored and controlled by the test manager.

HOW TO WRITE A TEST PLAN AS PER IEEE 829

STEP 1: ANALYZE THE PRODUCT

Thorough research of clients and the end-users to know their needs and expectations from the application is essential.

Some important questions to be answered are:

- Who will use the website?
- What is it used for?
- How will it work?
- What are software/ hardware that the product uses?

STEP 2: DEFINE THE TEST STRATEGY

Test Strategy is a **critical step** in making a Test Plan in Software Testing. A **Test Strategy Document** is a high-level document, which is usually developed by the Test Manager. This document defines:

- The project's **testing objectives** and the means to achieve them
- Determines testing **effort** and **costs**

Test Strategy Document includes

- Scope of Testing
- Identify testing types
- Document risk and issues
- Create test logistics

STEP 3: DEFINE TEST OBJECTIVES

Test Objective is the overall goal and achievement of the test execution. The objective of the testing is to find as many software defects as possible; ensure that the software under test is **bug-free** before release. To define the test objectives, the following steps must be done

- List all the software features (functionality, performance, GUI...) which may need to test.
- Define the target or the goal of the test based on the above features.

STEP 4: DEFINE TEST CRITERIA

Entry Criteria for testing can be defined as "Specific conditions or on-going activities that must be present before a process can begin." The STLC specifies the entry criteria required during each testing phase. It also defines the time interval or the expected amount of lead-time to make the entry criteria item available to the process.

The inputs can be divided into two categories

- received from development.
- produced from the test phases at the end of STLC.

Exit Criteria in testing are often viewed as a single document commemorating the end of a life cycle phase. It can be defined as "The specific conditions or on-going activities that should be fulfilled before completing the software testing life cycle. STLC specifies which exit criteria are required at each testing phase". The exit criteria can identify the intermediate deliverables and enable you to track them as independent events.

STEP 5: RESOURCE PLANNING

The resource plan is a **detailed summary** of all types of resources required to complete project task. The resource could be human, equipment and materials needed to complete a project.

Resource planning is an important factor in test planning because it helps in determining the number of resources required for the project. Therefore, the Test Manager can make the correct schedule & estimation for the project.

STEP 6: PLAN TEST ENVIRONMENT

A testing environment is a setup of software and hardware on which the testing team is going to execute test cases. The test environment consists of **real business** and **user** environment, as well as physical environments, such as server, front-end running environment.

Some important questions to be answered are:

- What is the maximum user connection which this website can handle at the same time?
- What are software/hardware requirements to install this website?
- Does the user's computer need any particular setting to browse the website?

STEP 7: SCHEDULE & ESTIMATION

Making a schedule is a common term in project management. By creating a solid schedule in the Test Planning, the Test Manager can use it as a tool for monitoring the project progress, control the cost overruns. To create the project schedule, the Test Manager needs several types of input as below:

- **Employee and project deadline:** The working days, the project deadline, resource availability are the factors which affect the schedule
- **Project estimation:** Based on the estimation, the Test Manager knows how long it will take to complete the project.
- **Project Risk:** Understanding the risk helps Test Manager add enough extra time to the project schedule to deal with the risks.

STEP 8: TEST DELIVERABLES

Test Deliverables is a list of all the documents, tools and other components that have to be developed and maintained in support of the testing effort.

Test deliverables provided **before** the testing phase:

- Test plans document,
- Test case documents,
- Test Design specifications.

Test deliverables provided **during** the testing phase:

- Test Scripts,
- Simulators,
- Test Data,
- Test Traceability Matrix,
- Error logs and execution logs.

Test deliverables provided **after** the testing phase:

- Test Results/reports,
- Defect Report,

- Installation/ Test procedures guidelines,
- Release notes.

TYPES OF SOFTWARE TESTING

UNIT TESTING

- Testing of an individual software component or module is termed as Unit Testing.
- It is typically done by the programmer and not by testers, as it requires detailed knowledge of the internal program design and code.
- It may also require developing test driver modules or test harnesses.

SMOKE TESTING

- Whenever a new build is provided by the development team then the Software Testing team validates the build and ensures that no major issue exists.
- The testing team ensures that the build is stable and a detailed level of testing is carried out further.
- Smoke Testing checks that no defect exists in the build which will prevent the testing team to test the application in detail.
- If testers find that the major critical functionality is broken down at the initial stage itself then the testing team can reject the build and inform accordingly to the development team.
- Smoke Testing is carried out to a detailed level of any Functional or Regression Testing.

SANITY TESTING

- Sanity testing is done to determine if a new software version is performing well enough to accept it for a major testing effort or not.
- If an application is crashing for the initial use then the system is not stable enough for further testing. Hence a build or an application is assigned to fix it.

INTEGRATION TESTING

- Testing of all integrated modules to verify the combined functionality after integration is termed as Integration Testing.
- Modules are typically code modules, individual applications, client and server applications on a network, etc.
- This type of testing is especially relevant to client/server and distributed systems.

REGRESSION TESTING

- Testing an application as a whole for the modification in any module or functionality is termed as Regression Testing.
- It is difficult to test the entire system in Regression Testing, so typically Automation Testing Tools are used for these types of testing.

ACCEPTANCE TESTING

- An Acceptance Test is performed by the client and verifies whether the end to end the flow of the system is as per the business requirements or not and if it is as per the needs of the end-user.
- The client accepts the software only when all the features and functionalities work as expected.

TEST AUTOMATION FRAMEWORKS

A testing framework is a set of guidelines or rules used for creating and designing test cases. A framework consists of a combination of practices and tools that are designed to help QA professionals test more efficiently.

These guidelines could include coding standards, test-data handling methods, object repositories, processes for storing test results, or information on how to access external resources.

ADVANTAGES OF TEST AUTOMATION TOOLS

- Improved test efficiency
- Lower maintenance costs
- Minimal manual intervention
- Maximum test coverage
- Reusability of code

TEST AUTOMATION TOOLS

The current test automation trends have increased in applying artificial intelligence and machine learning (AI/ML) to offer advanced capabilities for test optimization, intelligent test generation, execution, and reporting.

Some important test automation tools are

WEB TESTING TOOLS

SELENIUM

- Selenium is the household name when it comes to testing automation.
- It is considered the industry standard for user interface automation testing of Web applications.
- For developers and testers who have experience and skills in programming and scripting,
- Selenium offers flexibility that is unseen in many other test automation tools and frameworks.

- Users can write test scripts in many different languages (such as Java, Groovy, Python, C#, PHP, Ruby, and Perl) that run on multiple system environments (Windows, Mac, Linux) and browsers (Chrome, Firefox, IE, and Headless browsers).
- Selenium is a Test Automation Tools with added benefits of Test Management Tools.

API TESTING TOOLS

POSTMAN

Postman is an automation tool designed for API testing. Users can install this tool as a browser extension or a desktop application on Mac, Linux, Windows. It is popular not only among testers for API test automation but also developers who use the tool to develop and test APIs.

Some highlights of the tool:

- Comprehensive feature set for designing, debugging, testing, documenting, and publishing APIs
- Friendly and easy-to-use user interface
- Supporting both automated and exploratory testing

SECURITY TESTING TOOLS

OWASP ZAP

- The OWASP Zed Attack Proxy (ZAP) is one of the world's most popular web application security testing tools. It is made available for free as an open-source project and is contributed to and maintained by OWASP.
- The OWASP ZAP tool can be used during web application development by web developers or by experienced security experts during penetration tests to assess web applications for vulnerabilities.

PERFORMANCE TESTING TOOLS

JMETER

- JMeter is a software that can perform load tests, performance-oriented business (functional) tests, regression tests, etc., on different protocols or technologies.
- JMeter is a Java desktop application with a graphical interface that uses the Swing graphical API. It can therefore run on any environment/workstation that accepts a Java virtual machine, for example – Windows, Linux, Mac, etc.

Some features of JMeter are-

- It has a simple and intuitive GUI.
- JMeter can conduct load and performance test for many different server types.

- JMeter stores its test plans in XML format. This means you can generate a test plan using a text editor.
- Its full multi-threading framework allows concurrent sampling by many threads and simultaneous sampling of different functions by separate thread groups.

TEST MANAGEMENT TOOLS

- Test management tools are used to store information on how testing is to be done, plan testing activities and report the status of quality assurance activities.
- The tools have different approaches to testing and thus have different sets of features. Generally, they are used to maintain and plan manual testing, run or gather execution data from automated tests, manage multiple environments and enter information about found defects.
- Test management tools offer the prospect of streamlining the testing process and allow quick access to data analysis, collaborative tools and easy communication across multiple project teams.
- Many test management tools incorporate requirements management capabilities to streamline test case design from the requirements.
- Tracking of defects and project tasks are done within one application to further simplify the testing.

Example: **Zephyr**, **QTest**, etc.

REQUIREMENT GATHERING TOOLS

REQUIREMENTS ELICITATION

- It is related to the various ways used to gain knowledge about the project domain and requirements.
- The various sources of domain knowledge include customers, business manuals, the existing software of the same type, standards and other stakeholders of the project.
- The techniques used for requirements elicitation include **interviews**, **brainstorming**, **task analysis**, **Delphi technique**, **prototyping**, etc.

REQUIREMENTS SPECIFICATION

- This activity is used to produce formal software requirement models.
- All the requirements including the functional as well as the non-functional requirements and the constraints are specified by these models in totality.
- During specification, more knowledge about the problem may be required which can again trigger the elicitation process.
- The models used at this stage include **ER diagrams**, **data flow diagrams(DFDs)**, **function decomposition diagrams(FDDs)**, **data dictionaries**, etc.

REQUIREMENTS VERIFICATION AND VALIDATION

Verification: It refers to the set of tasks that ensures that the software correctly implements a specific function.

Validation: It refers to a different set of tasks that ensures that the software that has been built is traceable to customer requirements.

REQUIREMENTS MANAGEMENT

- Requirement management is the process of analyzing, documenting, tracking, prioritizing and agreeing on the requirement and controlling the communication to relevant stakeholders.
- This stage takes care of the changing nature of requirements. It should be ensured that the SRS is as modifiable as possible so as to incorporate changes in requirements specified by the end-users at later stages too.
- Being able to modify the software as per requirements in a systematic and controlled manner is an extremely important part of the requirements engineering process.

Some widely used requirements gathering tools are ***Modern Requirements, Jama, Visure.***

DEFECT MANAGEMENT TOOLS

- Defect management is a process to identify the defect of the software. The Development team needs a defect management tool so that they can find defect easily and at a very early stage of the process because as soon as the defect is detected the cost of fixing it will be low, but if the defect is detected in the later stage of development process then the cost of fixing that defect will be more.
- Defect Management Tools record all the defect of the software these records can be seen later as well if you want to review or want to check how you have fixed that. These records can be used as the knowledge-base for future reference. Defect management tools also provide a basis for Project Management.
- Project management tools are aids to assist an individual or team to effectively organize work and manage projects and tasks. Project management tools are made to be completely customizable so they can fit the needs of teams of different sizes and with different goals.

These goals include planning/scheduling, collaboration, documentation, evaluation.

Most widely used Defect Management Tools are **Jira and Bugzilla.**

OPEN SOURCE VS LICENSED SOFTWARE

All the tools discussed above can be divided further into 2 more categories:

- Open Source Software
- Licensed Software

OPEN SOURCE SOFTWARE

Open-source software (OSS) is a type of computer software in which source code is released under a license in which the copyright holder grants users the right to use, study, change, and distribute the software to anyone and for any purpose. Open-source software may

be developed in a collaborative public manner. Open-source software is a prominent example of open collaboration.

LICENSED SOFTWARE

A software license is a document that provides legally binding guidelines for the use and distribution of software.

Software licenses typically provide end-users with the right to one or more copies of the software without violating copyrights. The license also defines the responsibilities of the parties entering into the license agreement and may impose restrictions on how the software can be used. Software licensing terms and conditions usually include fair use of the software, the limitations of liability, warranties and disclaimers and protections if the software or its use infringes on the intellectual property rights of others.

WHY TEST AUTOMATION IS NEEDED IN INDUSTRY

The following reason support why test automation is crucial for testing-

- Ready for 24/7 execution,
- Quicker and more manageable Regression Testing,
- Reusability ,
- Allows integration of reporting frameworks to the automation test cases,
- Supports tests in multiple platforms simultaneously,
- Time saved,
- Better utilisation of manpower hence business cost reduced,
- Allows shift left testing,
- Ensures the quality and quantity of test cases by maximizing test coverage,
- Maximises Return on Investment (**ROI**) by deploying bugless applications,
- Supports cross browser testing,
- Quicker Smoke tests,
- Quicker Data-Driven tests,
- Supports distribution testing,
- Increases scalability,
- Check complex and lengthy scenarios.

PROCESS IN TEST AUTOMATION

- Convince management and team on the benefits of Test Automation.
- Find automation experts and seek their help and insights on the project.
- Choose the most suitable testing tool for the project.
- Analyze the area to be automated.
- Training the team on the testing tool for efficient results. Addition training on CICD and defect tools can also be done.
- Create or use a previous Automation Framework.
- Set an execution plan.
- Writing the scripts.
- Documenting a report on the work done on the project.

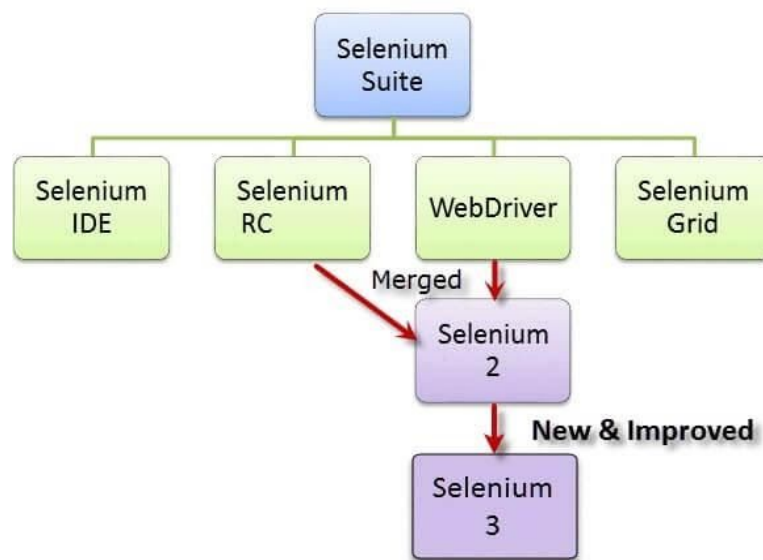
- Maintaining the script for future use.

SELENIUM

SELENIUM is a free (open-source) automated testing framework used to validate web applications across different browsers and platforms. You can use multiple programming languages like Java, C#, Python etc to create Selenium Test Scripts. Testing done using the Selenium tool is usually referred to as Selenium Testing.

Selenium Software is not just a single tool but a suite of software, each piece catering to different testing needs of an organization.

1. Selenium Integrated Development Environment (IDE)
2. Selenium Remote Control (RC)
3. WebDriver
4. Selenium Grid



HOW TO DOWNLOAD & INSTALL SELENIUM WEBDRIVER

1. Install Java on your computer
2. Install Eclipse IDE
3. Download the Selenium Java Client Driver
4. Configure Eclipse IDE with WebDriver

- a. Launch the "eclipse.exe" file inside the "eclipse" folder that we extracted in step 2. If you followed step 2 correctly, the executable should be located on C:\eclipse\eclipse.exe.
- b. When asked to select for a workspace.
- c. Create a new project through File > New > Java Project. Name the project as "newproject".
- d. Select New > Package, and name that package as "newpackage".
- e. Create a new Java class under newpackage by right-clicking on it and then selecting- New > Class, and then name it as "MyClass".

Now adding Selenium WebDriver's into Java Build Path

1. Right-click on "newproject" and select **Properties**.
2. On the Properties dialog, click on "Java Build Path".
3. Click on the **Libraries** tab, and then
4. Click on "Add External JARs.."

Once done, click "Apply and Close" button.

GETTING STARTED WITH SELENIUM

IMPORTING PACKAGES

To get started, you need to import following two packages:

1. **org.openqa.selenium.***- contains the WebDriver class needed to instantiate a new browser loaded with a specific driver
2. **org.openqa.selenium.chrome.ChromeDriver** - contains the ChromeDriver class needed to instantiate a Chrome-specific driver onto the browser instantiated by the WebDriver class

INSTANTIATING OBJECTS AND VARIABLES

WebDriver driver = new ChromeDriver();

A ChromeDriver class with no parameters means that the default Chrome profile will be launched by our Java program. The default Chrome profile is similar to launching Chrome in safe mode (no extensions are loaded).

LAUNCHING A BROWSER SESSION

driver.get(baseUrl);

WebDriver's **get()** method is used to launch a new browser session and directs it to the URL that you specify as its parameter.

GET THE ACTUAL PAGE TITLE

String actualTitle = driver.getTitle();

The WebDriver class has the **getTitle()** method that is always used to obtain the page title of the currently loaded page.

TERMINATING THE BROWSER SESSION

```
driver.close();
```

TERMINATING THE ENTIRE PROGRAM

```
System.exit(0);
```

If you use this command without closing all browser windows first, your whole Java program will end while leaving the browser window open.

NAVIGATION

- **Navigate to:** `driver.navigate().to("url");`
- **Forward:** `driver.navigate().forward();`
- **Backward:** `driver.navigate().backward();`
- **Refresh:** `driver.navigate().refresh();`

LOCATORS IN SELENIUM IDE

Locator is a command that tells Selenium IDE which GUI elements it needs to operate on. Identification of correct GUI elements is a prerequisite to creating an automation script. Hence, Selenium provides a number of Locators to precisely locate a GUI element.

Method	Target Syntax	Example
By ID	<code>id= <i>id_of_the_element</i></code>	<code>id=email</code>
By Name	<code>name=<i>name_of_the_element</i></code>	<code>name=userName</code>
By Name Using Filters	<code>name=<i>name_of_the_element</i> filter=<i>value_of_filter</i></code>	<code>name=tripType value=oneway</code>
By Link Text	<code>link=<i>link_text</i></code>	<code>link=REGISTER</code>
Tag and ID	<code>css=<i>tag</i>#<i>id</i></code>	<code>css=input#email</code>
Tag and Class	<code>css=<i>tag</i>.<i>class</i></code>	<code>css=input.inputtext</code>
Tag and Attribute	<code>css=<i>tag</i>[<i>attribute=value</i>]</code>	<code>css=input[name=lastName]</code>
Tag, Class, and Attribute	<code>css=<i>tag</i>.<i>class</i>[<i>attribute=value</i>]</code>	<code>css=input.inputtext[tabindex=1]</code>

FIND ELEMENT(S) IN SELENIUM WEBDRIVER

Interaction with a web page requires a user to locate the web element. Find Element command is used to uniquely identify a web element within the web page. Whereas, Find Elements command is used to uniquely identify the list of web elements within the web page.

DIFFERENCE BETWEEN FIND ELEMENT AND FIND ELEMENTS

FindElement	FindElements
Returns the first most web element if there are multiple web elements found with the same locator.	Returns a list of web elements.
Throws exception NoSuchElementException if there are no elements matching the locator strategy.	Returns an empty list if there are no web elements matching the locator strategy.
It will only find one web element	It will find a collection of elements whose match the locator strategy.
Indexing not applicable.	Each Web element is indexed with a number starting from 0 just like an array

XPATH IN SELENIUM WEBDRIVER

In Selenium automation, if the elements are not found by the general locators like id, class, name, etc. then XPath is used to find an element on the web page .

XPath in Selenium is an XML path used for navigation through the HTML structure of the page. It is a syntax or language for finding any element on a web page using XML path expression. XPath can be used for both HTML and XML documents to find the location of any element on a webpage using HTML DOM structure.

Syntax:- **Xpath=//tagname[@attribute='value']**

TYPES OF XPATH

ABSOLUTE XPATH

- It is the direct way to find the element, but the disadvantage of the absolute XPath is that if there are any changes made in the path of the element then that XPath gets failed.
- The key characteristic of XPath is that it begins with the single forward slash(/) ,which means you can select the element from the root node.

RELATIVE XPATH

- **Relative Xpath** starts from the middle of the HTML DOM structure.
- It starts with double forward slash (//). It can search elements anywhere on the webpage, means no need to write a long xpath and you can start from the middle of HTML DOM structure. Relative Xpath is always preferred as it is not a complete path from the root element.

USING XPATH FOR HANDLING COMPLEX & DYNAMIC ELEMENTS IN SELENIUM

BASIC XPATH

XPath expression selects nodes or list of nodes on the basis of attributes like **ID** , **Name**, **Classname**, etc. from the XML document as illustrated below.

Example- **Xpath=//input[@type='text']**

CONTAINS

Contains() is a method used in XPath expression. It is used when the value of any attribute changes dynamically, for example, login information.

Example- **Xpath=//*[contains(@type,'sub')]**

USING OR & AND

In OR expression, two conditions are used, whether 1st condition OR 2nd condition should be true. It is also applicable if any one condition is true or maybe both. Means any one condition should be true to find the element.

Example- **Xpath=//*[@type='submit' or @name='btnReset']**

MOUSE CLICK & KEYBOARD EVENT

ACTION CLASS IN SELENIUM

Action Class in Selenium is a built-in feature provided by the selenium for handling keyboard and mouse events. These operations from the action class are performed using the advanced user interaction API in Selenium Webdriver.

The following are the most commonly used keyboard and mouse events provided by the Actions class.

Method	Description
clickAndHold()	Clicks (without releasing) at the current mouse location.
contextClick()	Performs a context-click at the current mouse location. (Right Click Mouse Action)
doubleClick()	Performs a double-click at the current mouse location.
keyDown(modifier_key)	Performs a modifier key press. Does not release the modifier key - subsequent interactions may assume it's kept pressed.

	Parameters: modifier_key - any of the modifier keys (Keys.ALT, Keys.SHIFT, or Keys.CONTROL)
keyUp(modifier_key)	Performs a key release. Parameters: modifier_key - any of the modifier keys (Keys.ALT, Keys.SHIFT, or Keys.CONTROL)

Steps involved are-

- Import the **Actions** and **Action** classes.
Import org.openqa.selenium.interactions.Action;
Import org.openqa.selenium.interactions.Actions;
- Instantiate a new Actions object.
Actions builder = new Action(driver);
- Instantiate an Action using the Actions object in step 2.
Action mouseOverHome = builder.moveToElement(link_Home).build();
- Use the perform() method when executing the Action object we designed in Step 3.
mouseOverHome.perform();

SELENIUM WEB ELEMENTS

- Forms are the fundamental web elements to receive information from the website visitors. Web forms have different GUI elements like Text boxes, Password fields, Checkboxes, Radio buttons, dropdowns, file inputs, etc.
- Selenium encapsulates every form element as an object of WebElement. It provides an API to find the elements and take action on them like entering text into text boxes, clicking the buttons, etc. We will see the methods that are available to access each form element.
- We need to import this package to create objects of Web Elements
Import org.openqa.selenium.webElement;
- We need to call the findElement() method available on the WebDriver class and get an object of WebElement.

INPUT BOX

ADDING VALUES IN INPUT BOX

sendKeys() in **Selenium** is a method used to enter editable content in the text and password fields during test execution. Unlike the type method, sendkeys() method does not replace existing text in any text box.

DELETING VALUES IN INPUT BOX

The **clear()** method is used to delete the text in an input box. **This method does not need a parameter.**

BUTTONS

Submit buttons are used to submit the entire form to the server. We can either use the click () method on the web element like a normal button as we have done above or use the submit () method on any web element in the form or on the submit button itself. When submit() is used, WebDriver will look up the DOM to know which form the element belongs to, and then trigger its submit function.

HOW TO SELECT CHECKBOX AND RADIO BUTTON IN SELENIUM

- Radio Buttons too can be toggled on by using the **click()** method.
- Toggling a checkbox on/off is also done using the **click()** method.
- isSelected() method is used to know whether the Checkbox is toggled on or off.

Code Snippet-

```
WebElement radio1 = driver.findElement(By.id("vfb-7-1"));
radio1.click();
if (radio1.isSelected()) {
    System.out.println("Radio Button is Toggled On");
} else {
    System.out.println("Radio Button is Toggled Off");
}
```

SELECT VALUE FROM DROPDOWN USING SELENIUM

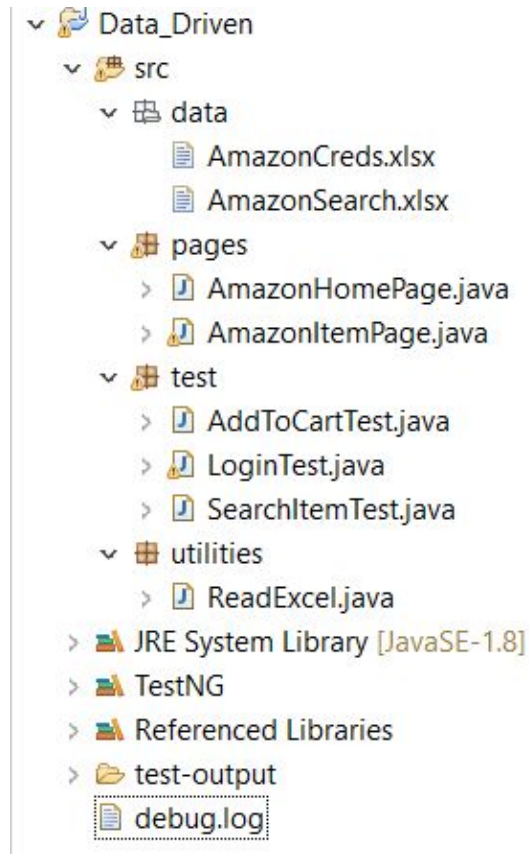
The **Select Class in Selenium** is a method used to implement the HTML SELECT tag. The html select tag provides helper methods to select and deselect the elements. The Select class is an ordinary class so New keyword is used to create its object and it specifies the web element location.

- Import the package **org.openqa.selenium.support.ui.Select**
- Instantiate the drop-down box as a "Select" object in WebDriver
Select drpCountry = new Select(driver.findElement(By.name("Country")));
- We can now start controlling "drpCountry" by using any of the available Select methods.
drpCountry .selectByVisibleText("INDIA");

Method	Description
<ul style="list-style-type: none"> • selectByVisibleText() • deselectByVisibleText() 	<p>Selects/deselects the option that displays the text matching the parameter.</p> <p>Parameter: The exactly displayed text of a particular option</p>
<ul style="list-style-type: none"> • selectByValue() • deselectByValue() 	<p>Selects/deselects the option whose "value" attribute matches the specified parameter.</p> <p>Parameter: value of the "value" attribute</p>
<ul style="list-style-type: none"> • selectByIndex() • deselectByIndex() 	<p>Selects/deselects the option at the given index.</p> <p>Parameter: the index of the option to be selected.</p>
<ul style="list-style-type: none"> • isMultiple() <pre>if(drpCountry.isMultiple()){ //do something }</pre>	<p>Returns TRUE if the drop-down element allows multiple selections at a time; FALSE if otherwise.</p> <p>No parameters needed</p>
<ul style="list-style-type: none"> • deselectAll() 	<p>Clears all selected entries. This is only valid when the drop-down element supports multiple selections.</p> <p>No parameters needed</p>

PROJECT - AUTOMATION TESTING FOR AMAZON WEB APPLICATION

FILE STRUCTURE



MAIN PACKAGE:

The main package consists of the **src package** that holds the actual project. It also consists of all the **external JAR files** added into the project.

DATA PACKAGE:

This package holds the excel files to be used for passing the data into our data driven model. The folder contains the **Amazon credentials file** and the **item list file**.

PAGES PACKAGE:

Our project uses the **Page Object Model** to hold the web elements and its properties for various pages. This package holds the Homescreen and ItemPage for testing various user flow.

TEST PACKAGE:

This file contains the different tests that have been conducted in our project. These tests are carried out for multiple data that have been provided from the data package. The tests created here are -

- Login test,
- Logout test,
- Item search test and
- Add item to cart test.

UTILITIES PACKAGE:

This package holds the utility file that will be used throughout the project for accessing and parsing the different data for our test cases.

MANUAL TESTING DOCUMENT:

LOGIN LOGOUT FLOW:

MANUAL TESTING						
Module Name:			Login and Logout Flow			
Project Name:			Amazon_Data_Driven_Test			
TEST CASE NO.	PRE-CONDITION	Test Case	Test Condition	Test Steps	Expected Results	Actual Results
1A	N/A	Verify Login Flow	Successful Login and Logout when genuine Username and Password is provided	•Go to amazon Website •Enter username •Enter password •Click Login •Hover over profile tab •Click Logout	Login Successful	Login and Logout Successful (VALID CREDENTIALS)
1B						Login and Logout Unsuccessful (INVALID CREDENTIALS)
1C						Login and Logout Unsuccessful (INVALID CREDENTIALS)
1D						Login and Logout Successful (VALID CREDENTIALS)

SEARCH ITEM FLOW:

Module Name:		Search Item Flow				
Project Name:		Amazon_Data_Driven_Test				
TEST CASE NO.	PRE-CONDITION	Test Case	Test Condition	Test Steps	Expected Results	Actual Results
2A	N/A	Verify that various items can be searched in the Amazon website	Successfu search of items in the website when genuineproducts are provided	•Go to amazon Website •Enter item to be searched •Click Search Button •Verify if products are available or not	Item Search Successful	Item Search Successful (PLATES)
2B						Item Search Successful (PS4)
2C						Item Search Successful (BOOK)
2D						Item Search Successful (PHONE)

ADD ITEM TO CART FLOW:

Module Name:		Add item to Cart Flow				
Project Name:		Amazon_Data_Driven_Test				
TEST CASE NO.	PRE-CONDITION	Test Case	Test Condition	Test Steps	Expected Results	Actual Results
3A	Sign-in to a valid amazon account to add items to the cart	Verify Add Items to Cart Flow	Successful additions of items into the Amazon cart if the Username and Password are genuine and the items searched is available	•Go to amazon Website •Enter username •Enter password •Click Login •Enter item to be searched •Click Search Button •Verify if products are available or not •Add item to cart	Add Item to Cart Successful	Item Added to Cart Successful (PLATES)
3B						Item Added to Cart Successful (PS4)
3C						Item Added to Cart Successful (BOOKS)
3D						Item Added to Cart Successful (PHONE)

IMPORTANT CODE SNIPPETS FOR LOGIN/LOGOUT FLOW: PAGE OBJECT MODEL

```
public class AmazonHomePage {  
    WebDriver driver;  
  
    By profileButton = By.xpath("//*[@id='nav-link-accountList']");  
  
    By amazonUsername = By.xpath("//*[@id='ap_email']");  
  
    By amazonPassword = By.xpath("//*[@id='ap_password']");  
  
    By continueButton = By.xpath("//*[@id='continue']");  
  
    By loginButton = By.xpath("//*[@id='signInSubmit']");  
  
    public AmazonHomePage(WebDriver driver){  
        this.driver = driver;  
    }  
  
    /**  
    * This POM method will be exposed in test case to login in the application  
    * @param amazonUsername  
    * @param amazonPassword  
    */  
  
    public void loginAmazon(String amazonUsername,String amazonPassword){  
        this.clickProfile();  
        this.setUserNames(amazonUsername);  
        this.clickContinue();  
    }  
}
```

```

        this.setPassword(amazonPassword);

        this.clickLogin();
    }

    /**
     * This POM method will be exposed in test case to logout in the
     application
     */

    public void logoutAmazon() throws InterruptedException{

        Actions actions = new Actions(driver);

        WebDriverWait wait = new WebDriverWait(driver, 10);

        WebElement profile =driver.findElement(profileButton);

        actions.moveToElement(profile).perform();

        WebElement signOutBtn =driver.findElement(signOutButton);

        wait.until(ExpectedConditions.elementToBeClickable(signOutBtn));

        signOutBtn.click();

    }

```

LOGIN TEST WITH DATA DRIVEN MODEL

```

public class LoginTest {

    AmazonHomePage amazonHomePage;

    WebDriver driver;

    @BeforeTest

    public void initialisation(){

        System.setProperty("webdriver.chrome.driver","E:\\Selenium\\Selenium\\chromedriver\\chromedriver.exe");

    }

    @Test(dataProvider = "loginData")

    public void test(String username, String password) throws InterruptedException {

        driver = new ChromeDriver();
    }

```

```

        driver.manage().window().maximize();

        driver.get("https://www.amazon.in/");

        driver.manage().timeouts().implicitlyWait(15, TimeUnit.SECONDS);

        amazonHomePage = new AmazonHomePage(driver);

        amazonHomePage.loginAmazon(username, password);

        amazonHomePage.logoutAmazon();
    }

    @AfterMethod
    public void teardown(){
        driver.quit();
    }

    @DataProvider(name = "loginData")
    public String [][] loginData() throws IOException{
        String [][] data = new String[4][2];

        ReadExcel readCredentials = new ReadExcel();

        String filePath = System.getProperty("user.dir")+"\\src\\data";

        String fileName = "AmazonCreds.xlsx";

        String sheetName = "Sheet1";












        data = readCredentials.readExcel(filePath, fileName, sheetName);

        return data;
    }
}












```


TESTCASE OUTPUT












LOGIN/LOGOUT FLOW

- ▼  Default suite (2/2/0/0) (81.951 s)
 - ▼  Default test (81.951 s)
 - ▼  test.LoginTest
 - ▼  test (39.353 s)
 -  "siddharthmehta1234@gmail.com", " " (39.353 s)
 - ▼  test (8.51 s)
 -  "demo1@yahoo.com", "testCheck1" (8.51 s)
 - ▼  test (7.982 s)
 -  "demo2@yahoo.com", "testCheck2" (7.982 s)
 - ▼  test (26.106 s)
 -  "vishakhmehta1234@gmail.com", " " (26.106 s)

SEARCH FLOW

- ▼  Default suite (4/0/0/0) (37.375 s)
 - ▼  Default test (37.375 s)
 - ▼  test.SearchItemTest
 - ▼  test (9.938 s)
 -  "plates", "" (9.938 s)
 - ▼  test (8.393 s)
 -  "ps4", "" (8.393 s)
 - ▼  test (10.116 s)
 -  "books", "" (10.116 s)
 - ▼  test (8.928 s)
 -  "phone", "" (8.928 s)

ADD TO CART FLOW

- ▼  Default suite (4/0/0/0) (37.375 s)
- ▼  Default test (37.375 s)
- ▼  test.SearchItemTest
 - ▼  test (9.938 s)
 -  "plates","" (9.938 s)
 - ▼  test (8.393 s)
 -  "ps4","" (8.393 s)
 - ▼  test (10.116 s)
 -  "books","" (10.116 s)
 - ▼  test (8.928 s)
 -  "phone","" (8.928 s)