

Diploma in Software Testing and Automation Course Slide

Software Testing

Modules

- Fundamental
- Manual Testing
- Other Testing
- Automation Core Testing

Module - 1 [Fundamentals]

Agenda

- Introduction

- Software Architecture

- Software Development Life Cycle

- SDLC Models

- OOPS

Fundamental of Testing

What is Testing?(I)

Testing is the process of evaluating a system or its component(s) with the intent to find that whether it satisfies the specified requirements or not.

This activity results in the actual, expected and difference between their results.

In simple words **testing is executing a system in order to identify any gaps, errors or missing requirements in contrary to the actual desire or requirements**

According to **ANSI/IEEE 1059** standard, Testing can be defined as A process of analyzing a software item to detect the differences between existing and required conditions (that is defects/errors/bugs) and to evaluate the features of the software item.

Software testing is a process of executing a program or application with the intent of finding the software bugs.

What is Testing?(II)

Software Testing is a process used to identify the correctness, completeness, and quality of developed computer software.

- When asked, people often think that Testing only consists of running tests, i.e. executing the software
- Test execution is only a part of testing, but not all of the testing activities
- Test activities exist before and after test execution
- It can also be stated as the **process of validating and verifying** that a software program or application or product:
 - Meets the business and technical requirements that guided it's design and development
 - Works as expected
 - Can be implemented with the same characteristic

What is Testing?(III)

- A Definition (and a Misconception)

- **'The process consisting of all life cycle activities, both static and dynamic, concerned with planning, preparation and evaluation of software products and related work products to determine that they satisfy specified requirements, to demonstrate that they are fit for purpose and to detect defects.'**

What is Testing?

Let's break the definition of Software testing into the following parts:

- **Process:** Testing is a process rather than a single activity.
- **All Life Cycle Activities:** Testing is a process that's take place throughout the Software Development Life Cycle (SDLC).
 - The process of designing tests early in the life cycle can help to prevent defects from being introduced in the code. Sometimes it's referred as "**verifying the test basis via the test design**".
 - The **test basis** includes documents such as the requirements and design specifications.
- **Static Testing:** It can test and find defects without executing code. Static Testing is done during verification process. This testing includes reviewing of the documents (including source code) and static analysis. This is useful and cost effective way of testing. For example: reviewing, walkthrough, inspection, etc.

What is Testing?(Cont...)

Let's break the definition of Software testing into the following parts:

- **Dynamic Testing:** In dynamic testing the software code is executed to demonstrate the result of running tests. It's done during validation process. For example: unit testing, integration testing, system testing, etc.
- **Planning:** We need to plan as what we want to do. We control the test activities, we report on testing progress and the status of the software under test.
- **Preparation:** We need to choose what testing we will do, by selecting test conditions and designing test cases.
- **Evaluation:** During evaluation we must check the results and evaluate the software under test and the completion criteria, which helps us to decide whether we have finished testing and whether the software product has passed the tests.
- **Software products and related work products:** Along with the testing of code the testing of requirement and design specifications and also the related documents like operation, user and training material is equally important.

Testing Activities

- Planning and control
- Choosing test conditions
- Designing test cases
- Checking results
- Evaluating completion criteria
- Reporting on the testing process and system under test
- Finalizing or closure (e.g. after a test phase has been completed)
- Testing also includes reviewing of documents (including source code) and static analysis

Test Objectives

- Finding defects

- Gaining confidence in and providing information about the level of quality.

- Preventing defects

- Both dynamic testing and static testing can be used as a means for achieving these objectives

- By designing tests early in the project life cycle it can help to prevent defects from being introduced into code

- Reviews of documents throughout the lifecycle (e.g. requirements and design) also help to prevent defects appearing in the code. More about this when we cover Static techniques

- They provide information in order to improve:

- The system to be tested

- The development and testing processes

- Live operations (e.g. how long it takes for a process to run)

Objectives and purpose

- Software testing makes sure that the testing is being done properly and hence the system is ready for use.
- Good coverage means that the testing has been done to cover the various areas like functionality of the application, compatibility of the application with the OS, hardware and different types of browsers, performance testing to test the performance of the application and load testing to make sure that the system is reliable and should not crash or there should not be any blocking issues.
- It also determines that the application can be deployed easily to the machine and without any resistance. Hence the application is easy to install, learn and use.

When to test ?

For the betterment, reliability and performance of an Information System, it is always better to involve the Testing team right from the beginning of the Requirement Analysis phase. The active involvement of the testing team will give the testers a clear vision of the functionality of the system by which we can expect a better quality and error-free product.

Once the Development Team-lead analyzes the requirements, he will prepare the System Requirement Specification, Requirement Traceability Matrix. After that he will schedule a meeting with the Testing Team (Test Lead and Tester chosen for that project). The Development Team-lead will explain regarding the Project, the total schedule of modules, Deliverable and Versions.

Why Testing is Necessary?

- Testing is necessary because we all make mistakes.
- Some of those mistakes are unimportant, but some of them are **expensive or dangerous**.
- We need to check everything and anything we produce because things can always go wrong – humans make mistakes all the time.
- Since we assume that our work may have mistakes, hence we all need to check our own work.
- However some mistakes come from bad assumptions and blind spots, so we might make the same mistakes when we check our own work as we made when we did it.
- So we may not notice the flaws in what we have done.
- Ideally, we should get someone else to check our work because another person is more likely to spot the flaws.

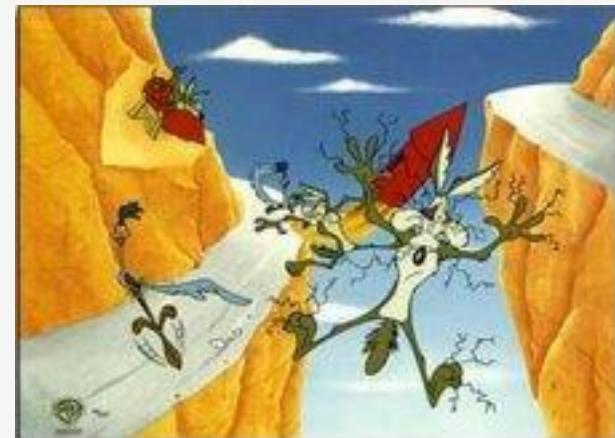
Why Testing is Necessary?

Software Systems – Some context

Software Systems are now part of our everyday life

They are used almost everywhere, for example in:

- Banking and Financial institutions
 - Retail industry
 - Central and Local Government
 - Transport (e.g. Planes, Trains and Automobiles)
 - Medicine (Hospitals, research centres)
 - Home Entertainment
- We have all experienced
Software Systems failing!



Why Testing is Necessary?

Software Systems – When things go wrong

Software System Failures can lead to:

- Human Injury or Death
 - e.g. airplanes crashing
- Technological disasters
 - e.g. Missile Systems malfunctioning
- Legal action and associated costs
 - e.g. failure to meet contractual obligations
- Loss of face for suppliers and/or their customers;
 - e.g. mis-spelling company name on mail shots



When to start software testing?

- Testing is sometimes incorrectly thought as an after-the-fact activity; performed after programming is done for a product. Instead, testing should be performed at every development stage of the product .
- If we divide the lifecycle of software development into “Requirements Analysis”, “Design”, “Programming/Construction” and “Operation and Maintenance”, then testing should accompany each of the above phases. If testing is isolated as a single phase late in the cycle, errors in the problem statement or design may incur exorbitant costs.
- Not only must the original error be corrected, but the entire structure built upon it must also be changed. Therefore, testing should not be isolated as an inspection activity. Rather testing should be involved throughout the SDLC in order to bring out a quality product.

When to stop software testing ?

“When to stop testing” is one of the most difficult questions to a test engineer. The following are few of the common Test Stop criteria:

- All the high priority bugs are fixed.

- The rate at which bugs are found is too small.

- The testing budget is exhausted.

- The project duration is completed.

- The risk in the project is under acceptable limit.

Practically, we feel that the decision of stopping testing is based on the level of the risk acceptable to the management. The risk can be measured by Risk analysis but for small duration / low budget / low resources project, risk can be deduced by simply: –

- Measuring Test Coverage.

- Number of test cycles.

- Number of high priority bugs.

7 Key Principle

General Testing Principles

1. Testing shows presence of Defects
2. Exhaustive Testing is Impossible!
3. Early Testing
4. Defect Clustering
5. The Pesticide Paradox
6. Testing is Context Dependent
7. Absence of Errors Fallacy

Testing shows presence of Defects

- Testing can show that defects are present, but cannot prove that there are no defects.
- Testing **reduces the probability of undiscovered defects** remaining in the software but, even if no defects are found, it is not a proof of correctness.
- We test to find Faults
- As we find more defects, the **probability of undiscovered defects** remaining in a system reduces.
- However Testing **cannot prove** that there are **no** defects present

Exhaustive Testing is Impossible!

- Testing everything including **all combinations of inputs and preconditions is not possible.**

- So, instead of doing the exhaustive testing we can use risks and priorities to focus testing efforts.

- For example: In an application in **one screen there are 15 input fields**, each having 5 possible values, then to test all the valid combinations you would need **30 517 578 125 (5^{15}) tests.**

- This is very unlikely that the project timescales would allow for this number of tests.

- So, assessing and managing risk is one of the most important activities and reason for testing in any project.

- We have learned that we cannot test everything (i.e. all combinations of inputs and pre-conditions).

- That is we must **Prioritise** our testing effort using a **Risk Based Approach.**

Why do not Testing Everything?

- Exhaustive testing of complex software applications:

- requires enormous resources
- is too expensive
- takes too long

- It is therefore impractical

- Need an alternative that is pragmatic, affordable, timely and provides results



Why do not Testing Everything?

Examples:

System has 20 screens
Average 4 menus / screen
Average 3 options / menu
Average of 10 fields / screen
2 types of input per field
Around 100 possible values

Approximate total for exhaustive testing
 $20 \times 4 \times 3 \times 10 \times 2 \times 100 = 480,000$ tests



Test length = 1 sec then test duration = 17.7 days
Test length = 10 sec then test duration = 34 weeks
Test length = 1 min then test duration = 4 years
Test length = 10 mins then test duration = 40 years!

Early Testing

- Testing activities should start as early as possible in the software or system development life cycle, and should be focused on defined objectives.
- Testing activities should start as **early** as possible in the development life cycle
- These activities should be focused on defined objectives – outlined in the Test Strategy
- Remember from our Definition of Testing, that Testing doesn't start once the code has been written!



Defect Clustering

- A small number of modules contain most of the defects discovered during pre-release testing, or are responsible for the most operational failures.
- Defects are not evenly spread in a system
- They are ‘clustered’
- In other words, most defects found during testing are usually confined to a small number of modules
- Similarly, most operational failures of a system are usually confined to a small number of modules
- An important consideration in test prioritisation!



Pesticide Paradox

If the same tests are repeated over and over again, eventually the same set of **test cases will no longer find any new defects**.

To overcome this “pesticide paradox”, the test cases need to be **regularly reviewed and revised, and new and different tests need to be written to exercise different parts of the software or system to potentially find more defects**.

Testing identifies bugs, and programmers respond to fix them

As bugs are eliminated by the programmers, the software improves

As software improves the effectiveness of previous tests erodes



Pesticide Paradox

Therefore we must learn, create and use new tests based on new techniques to catch new bugs

N.B It's called the "pesticide paradox" after the agricultural phenomenon, where bugs such as the boll weevil build up tolerance to pesticides, leaving you with the choice of ever-more powerful pesticides followed by ever-more powerful bugs or an altogether different approach.' – Beizer 1995



Testing is Context Dependent

- Testing is basically context dependent.
- Testing is done differently in different contexts
- Different kinds of sites are tested differently.**
- For example
 - **Safety - critical software is tested differently from an e-commerce site.**
- Whilst, Testing can be 50% of development costs, in NASA's Apollo program it was 80% testing
- 3 to 10 failures per thousand lines of code (KLOC) typical for commercial software
- 1 to 3 failures per KLOC typical for industrial software
- 0.01 failures per KLOC for NASA Shuttle code!
- Also different industries impose different testing standards

Absence of Errors Fallacy

- If the system built is unusable and does not fulfill the user's needs and expectations then finding and fixing defects does not help.
- If we build a system and, in doing so, find and fix defects
 - It doesn't make it a **good** system
- Even after defects have been resolved it may still be **unusable** and/or does not fulfil the users' **needs and expectations**

Project Vs Product

S.NO.	Project	Product		
1.	It comprises the steps involved in making a software before it is actually available to the market.	It is the manufacture of the project for users.	4.	It focuses on increasing the performance of the software that is being built. A product focuses on the final result and the efficiency with which it can solve the given problem.
2.	The main goal of a project is to form a new product that has not already been made.	The main goal of the product is to complete the work successfully (solve a specific problem).	5.	A project is done only once to get a new software. A product can be made again and again for the purpose of distribution among users.
3.	Project is undertaken to form a new software.	Product is the final production of the project.	6.	It is more risky as here, a software is being made for the first time. It is relatively less risky as the software has already been made and tested. The only risk in most cases would be of wear and tear.
			7.	It is handled by the project managers. It is handled by the product managers.

Software Architecture

- Software Architecture consists of One Tier, Two Tier, Three Tier, and N-Tier architectures.
- A “tier” can also be referred to as a “layer”.
- Three layers are involved in the application namely Presentation Layer, Business Layer, and Data Layer.

1. Presentation Layer

It is also known as the Client layer.

The top most layer of an application.

This is the layer we see when we use the software.

By using this layer we can access the web pages.

The main function of this layer is to communicate with the Application layer.

This layer passes the information which is given by the user in terms of keyboard actions, mouse clicks to the Application Layer..

For example, the login page of Gmail where an end-user could see text boxes and buttons to enter user id, password, and to click on sign-in.

In simple words, it is to view the application.

2. Application Layer

- It is also known as Business Logic Layer which is also known as the logical layer.

- As per the Gmail login page example, once the user clicks on the login button, the Application layer interacts with the Database layer and sends required information to the Presentation layer.

- It controls an application's functionality by performing detailed processing.

- This layer acts as a mediator between the Presentation and the Database layer. Complete business logic will be written in this layer.

- In simple words, it is to perform operations on the application.

3. Data Layer

- The data is stored in this layer.

- The application layer communicates with the Database layer to retrieve the data.

- It contains methods that connect the database and performs required action e.g.: insert, update, delete, etc.

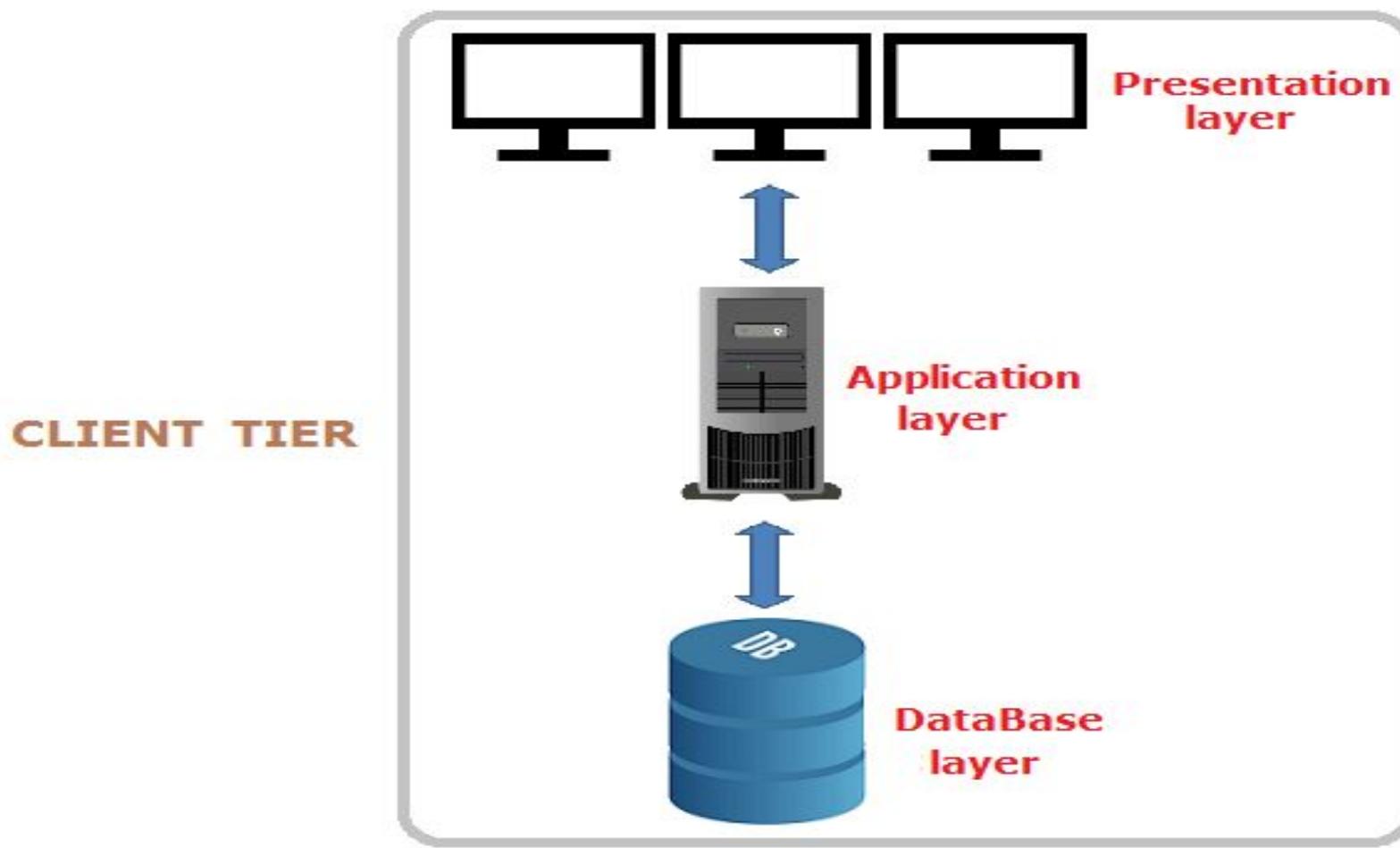
Types of Software Architecture:

1. One Tier Architecture:

- **One-tier architecture has all the layers such as Presentation, Business, Data Access layers in a single software package.**
- Applications that handle all the three tiers such as **MP3 player, MS Office** come under the one-tier application.
- **The data is stored in the local system or a shared drive.**

1. One Tier Architecture:

ONE-TIER ARCHITECTURE

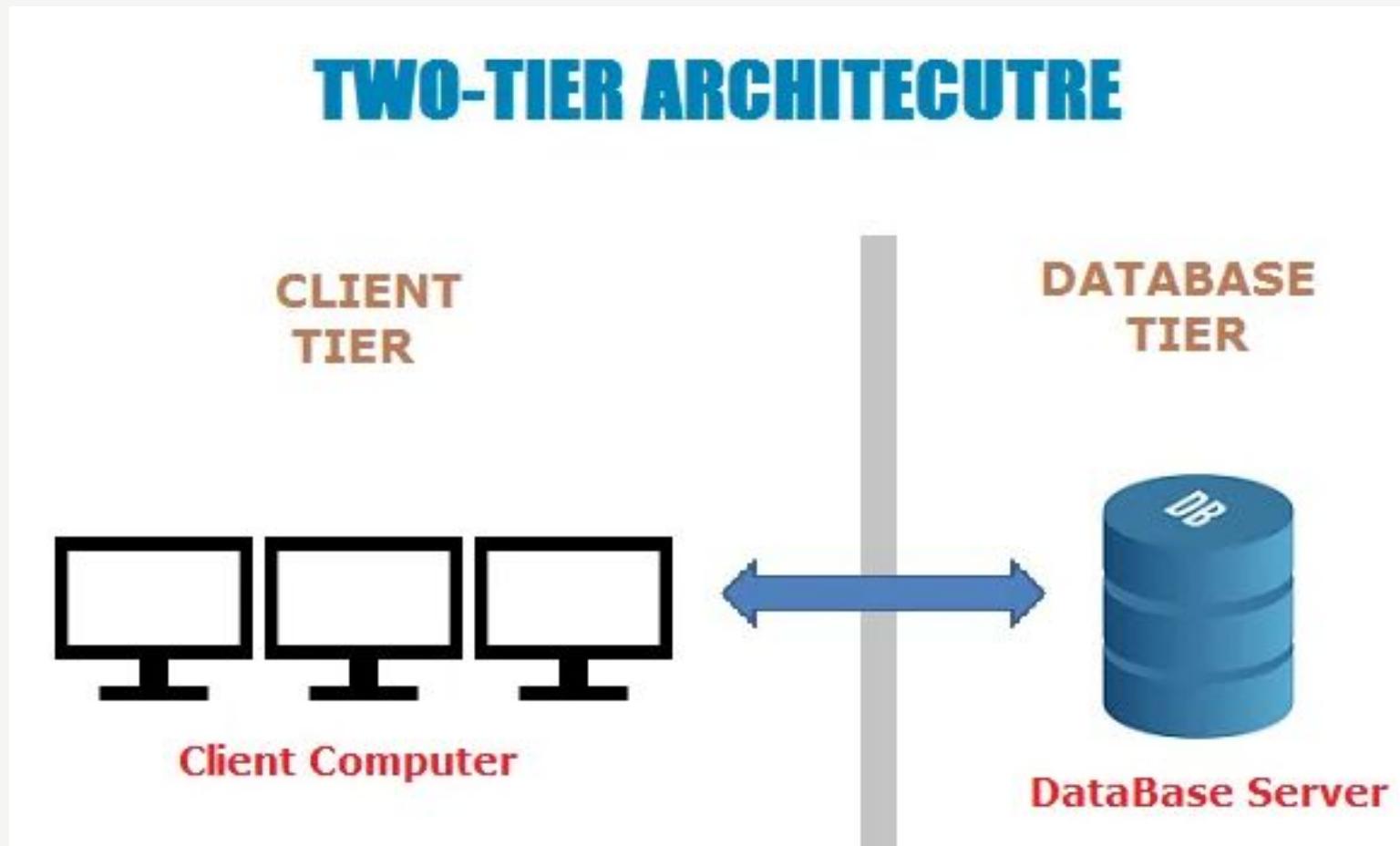


Types of Software Architecture:

2. Two Tier Architecture:

- **The Two-tier architecture is divided into two parts:**
 1. Client Application (Client Tier)
 2. Database (Data Tier)
- The client system handles both Presentation and Application layers and the Server system handles the Database layer. It is also known as a client-server application.
- The communication takes place between the Client and the Server. The client system sends the request to the server system and the Server system processes the request and sends back the data to the Client System

2. Two Tier Architecture:

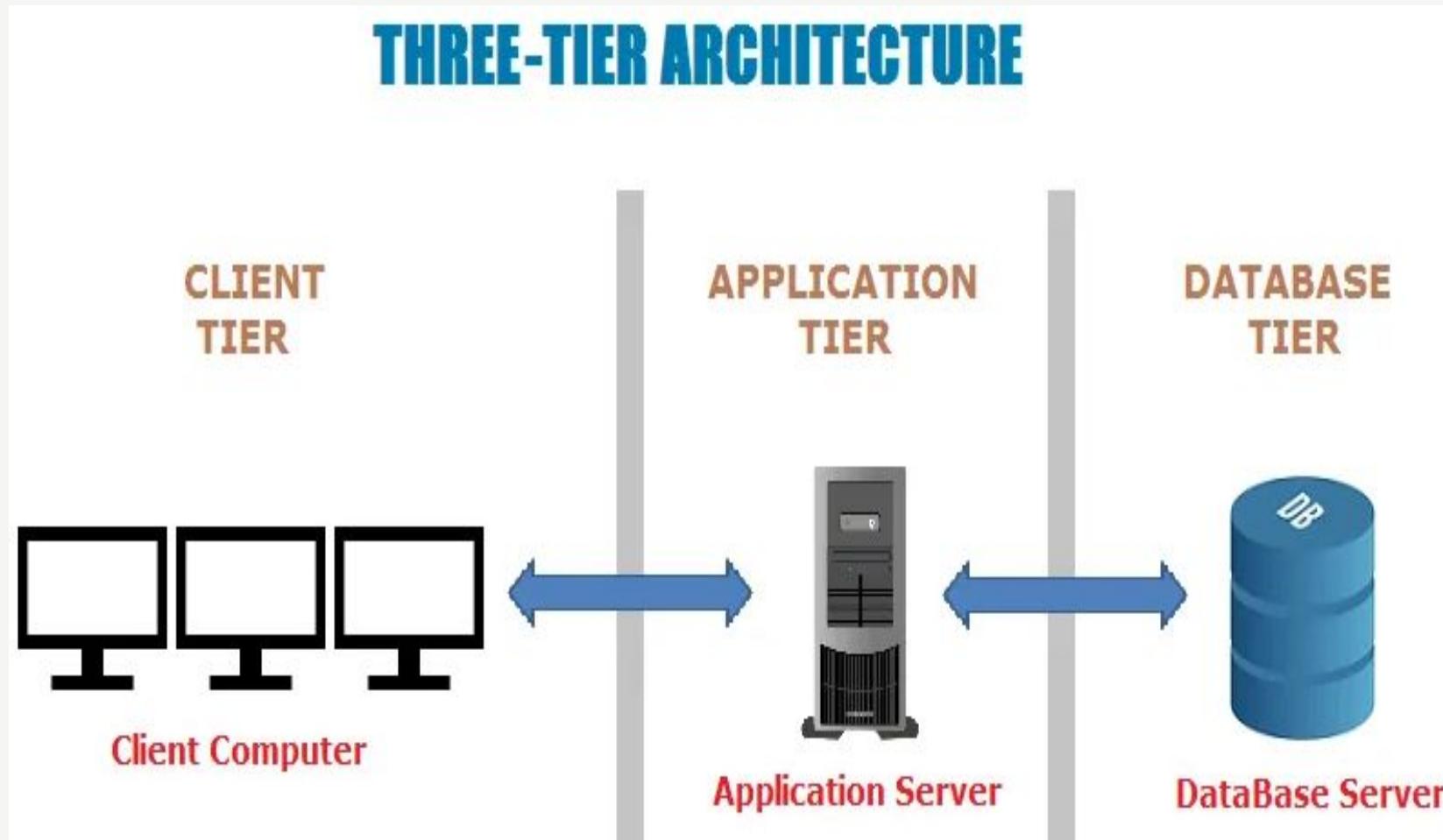


Types of Software Architecture:

3. Three Tier Architecture:

- **The Three-tier architecture is divided into three parts:**
 1. Presentation layer (Client Tier)
 2. Application layer (Business Tier)
 3. Database layer (Data Tier)
- The client system handles the Presentation layer, the Application server handles the Application layer, and the Server system handles the Database layer.
- **Another layer is the N-Tier application.**
- **It is similar to the three-tier architecture but the number of application servers is increased and represented in individual tiers in order to distribute the business logic so that the logic will be distributed.**

3. Three Tier Architecture:



System Environments

1. Dev :

- Dev environment is used for **developer's tasks**, like merging commits in the first place, running unit tests.
- Dev environment is usually not guaranteed to be stable.
- The operation can be disrupted by commit, and it doesn't do harm for the whole company.
- DEV environment is usually hooked to some CI/CD system.
- When developers do code merge, the build is automatically triggered, and application code is automatically redeployed to Dev.

System Environments

2. QA :

- QA is for testing by Quality Assurance team, both manual and automated, including running automated integration tests.
- It's considered to be more stable than Dev, because code doesn't change so often on every merge, as in Dev. So, developers cannot disrupt ongoing work of QA engineers by "risky" change.

System Environments

3. UAT :

- UAT environment is for pre-release testing.
- It is used by QA engineers, business analysts, product owners, for verifying functional requirements.
- UAT is required to be stable, because it's used not only by developers but also by business users , who serve as “functional testers”.
- It also can be used as demo environment for showing new features to the customers.

System Environments

4. PROD :

- Prod is production environment, serving primary business purpose.
- Access of developers to it usually limited only to perform technical support duties.
- example:

DEV (dev tests passed) -> deploy to QA (QA integration tests passed) -> deploy to UAT (UAT acceptance tests passed) -> deploy to Prod

Software Development Life Cycle

- SDLC is a structure imposed on the development of a software product that defines the process for planning, implementation, testing, documentation, deployment, and ongoing maintenance and support. There are a number of different development models.

- A Software Development Life Cycle is essentially a series of steps, or phases, that provide a model for the development and lifecycle management of an application or piece of software.

- The methodology within the SDLC process can vary across industries and organizations, but standards such as ISO/IEC 12207 represent processes that establish a lifecycle for software, and provide a mode for the development, acquisition, and configuration of software systems.

SDLC Phases

Requirements Collection/Gathering	Establish Customer Needs
Analysis	Model And Specify the requirements- "What"
Design	Model And Specify a Solution – "Why"
Implementation	Construct a Solution In Software
Testing	Validate the solution against the requirements
Maintenance	Repair defects and adapt the solution to the new requirements

Requirement Gathering

- Features
- Usage scenarios
- Although requirements may be documented in written form, they may be incomplete, unambiguous, or even incorrect.
- Requirements will Change!
 - Inadequately captured or expressed in the first place
 - User and business needs change during the project
 - Validation is needed throughout the software lifecycle, not only when the “final system” is delivered.
 - Build constant feedback into the project plan
 - Plan for change
 - Early prototyping [e.g., UI] can help clarify the requirements
 - Functional and Non-Functional

Requirement Gathering(Cont...)

- Requirements definitions usually consist of **natural language**, supplemented by (e.g., UML) **diagrams and tables**.
- Three types of problems can arise:
 - **Lack of clarity:** It is hard to write documents that are both **precise and easy-to-read**.
 - **Requirements confusion:** **Functional and Non-functional** requirements tend to be intertwined.
 - **Requirements Amalgamation:** Several **different requirements** may be expressed together.

Requirement Gathering(Cont...)

- Types of Requirements:

- **Functional Requirements:** describe system **services or functions.**
 - Compute sales tax on a purchase
 - Update the database on the server
- **Non-Functional Requirements:** are **constraints** on the system or the development process.
- **Non-functional requirements may be more critical than functional requirements.**
- **If these are not met, the system is useless!**

Analysis Phase

- The analysis phase defines the requirements of the system, independent of how these requirements will be accomplished.
- This phase defines the problem that the customer is trying to solve.
- The deliverable result at the end of this phase is a requirement document.
- Ideally, this document states in a clear and precise fashion what is to be built.
- This analysis represents the “**what**” phase.
- The requirement documentaries to capture the requirements from the customer's perspective by defining goals.

Analysis Phase

- This phase starts with the requirement document delivered by the requirement phase and maps the requirements into architecture.
- The architecture defines the components, their interfaces and behaviors.
- The deliverable design document is the architecture.
- This phase represents the “**how**” phase.
- Details on computer programming languages and environments, machines, packages, application architecture, distributed architecture layering, memory size, platform, algorithms, data structures, global type definitions, interfaces, and many other engineering details are established.
- The design may include the usage of existing components.

Design Phase

- Design Architecture Document
- Implementation Plan
- Critical Priority Analysis
- Performance Analysis
- Test Plan
- The Design team can now expand upon the information established in the requirement document.
- The requirement document must guide this decision process.
- Analyzing the trade-offs of necessary complexity allows for many things to remain simple which, in turn, will eventually lead to a higher quality product.
- The architecture team also converts the typical scenarios into a test plan.

Implementation Phase

- In the implementation phase, the team builds the components either from scratch or by composition.
- Given the architecture document from the design phase and the requirement document from the analysis phase, the team should build exactly what has been requested, though there is still room for innovation and flexibility.
- For example, a component may be narrowly designed for this particular system, or the component may be made more general to satisfy a reusability guideline.
 - Implementation - Code
 - Critical Error Removal
- The implementation phase deals with issues of quality, performance, baselines, libraries, and debugging.
- The end deliverable is the product itself. There are already many established techniques associated with implementation.

Testing Phase

- Simply stated, quality is very important. Many companies have not learned that quality is important and deliver more claimed functionality but at a lower quality level.
- It is much easier to explain to a customer why there is a missing feature than to explain to a customer why the product lacks quality.
- A customer satisfied with the quality of a product will remain loyal and wait for new functionality in the next version.
- Quality is a distinguishing attribute of a system indicating the degree of excellence.
- Regression Testing
- Internal Testing
- Unit Testing
- Application Testing
- Stress Testing

Testing Phase(Cont...)

- The testing phase is a separate phase which is performed by a different team after the implementation is completed.
- There is merit in this approach; it is hard to see one's own mistakes, and a fresh eye can discover obvious errors much faster than the person who has read and re-read the material many times.
- Unfortunately, delegating (alternate) testing to another team leads to as lack (dull) attitude regarding quality by the implementation team.
- If the teams are to be known as craftsmen, then the teams should be responsible for establishing high quality across all phases.
- an attitude change must take place to guarantee quality. Regardless if testing is done after the-fact or continuously, testing is usually based on a regression technique split into several major focuses, namely internal, unit, application, and stress.

Maintenance Phase

- Software maintenance is one of the activities in software engineering, and is the process of enhancing and optimizing deployed software (software release), as well as fixing defects.
- Software maintenance is also one of the phases in the System Development Life Cycle (SDLC), as it applies to software development. The maintenance phase is the phase which comes after deployment of the software into the field.
- The developing organization or team will have some mechanism to document and track defects and deficiencies.
- configuration and version management
- reengineering (redesigning and refactoring)
- updating all analysis, design and user documentation
- Repeatable, automated tests enable evolution and refactoring

Maintenance Phase(Cont...)

Maintenance is the process of changing a system after it has been deployed.

- **Corrective maintenance:** identifying and repairing defects
- **Adaptive maintenance:** adapting the existing solution to the **new platforms**.
- **Perfective Maintenance:** implementing the **new requirements**

In a spiral lifecycle, everything after the delivery and deployment of the first prototype can be considered “**maintenance**”!

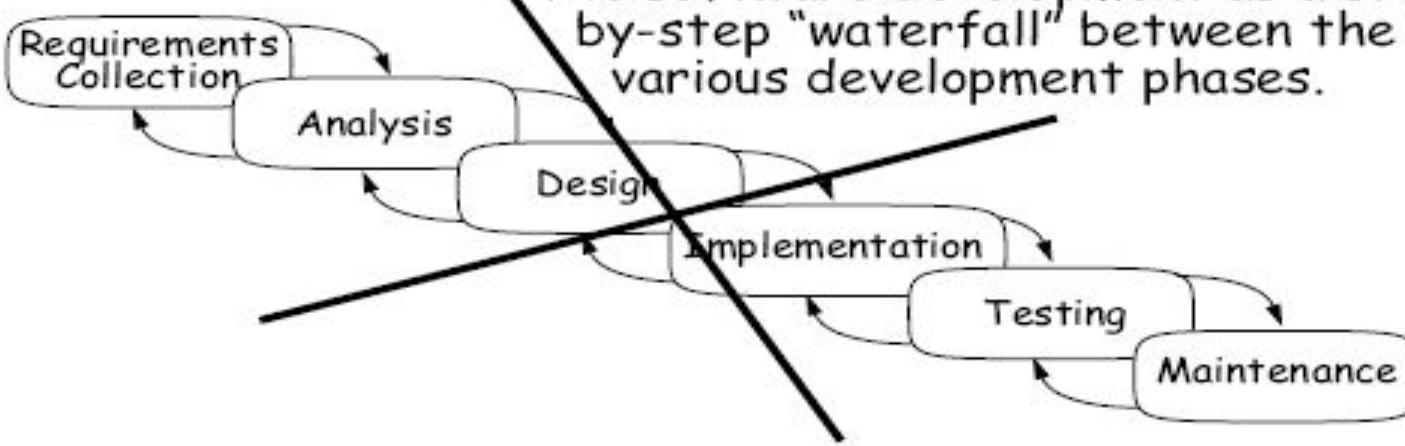
- Software just like most other products is typically released with a known set of defects and deficiencies.
- The software is released with the issues because the development organization decides the utility and value of the software at a particular level of quality outweighs the impact of the known defects and deficiencies.

Software Testing Methodologies

- Introduction
- Waterfall Model/Methodology (Classical Software Cycle)
- Iterative & Incremental Model/Methodology
- Spiral Model/Methodology
- Agile Model/Methodology
- Use Case

Waterfall Model (Classical Software Cycle)

The classical software lifecycle models the software development as a step-by-step "waterfall" between the various development phases.



The waterfall is unrealistic for many reasons, especially:

- Requirements must be "**frozen**" too early in the life cycle
- Requirements are **validated too late**

Applications(When to use?)

- Requirements are very well documented, clear and fixed.
- Product definition is stable.
- Technology is understood and is not dynamic.
- There are no ambiguous requirements.
- Ample resources with required expertise are available to support the product.
- The project is short.

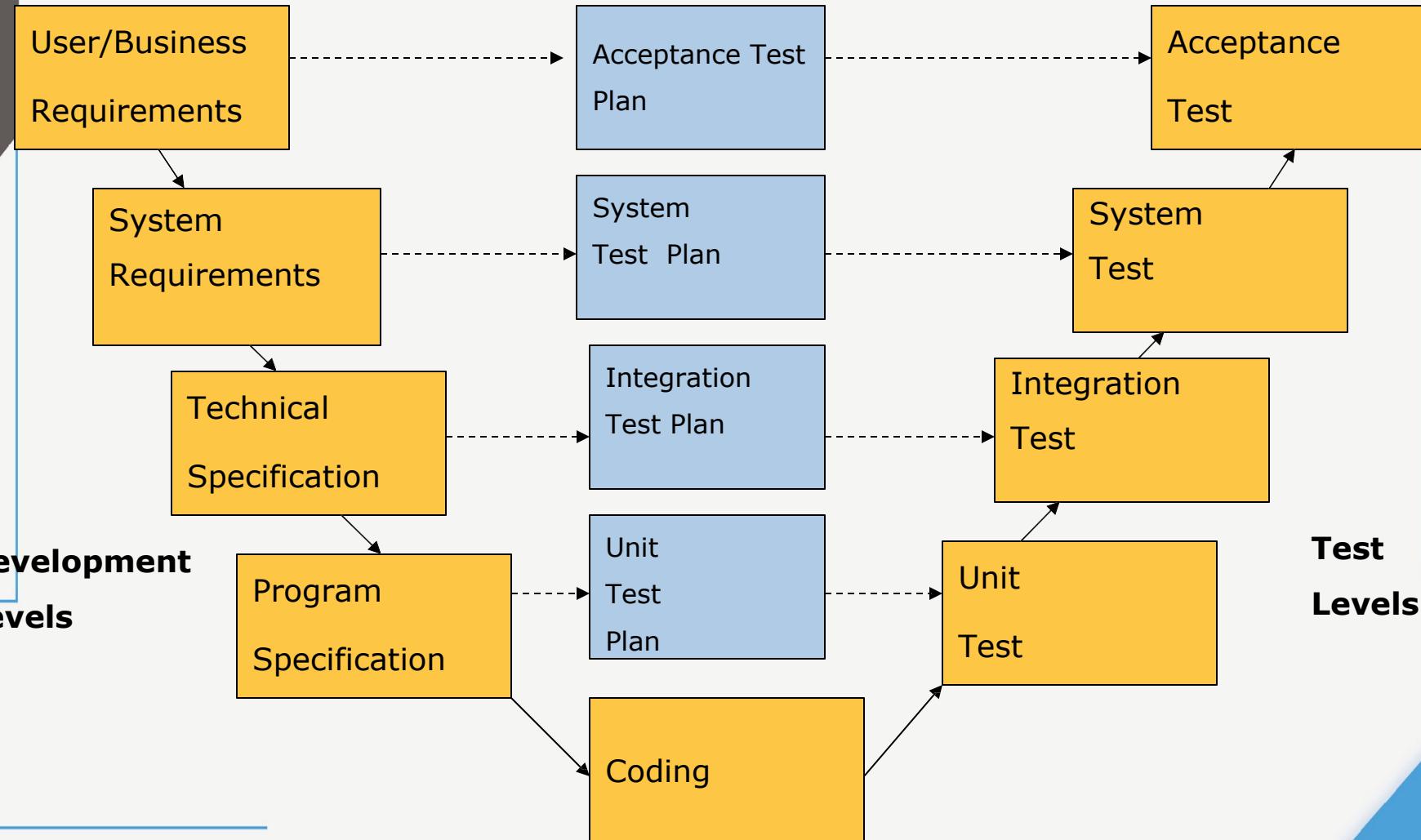
Pros (Why Waterfall Model)

- 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.
- Phases are processed and completed one at a time.
- Works well for smaller projects where requirements are very well understood.
- Clearly defined stages.
- Well understood milestones.
- Easy to arrange tasks.
- Process and results are well documented.

Cons (Why not Waterfall Model)

- No working software is produced until late during the life cycle.
- High amounts of 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. So risk and uncertainty is high with this process model.
- It is difficult to measure progress within stages.
- Cannot accommodate changing requirements.
- No working software is produced until late in the life cycle.
- Adjusting scope during the life cycle can end a project.
- Integration is done as a "big-bang. at the very end, which doesn't allow identifying any technological or business bottleneck or challenges early.

V-Model Design



V-Model Description

The V - model is SDLC model where execution of processes happens in a sequential manner in V-shape. **It is also known as Verification and Validation model.**

Under V-Model, the corresponding testing phase of the development phase is planned in parallel.

So there are Verification phases on one side of the .V. and Validation phases on the other side. Coding phase joins the two sides of the V- Model.

Although variants of the V-model exist, a common type of V-model uses four test levels, corresponding to the four development levels.

The four levels used in this syllabus are:

- **Component (unit) testing**
- **Integration testing**
- **System testing**
- **Acceptance testing**

Verification Phase

Business Requirement Analysis: This is the first phase in the development cycle where the product requirements are understood from the customer perspective. This phase involves detailed communication with the customer to understand his expectations and exact requirement. This is a very important activity and need to be managed well, as most of the customers are not sure about what exactly they need. The acceptance test design planning is done at this stage as business requirements can be used as an input for acceptance testing.

System Design (System Requirement): Once you have the clear and detailed product requirements, it's time to design the complete system. System design would comprise of understanding and detailing the complete hardware and communication setup for the product under development. System test plan is developed based on the system design. Doing this at an earlier stage leaves more time for actual test execution later.

Verification Phase(Cont...)

Architectural Design (Technical Specification): Architectural specifications are understood and designed in this phase. Usually more than one technical approach is proposed and based on the technical and financial feasibility the final decision is taken. System design is broken down further into modules taking up different functionality. This is also referred to as High Level Design (HLD).

The data transfer and communication between the internal modules and with the outside world (other systems) is clearly understood and defined in this stage. With this information, integration tests can be designed and documented during this stage.

Module Design (Program Specification): In this phase the detailed internal design for all the system modules is specified, referred to as Low Level Design (LLD). It is important that the design is compatible with the other modules in the system architecture and the other external systems. Unit tests are an essential part of any development process and helps eliminate the maximum faults and errors at a very early stage. Unit tests can be designed at this stage based on the internal module designs.

Code Phase

The actual coding of the system modules designed in the design phase is taken up in the Coding phase. The best suitable programming language is decided based on the system and architectural requirements. The coding is performed based on the coding guidelines and standards. The code goes through numerous code reviews and is optimized for best performance before the final build is checked into the repository.



Validation Phase

- **Unit Testing:** Unit tests designed in the module design phase are executed on the code during this validation phase. Unit testing is the testing at code level and helps eliminate bugs at an early stage, though all defects cannot be uncovered by unit testing.
- **Integration Testing:** Integration testing is associated with the architectural design phase. Integration tests are performed to test the coexistence and communication of the internal modules within the system.
- **System Testing:** System testing is directly associated with the System design phase. System tests check the entire system functionality and the communication of the system under development with external systems. Most of the software and hardware compatibility issues can be uncovered during system test execution.
- **Acceptance Testing:** Acceptance testing is associated with the business requirement analysis phase and involves testing the product in user environment. Acceptance tests uncover the compatibility issues with the other systems available in the user environment. It also discovers the non-functional issues such as load and performance defects in the actual user environment.

V-Model Application

- V- Model application is almost same as waterfall model, as both the models are of sequential type.
- 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 disciplined domain.
- Following are the suitable scenarios to use V-Model:
 - Requirements are well defined, clearly documented and fixed.
 - Product definition is stable.
 - Technology is not dynamic and is well understood by the project team.
 - There are no ambiguous or undefined requirements.
 - The project is short.

V-Model Pros & Cons

Pros of V-Model

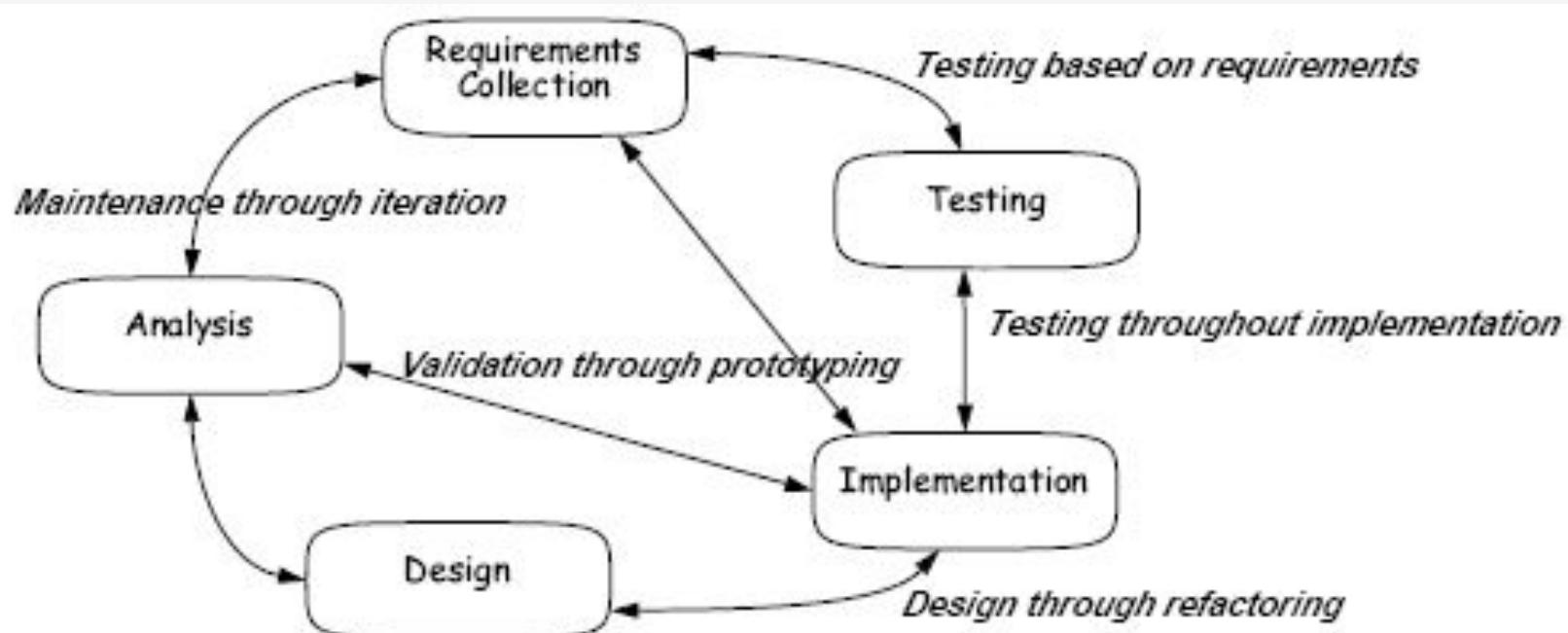
- This is a highly disciplined model and Phases are completed one at a time.
- Works well for smaller projects where requirements are very well understood.
- Simple and easy to understand and use.
- Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.

Cons of V-Model

- 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.
- Once an application is in the testing stage, it is difficult to go back and change a functionality
- No working software is produced until late during the life cycle.

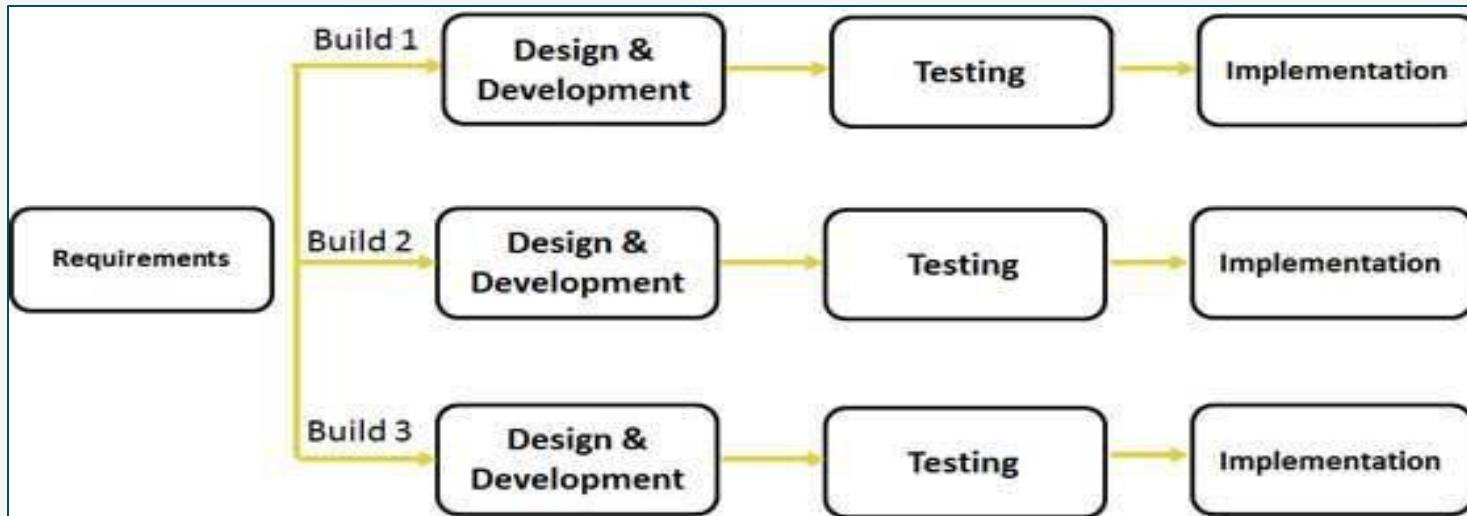
Criteria	Verification	Validation
Definition	The process of evaluating work-products (not the actual final product) of a development phase to determine whether they meet the specified requirements for that phase.	The process of evaluating software during or at the end of the development process to determine whether it satisfies specified business requirements.
Objective	To ensure that the product is being built according to the requirements and design specifications. In other words, to ensure that work products meet their specified requirements.	To ensure that the product actually meets the user's needs, and that the specifications were correct in the first place. In other words, to demonstrate that the product fulfills its intended use when placed in its intended environment.
Question	Are we building the product right?	Are we building the right product?
Evaluation Items	Plans, Requirement Specs, Design Specs, Code, Test Cases	The actual product/software.
Activities	<ul style="list-style-type: none"> • Reviews • Walkthroughs • Inspections 	<ul style="list-style-type: none"> • Testing

Iterative Model/Methodology



- In Practice, development is always iterative, and all activates progress in parallel.
- If the waterfall model is pure fiction, why is it still the standard software process?

Iterative & Incremental Model



Iterative process starts with a simple implementation of a subset of the software requirements and iteratively enhances the evolving versions until the full system is implemented. At each iteration, design modifications are made and new functional capabilities are added. The basic idea behind this method is to develop a system through **repeated cycles (iterative)** and in smaller portions at a time (**incremental**).

Applications(When to use?)

- Requirements of the complete system are clearly defined and understood.
- Major requirements must be defined; however, some functionalities or requested enhancements may evolve with time.
- There is a time to the market constraint.
- A new technology is being used and is being learnt by the development team while working on the project.
- Resources with needed skill set are not available and are planned to be used on contract basis for specific iterations.
- There are some high risk features and goals which may change in the future.

Pros

- Some working functionality can be developed quickly and early in the life cycle.
- Results are obtained early and periodically.
- Parallel development can be planned.
- Progress can be measured.
- Less costly to change the scope/requirements.
- Testing and debugging during smaller iteration is easy.
- Risks are identified and resolved during iteration; and each iteration is an easily managed milestone.

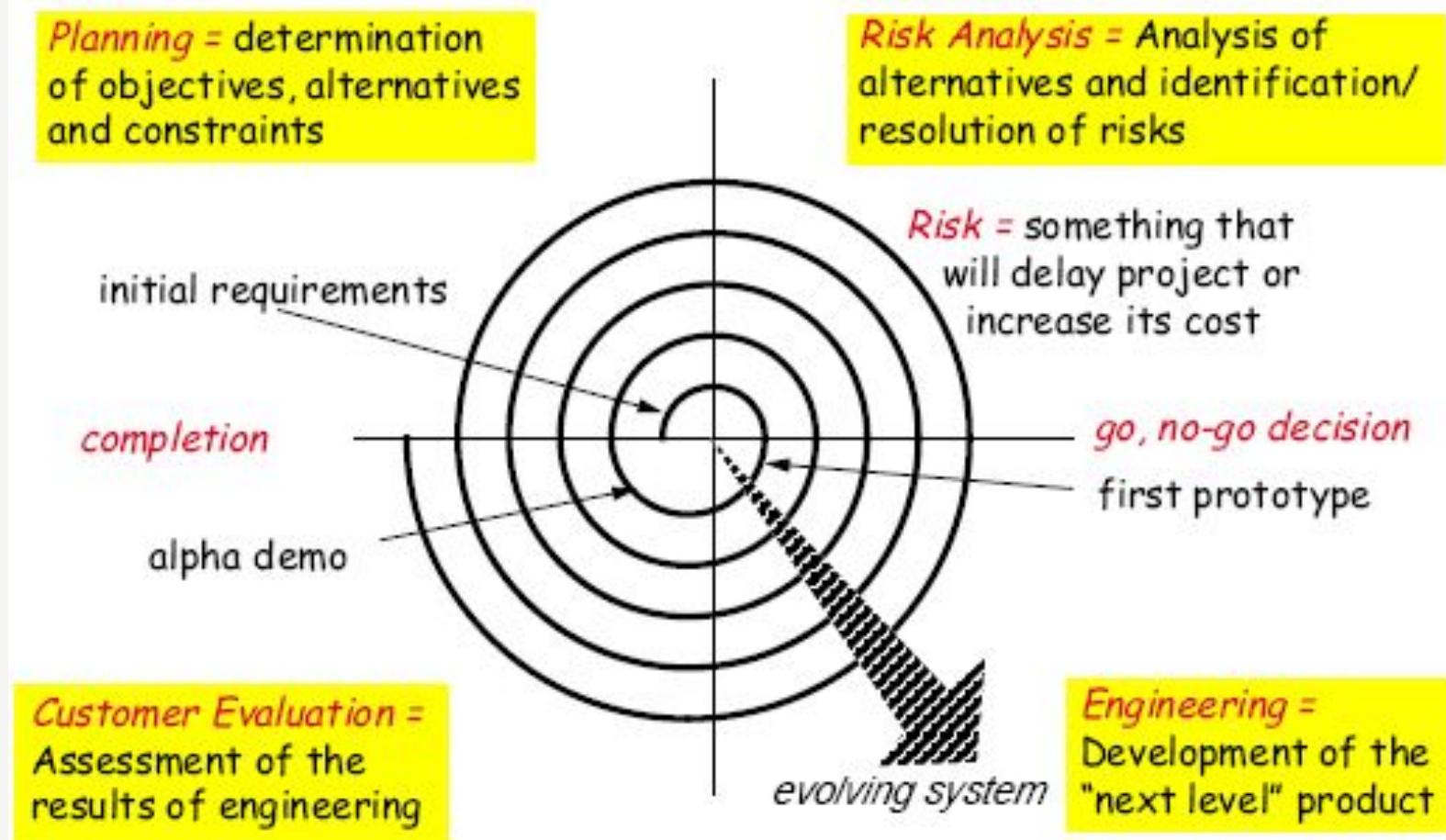
Pros

- Easier to manage risk - High risk part is done first.
- With every increment operational product is delivered.
- Issues, challenges & risks identified from each increment can be utilized/applied to the next increment.
- Risk analysis is better.
- It supports changing requirements.
- Initial Operating time is less.
- Better suited for large and mission-critical projects.
- During life cycle software is produced early which facilitates customer evaluation and feedback.

Cons

- More resources may be required.
- Although cost of change is lesser but it is not very suitable for changing requirements.
- More management attention is required.
- System architecture or design issues may arise because not all requirements are gathered in the beginning of the entire life cycle.
- Defining increments may require definition of the complete system.
- Not suitable for smaller projects.
- Management complexity is more.
- End of project may not be known which a risk is.
- Highly skilled resources are required for risk analysis.
- Project's progress is highly dependent upon the risk analysis phase.

Bohem's Spiral Model



Application

Spiral Model is very widely used in the software industry as it is in sync with the natural development process of any product i.e. learning with maturity and also involves minimum risk for the customer as well as the development firms. Following are the typical uses of Spiral model:

- When costs there are a budget constraint and risk evaluation is important.
- For medium to high-risk projects.
- Long-term project commitment because of potential changes to economic priorities as the requirements change with time.
- Customer is not sure of their requirements which are usually the case.
- Requirements are complex and need evaluation to get clarity.
- New product line which should be released in phases to get enough customer feedback.
- Significant changes are expected in the product during the development cycle.

Pros (Why It works)

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

Cons (Why It doesn't work)

- Management is more complex.
- End of project may not be known early.
- Not suitable for small or low risk projects and could be expensive for small projects.
- Process is complex
- Spiral may go indefinitely.
- Large number of intermediate stages requires excessive documentation.

Agile Model/Methodology

- 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.
- At the end of the iteration a working product is displayed to the customer and important stakeholders.

What is Agile?

Agile model believes that every project needs to be handled differently and the existing methods need to be tailored to best suit the project requirements. In agile the tasks are divided to time boxes (small time frames) to deliver specific features for a release.

Iterative approach is taken and working software build is delivered after each iteration. Each build is incremental in terms of features; the final build holds all the features required by the customer.

Agile thought process had started early in the software development and started becoming popular with time due to its flexibility and adaptability.

Pros

- Is a very realistic approach to software development
- Promotes teamwork and cross training.
- Functionality can be developed rapidly and demonstrated.
- Resource requirements are minimum.
- Suitable for fixed or changing requirements
- Delivers early partial working solutions.
- Good model for environments that change steadily.
- Minimal rules, documentation easily employed.
- Enables concurrent development and delivery within an overall planned context.
- Little or no planning required
- Easy to manage
- Gives flexibility to developers

Cons

- Not suitable for handling complex dependencies.
- 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.
- Strict delivery management dictates the scope, functionality to be delivered, and adjustments to meet the deadlines.
- Depends heavily on customer interaction, so if customer is not clear, team can be driven in the wrong direction.
- There is very high individual dependency, since there is minimum documentation generated.
- Transfer of technology to new team members may be quite challenging due to lack of documentation.

Use-case

A use-case **is the specification of a sequence of actions, including variants, that a system (or other entity) can perform, interacting with actors of the system.**

- Ex: buy a DVD through the internet

A scenario **is a particular trace of action occurrences, starting from a known initial state**

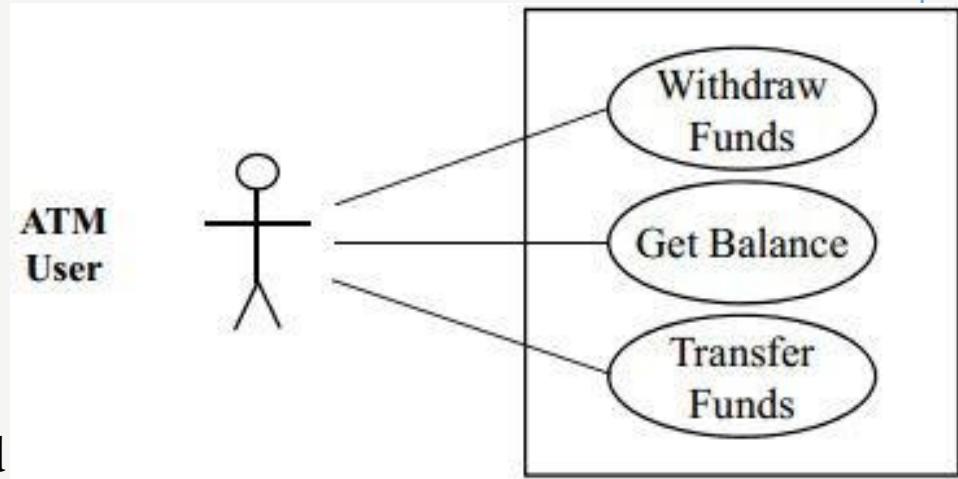
- Ex: connect to my DVD.com
- Ex: go to “search” page.

Use cases are deceptively simple tools for describing the behavior of software or systems.

A use case contains a textual description of all of the ways which the intended users could work with the software or system.

Example of ATM

- Use cases are commonly elaborated (or documented)
- Elaboration is first written textually
 - Details of operation
 - Alternatives model choices and conditions during execution
- **Actors:** Humans or software components that use the software being modeled
- **Use cases:** Shown as circles or ovals
- **Node Coverage:** Try each use case once ...
- **Use Case graphs, by themselves, are not useful for testing**



Software Requirement Specification

- A software requirements specification (SRS) is a complete description of the behavior of the system to be developed.
- It includes a set of use cases that describe all of the interactions that the users will have with the software.
- Use cases are also known as functional requirements. In addition to use cases, the SRS also contains nonfunctional (or supplementary) requirements.
- Non-functional requirements are requirements which impose constraints on the design or implementation (such as performance requirements, quality standards, or design constraints).
- Recommended approaches for the specification of software requirements are described by **IEEE 830-1998**.
- This standard describes possible structures, desirable contents, and qualities of a software requirements specification.

Types of Requirements

Requirements are categorized in several ways. The following are common categorizations of requirements that relate to technical management:

- Customer Requirements
- Functional Requirements
- Non-Functional Requirements

Customer Requirements

The customers are those that perform the eight primary functions of systems engineering, with special emphasis on the operator as the key customer. Operational requirements will define the basic need and, at a minimum, answer the questions posed in the following listing:

- Operational distribution or deployment: Where will the system be used?
- Mission profile or scenario: How will the system accomplish its mission objective?
- Performance and related parameters: What are the critical system parameters to accomplish the mission?
- Utilization environments: How are the various system components to be used?
- Effectiveness requirements: How effective or efficient must the system be in performing its mission?
- Operational life cycle: How long will the system be in use by the user?
- Environment: What environments will the system be expected to operate in an effective manner?

Functional Requirements

Functional Requirements are very important system requirements in the system design process. These requirements are the technical specifications, system design parameters and guidelines, data manipulation, data processing, and calculation modules etc, of the proposed system.

For Example: The following are the requirements of Google Email Service

- The system shall support the ability to receive emails
- The system shall support the ability to send emails
- The system shall support the ability to create new folders
- The system shall support the ability to filter emails in different folders
- The system shall support the ability to attach different kind of attachments
- The system shall support the ability to create and maintain address book
- The system shall support the ability to create unlimited user accounts with different email addresses

Non-Functional Requirements

Non-functional requirements are requirements that specify criteria that can be used to judge the operation of a system, rather than specific behaviors. Non-functional requirements are qualities or standards that the system under development must have or comply with, but which are not tasks that will be automated by the system.

Example non-functional requirements for a system include:

- system must be built for a total installed cost of \$1,050,000.00
- system must run on Windows Server 2003
- system must be secured against Trojan attacks

A software development methodology helps to identify, document, and realize the requirements. Non functional requirements can be divided into following categories:

- Usability
- Reliability
- Performance
- Security

Product & Project Based Application

Before we delve into the differences, for a better clarity, I would like to explain what is a software product and a software project.

Software product: A software application that is **developed by a company with its own budget**. The requirements are driven by market surveys. The developed product is then sold to different customers with licenses.

- Example for software products: Tally (by TCS), Acrobat reader/writer (Adobe), Internet Explorer (MS), Finale (Infosys), Windows (MS), QTP (HP) etc.

Software project: A software application that is **developed by a company with the budget from a customer**. In fact, the customer gives order to develop some software that helps him in his business. Here, the requirements come from the customer.

- Example for software projects: A separate application ordered by a manufacturing company to maintain its office inventory etc.

Object Oriented Programming



- Programming is like writing.
- If you can write a demonstration, you can make a program.
- So, programming is also easy.
- But, actually, programming is not so easy, because a real good program is not easily programmed. It needs the programmers' lots of wisdom, lots of knowledge about programming and lots of experience.
- It is like writing, to be a good writer needs lots of experience and lots of knowledge about the world.
- Learning and practise is necessary

Object-Oriented Languages

An object-based programming language is one which easily supports object-orientation.

Smalltalk : 1972-1980

- Founder of Alan Kay

C++ : 1986, Bjarne Stroustrup

Java(Oak) : 1992 (Smalltalk + C++)

- Founder of James Gosling

- Developed by Sun Microsystem overtake by Oracle.

C# :

- Developed at Microsoft by Anders Hejlsberg et al, 2000

- Event driven, object oriented, visual programming language (C++ and Java)

Others:

- Effile, Objective-C, Ada, ...

What is OOP?



- Identifying **objects** and assigning **responsibilities** to these objects.
- Objects communicate to other objects by sending **messages**.
- Messages are received by the **methods** of an object
- **An object is like a black box.**
- **The internal details are hidden.**
- Object is derived from abstract data type
- Object-oriented programming has a web of interacting objects, each house-keeping its own state.
- Objects of a program interact by sending messages to each other.

Everything in the world is an object

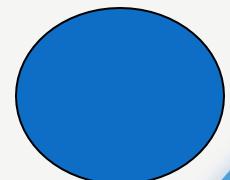
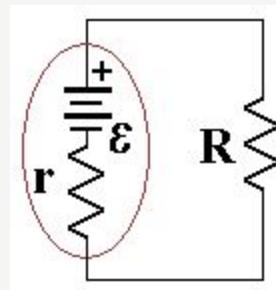
- A flower, a tree, an animal
- A student, a professor
- A desk, a chair, a classroom, a building
- A university, a city, a country
- The world, the universe
- A subject such as CS, IS, Math, History, ...

Concepts of OO

- Object
- Class
- Encapsulation
- Inheritance
- Polymorphism
 - Overriding
 - Overloading
- Abstraction

What is an object?

- Tangible Things as a car, printer, ...
- Roles as employee, boss, ...
- Incidents as flight, overflow, ...
- Interactions as contract, sale, ...
- Specifications as colour, shape, ...



So, what are objects?

An object represents an individual, identifiable item, unit, or entity, either real or abstract, with a well-defined role in the problem domain.

An "object" is anything to which a concept applies.

This is the basic unit of object oriented programming(OOP).

That is both data and function that operate on data are bundled as a unit called as object.



The two parts of an object

Object = Data + Methods

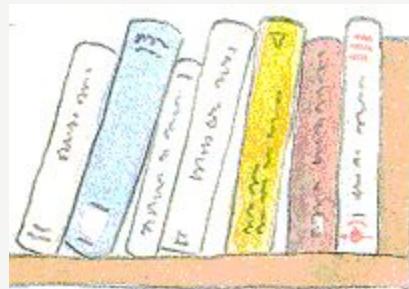
or

to say the same differently

An object has the responsibility to *know* and the responsibility to *do*.



=



+



Class

- When you define a class, you define a **blueprint for an object**.
- This doesn't actually define any data, but it does define what the class name means, that is, what an object of the class will consist of and what operations can be performed on such an object.
- A class represents an **abstraction of the object and abstracts the properties and behavior of that object**.
- Class can be considered as the blueprint or definition or a template for an object and describes the properties and behavior of that object, but without any actual existence.
- **An object is a particular instance of a class** which has actual existence and there can be many objects (or instances) for a class.
- In the case of a car or laptop, there will be a blueprint or design created first and then the actual car or laptop will be built based on that.
- We do not actually buy these blueprints but the actual objects.

The two steps of Object Oriented Programming

- **Making Classes:** Creating, extending or reusing abstract data types.
- **Making Objects interact:** Creating objects from abstract data types and defining their relationships.

Encapsulation

Encapsulation is the practice of including in an object everything it needs hidden from other objects. The internal state is usually not accessible by other objects.

Encapsulation is placing the data and the functions that work on that data in the same place. While working with procedural languages, it is not always clear which functions work on which variables but object-oriented programming provides you framework to place the data and the relevant functions together in the same object.

Encapsulation in Java is the process of wrapping up of data (properties) and behavior (methods) of an object into a single unit; and the unit here is a Class (or interface).

Encapsulate in plain English means *to enclose or be enclosed in or as if in a capsule*. In Java, a class is the capsule (or unit).

Encapsulation(Cont...)

- In Java, everything is enclosed within a class or interface, unlike languages such as C and C++, where we can have global variables outside classes.

- Encapsulation enables **data hiding**, hiding irrelevant information from the users of a class and exposing only the relevant details required by the user.

- We can expose our operations hiding the details of what is needed to perform that operation.

- We can protect the internal state of an object by hiding its attributes from the outside world (*by making it private*), and then exposing them through setter and getter methods. Now modifications to the object internals are only controlled through these methods.

Abstraction

Abstraction is the representation of the essential features of an object. These are ‘encapsulated’ into an *abstract data type*.

Data abstraction refers to, providing only essential information to the outside world and hiding their background details, i.e., to represent the needed information in program without presenting the details.

For example, a database system hides certain details of how data is stored and created and maintained.

Similar way, C++ classes provides different methods to the outside world without giving internal detail about those methods and data.

In plain English, abstract means a concept or idea not associated with any specific instance and does not have a concrete existence.

Abstraction(Cont...)

- Abstraction in Object Oriented Programming refers to the ability to make a class abstract.
- Abstraction captures only those details about an object that are relevant to the current perspective.
- Abstraction tries to reduce and factor out details so that the programmer can focus on a few concepts at a time. Java provides interfaces and abstract classes for describing abstract types.
- An **interface** is a contract or specification without any implementation. An interface can't have behavior or state.
- An **abstract class** is a class that cannot be instantiated. All other functionality of the class still exists. Abstract classes can have state and can be used to provide a skeletal implementation.
- A detailed comparison of interfaces and abstract classes can be found at **interface vs abstract-class**.

Polymorphism

- Polymorphism means “having many forms”.
- It allows different objects to respond to the same message in different ways, the response specific to the type of the object.
- The most important aspect of an object is its **behaviour** (the things it can do). A behaviour is initiated by **sending a message** to the object (usually by calling a method).
- The ability to use an operator or function in different ways in other words giving different meaning or functions to the operators or functions is called polymorphism.



Polymorphism

- Poly refers to many. That is a single function or an operator functioning in many ways different upon the usage is called polymorphism.
- E.g. the message *displayDetails()* of the Person class should give different results when send to a Student object (e.g. the enrolment number).

The ability to change form is known as polymorphism.

- There are two types of polymorphism in Java

- Compile time polymorphism(Overloading)
- Runtime polymorphism(Overriding)



Overloading

- The concept of overloading is also a branch of polymorphism. When the exiting operator or function is made to operate on new data type, it is said to be overloaded.

- The **same method name (method overloading) or operator symbol (operator overloading)** can be used in different contexts.

- In method overloading, **multiple methods having same name can appear in a class, but with different signature.**

- And based on the number and type of arguments we provide while calling the method, the correct method will be called.

- Java doesn't allow operator overloading yet + is overloaded for class String. The '+' operator can be used for addition as well as string concatenation.

Inheritance

Inheritance means that one class inherits the characteristics of another class. This is also called a “is a” relationship

One of the most useful aspects of object-oriented programming is code reusability. As the name suggests Inheritance is the process of forming a new class from an existing class that is from the existing class called as base class, new class is formed called as derived class.

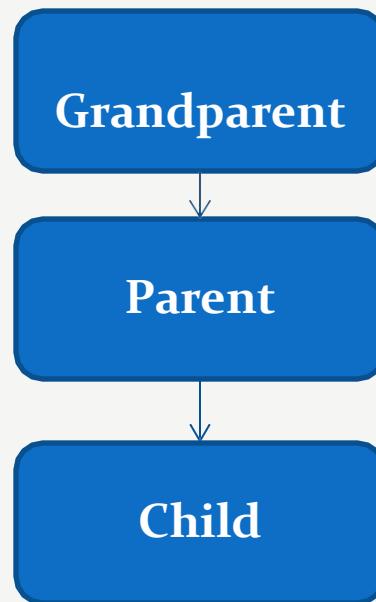
This is a very important concept of object-oriented programming since this feature helps to reduce the code size.

Inheritance describes the relationship between two classes. A class can get some of its characteristics from a parent class and then add unique features of its own.

Inheritance

In general, Java supports single-parent, multiple-children inheritance and multilevel inheritance (Grandparent-> Parent -> Child) for classes and interfaces. Java supports multiple inheritances (multiple parents, single child) only through interfaces.

In a class context, inheritance is referred to as implementation inheritance, and in an interface context, it is also referred to as interface inheritance.



Inheritance(Cont...)

For example consider a Vehicle parent class and its child class Car.

- Vehicle class will have all common properties and functionalities for all vehicles in common and Car will inherit those common properties from the Vehicle class and then add those properties which are specific to a car.
- Here, Vehicle is known as base class, parent class, or super class.
- Car is known as derived class, Child class or subclass.

A car *is a* vehicle

A dog *is an* animal

A teacher *is a* person

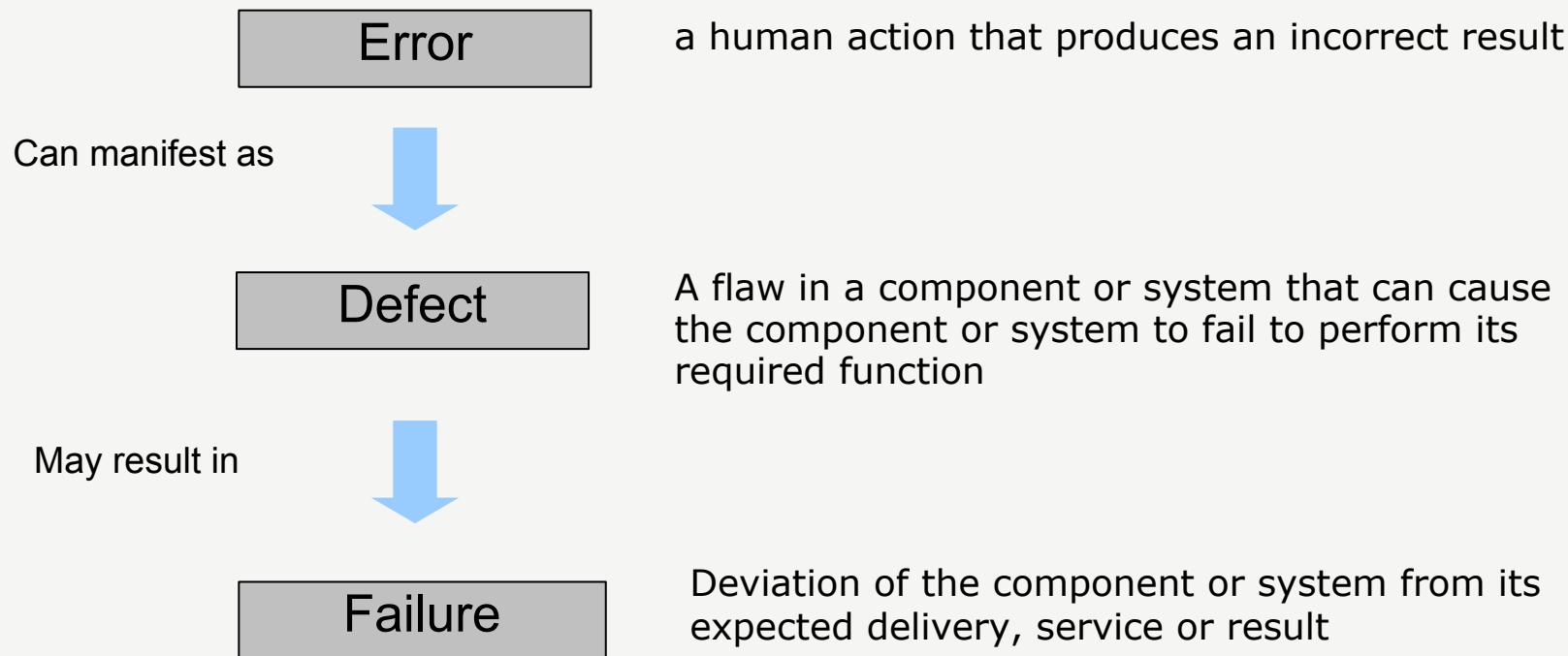
Module - 2 [Manual Testing]

Agenda

- Basic Concept
- Test Process Documentation
- STLC
- Level of Testing
 - Unit Testing
 - Integration Testing
 - System Testing
 - Functional & Non Functional Testing
 - Regression Testing
 - User Acceptance Testing
- Maintenance Testing
- Test Case Authoring / Static Testing
- Build Release Process
- Defect Reporting & Tracking
- Agile Methods & Approaches
- Bug Tracking Tools JIRA/Bugzilla

Error , Defect , Bug and Failure

Errors, Defects and Failures



Errors, Defects and Failures

“A mistake in coding is called error, error found by tester is called defect, defect accepted by development team then it is called bug, build does not meet the requirements then it is failure”

Error: A discrepancy between a computed, observed, or measured value or condition and the true, specified, or theoretically correct value or condition. This can be a misunderstanding of the internal state of the software, an oversight in terms of memory management, confusion about the proper way to calculate a value, etc.

Errors, Defects and Failures

Failure: The inability of a system or component to perform its required functions within specified performance requirements. See: bug, crash, exception, and fault.

Bug: A fault in a program which causes the program to perform in an unintended or unanticipated manner. See: anomaly, defect, error, exception, and fault. Bug is terminology of Tester.

Fault: An incorrect step, process, or data definition in a computer program which causes the program to perform in an unintended or unanticipated manner. See: bug, defect, error, exception.

Defect: Commonly refers to several troubles with the software products, with its external behavior or with its internal features.

Types of Errors

- User Interface Errors
- Error Handling
- Boundary related errors
- Calculation errors
- Initial and Later states
- Control flow errors
- Errors in Handling or Interpreting Data
- Race Conditions
- Load Conditions
- Hardware
- Source, Version and ID Control
- Testing Errors

The Role of Testing

Rigorous testing of systems and documentation can:

- reduce the risk of problems occurring in an operational environment
- contribute to the quality of the software system

How?

- By finding and correcting defects before the system is released for operational use
- Software testing may also be required to meet contractual or legal requirements, or industry-specific standards

Quality

Quality

- Quality - '**The degree to which a component, system or process meets specified requirements and/or user/customer needs and expectations'**'

- Defects covering:

- functional software requirements and characteristics
- and non-functional software requirements and characteristics (e.g. reliability, usability, efficiency, portability and maintainability)

- Testing can give confidence in the Quality of the software if it finds few or no defects

- Quality software is reasonably **bug or defect free**, delivered on time and within budget, meets requirements and/or expectations, and is maintainable.

- ISO 8402-1986 standard defines quality as "**the totality of features and characteristics of a product or service that bears its ability to satisfy stated or implied needs.**"

Quality(Cont...)

- Key aspects of quality for the customer include:

- Good design – looks and style
- Good functionality – it does the job well
- Reliable – acceptable level of breakdowns or failure
- Consistency
- Durable – lasts as long as it should
- Good after sales service
- Value for money

Following are two cases that demonstrate the importance of software quality

- Failure due to error in a transfer of information between a team in Colorado and a team in California
- One team used English units (e.g., inches, feet and pounds) while the other used metric units for a key spacecraft operation.

Risk

Risk

- A properly designed test that passes, reduces the overall level of Risk in a system
- Risk – **'A factor that could result in future negative consequences; usually expressed as impact and likelihood'**
- When testing does find defects, the Quality of the software system increases when those defects are fixed
- The Quality of systems can be improved through Lessons learned from previous projects
- Analysis of root causes of defects found in other projects can lead to Process Improvement
- Process Improvement can prevent those defects reoccurring
- Which in turn, can improve the Quality of future systems
- Testing should be integrated as one of the Quality assurance activities

Types of Risk

A Risk could be any future event with a negative consequence

You

need to identify the risks associated with your project

Risks are of two types

- Project Risks
- Product Risk

Types of Risk Examples

Example of **Project risk** is Senior Team Member leaving the project abruptly.

- Every risk is assigned a likelihood i.e. chance of it occurring, typically on a scale of 1 to 10. Also the impact of that risk is identified on a scale of 1-10 .
- But just identifying the risk is not enough. You need to identify mitigation. In this case mitigation could be Knowledge Transfer to other team members & having a buffer tester in place

Example of **product risks** would be Flight Reservation system not installing in test environment

- Mitigation in this case would be conducting a smoke or sanity testing. Accordingly you will make changes in your scope items to include sanity testing

Test Organization

Who does Testing?

- It depends on the process and the associated stakeholders of the project(s).
- In the IT industry, large companies have a team with responsibilities to **evaluate the developed software in the context of the given requirements.**
- Moreover, **developers also conduct testing which is called Unit Testing.**
- In most cases, following professionals are involved in testing of a system within their respective capacities:

- **Software Tester**
- **Software Developer**
- **Project Lead/Manager**
- **End User**

Different companies have different designations for people who test the software on the **basis of their experience and knowledge such as Software Tester, Software Quality Assurance Engineer, and QA Analyst** etc.

- It is not possible to test the software at any time during its cycle.
- The next two sections state when testing should be started and when to end it during the SDLC.

What do Tester?

- Make up the majority of the resources in the testing group.
- May be specialists in a particular area
 - Automation
 - Performance
 - Usability
 - Security
- Or alternatively may work more generally doing:
 - Test Analysis
 - Test Design
 - Test Execution
 - Test Environment management
 - Test Data management
- Typically testers at the component level are developers
- At the Acceptance level testers are typically business experts or users or operators (for operational acceptance testing)

Role of Software Tester

Apart from exposing faults (“bugs”) in a software product confirming that the program meets the program specification, as a test engineer you need to create test cases, procedures, scripts and generate data.

You execute test procedures and scripts, analyze standards and evaluate results of system/integration/regression testing. You also...

- Speed up development process by identifying bugs at an early stage (e.g. specifications stage)
- Reduce the organization's risk of legal liability
- Maximize the value of the software
- Assure successful launch of the product, save money, time and reputation of the company by discovering bugs and design flaws at an early stage before failures occur in production, or in the field
- Promote continual improvement



Test Planning Template Or Test Planning Process

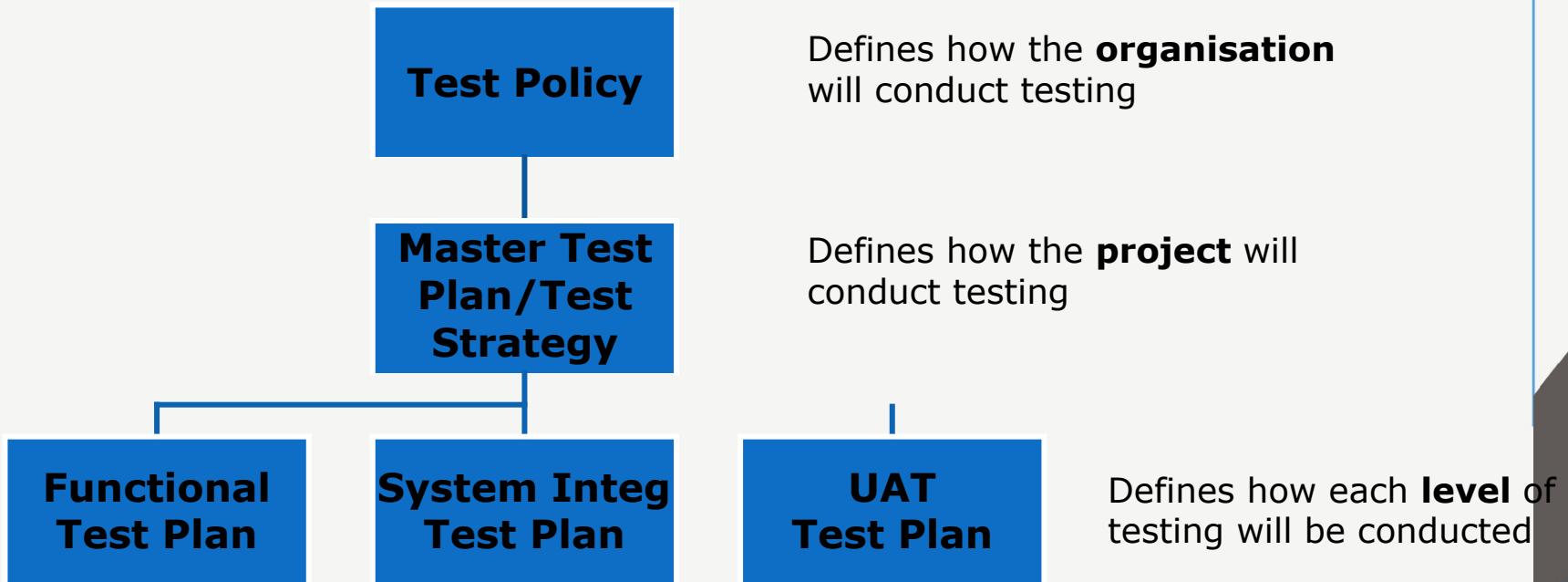
Test Planning

A document describing the scope, approach, resources and schedule of intended test activities

- Determining the **scope and risks, and identifying** the objectives of testing.
- Defining the overall approach of testing (the test strategy), including the definition of the test levels and entry and exit criteria.
- Integrating and coordinating the testing activities into the software life cycle activities:
 - **acquisition, supply, development, operation and maintenance.**
- Making decisions about what to test, what roles will perform the test activities, how the test activities should be done, and how the test results will be evaluated?
- Scheduling test analysis and design activities.
- Scheduling test implementation, execution and evaluation.
- Assigning resources for the different activities defined
 - **Defining the amount, level of detail, structure and templates for the test documentation.**

Test Plan & Strategy

- All projects require a set of plans and strategies which define how the testing will be conducted.
- There are number of levels at which these are defined:



Test Planning Factors

- Factors which affect test planning

- The organisation's test policy
- Scope of the testing being performed
- Testing objectives
- Project Risks – e.g. business, technical, people
- Constraints – e.g. business imposed, financial, contractual etc
- Criticality (e.g. system/component level)
- Testability
- Availability of resources

- Test plans are continuously refined

- As more information becomes available
- As new risks arise or others are mitigated
- Not set in concrete, but changes must be carefully managed

Test Planning Activities

- **Approach:** Defining the overall approach of testing (the test strategy), including the definition of the test levels and entry and exit criteria.
- **Integrating and coordinating the testing activities into the software life cycle activities:** acquisition, supply, development, operation and maintenance.
- Making decisions about:
 - **what** to test
 - **who** do testing? i.e. what roles will perform the test activities
 - **when** and how the test activities should be done and when they should be stopped (exit criteria – see next slides)
 - **how** the test results will be evaluated
- Assigning resources for the different tasks defined.
- **Test ware:** Defining the amount, level of detail, structure and templates for the test documentation.
- Selecting metrics for monitoring and controlling test preparation and execution, defect resolution and risk issues.
- **Process:** Setting the level of detail for test procedures in order to provide enough information to support reproducible test preparation and execution.

Exit Criteria

How do we know when to stop testing?

- Run out of time?
- Run out of budget?
- The business tells you it went live last night!
- Boss says stop?
- All defects have been fixed?
- When our exit criteria have been met?

Purpose of exit criteria is to define when we STOP testing either at the:

- End of all testing – i.e. product Go Live
- End of phase of testing (e.g. hand over from System Test to UAT)

Exit Criteria typically measures:

- Thoroughness measures, such as coverage of requirements or of code or risk coverage
- Estimates of defect density or reliability measures. (e.g. how many defects open by category)
- Cost.
- Residual Risks, such as defects not fixed or lack of test coverage in certain areas.
- Schedules - such as those based on time to market.

QA vs QC vs Testing

S.N.	Quality Assurance	Quality Control	Testing
1	Activities which ensure the implementation of processes, procedures and standards in context to verification of developed software and intended requirements.	Activities which ensure the verification of developed software with respect to documented (or not in some cases) requirements.	Activities which ensure the identification of bugs/error/defects in the Software.
2	Focuses on processes and procedures rather than conducting actual testing on the system.	Focuses on actual testing by executing Software with intend to identify bug/defect through implementation of procedures and process.	Focuses on actual testing.
3	Process oriented activities.	Product oriented activities.	Product oriented activities.
4	Preventive activities.	It is a corrective process.	It is a preventive process.
5	It is a subset of Software Test Life Cycle (STLC).	QC can be considered as the subset of Quality Assurance.	Testing is the subset of Quality Control.

How much Testing is Enough?

Deciding how much testing is enough should take account of:

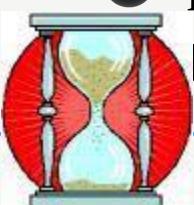
- the level of Risk
- project constraints such as time and budget

Risks should be evaluated at the Business Level, Technological Level, Project Level and Testing Level

Risks are also used to decide where to start testing and where more testing is needed

Risk considerations can include:

- financial implication of software being released that is unreliable
(support costs / possible legal action)
- software being delivered late to market
- potential loss of Life (safety critical systems)
- potential loss of face (may have implications as well)



How much Testing is Enough?

- Risk analysis should be used to determine what to test in each component and just as importantly what not to test

- For example, an unacceptable risk would say we must test, an acceptable one perhaps not to test

- Testing is a risk-control activity that provides feedback to the stakeholders

- With this feedback the stakeholders can make informed decisions about the release of the software (or system) being tested

- More about Risks later in the Course

- **Exit criteria** is used to determine when testing at any stage is complete

The set of generic and specific conditions, agreed upon with the stakeholders, for permitting a process to be officially completed

- Exit criteria may be defined in terms of :

- Thoroughness – i.e. coverage or requirements
- cost or time constraints
- percentage of tests run without incident
- number of faults remaining

Testing v/s Debugging

The responsibility for each activity is very different, i.e.

- Testers test
- Developers debug

Testing

- It involves the identification of bug/error/defect in the software without correcting it.
- Normally professionals with a Quality Assurance background are involved in the identification of bugs. Testing is performed in the testing phase.
- Testing can show failures that are caused by defects

Debugging

- It involves identifying, isolating and fixing the problems/bug. Developers who code the software conduct debugging upon encountering an error in the code.
- Debugging is the part of White box or Unit Testing.
- Debugging can be performed in the development phase while conducting Unit Testing or in phases while fixing the reported bugs
- Debugging identifies the cause of a defect, repairs the code and checks that the defect has been fixed correctly

Test Process Documents

Introduction(Cont...)

Test Analysis

Test Plan/Strategy

Test Script/Test Step/Test
Procedure

Test Scenario

Test Case

● Test Condition

● Test Procedure Specification

Traceability

Tractability Matrix

Test Analysis

- Test analysis is the process of looking at something that can be used to derive test information. This basis for the tests is called the **test basis**.
- The test basis is the information we need in order to start the test analysis and create our own test cases. Basically it's a documentation on which test cases are based, such as requirements, design specifications, product risk analysis, architecture and interfaces.
- We can use the test basis documents to understand what the system should do once built. The test basis includes whatever the tests are based on. Sometimes tests can be based on experienced user's knowledge of the system which may not be documented.

Test Analysis(Cont...)

From testing perspective we look at the test basis in order to see what could be tested. These are the test conditions. A **test condition** is simply something that we could test.

While identifying the test conditions we want to identify as many conditions as we can and then we select about which one to take forward and combine into test cases. We could call them **test possibilities**.

The test conditions that are chosen will depend on the test strategy or detailed test approach. For example, they might be based on risk, models of the system, etc.

Once we have identified a list of test conditions, it is important to prioritize them, so that the most important test conditions are identified.

Test conditions can be identified for **test data** as well as for test inputs and test outcomes, for example, different types of record, different sizes of records or fields in a record.

High Level Requirement

#	Name	Description
	Search	
100	DVD name	The system shall provide the ability for the user to search by a DVD Name
101	Actor name	The system shall provide the ability for the user to search by a Actor name
102	Categories within search	The system shall provide the ability for the user to search by categories within search
	User account	
200	Login	
201	Logout	
202	Forgot user name and password	
203	Create user account	

Formate :<https://github.com/TopsCode/Software-Testing>

Test Script

- **A set of sequential instruction that detail how to execute a core business function**

- One script is written to explain how to simulate each business scenario
- Written to a level of detail for which someone else (other than the script writer) would be able to easily execute
- Identifies the test condition that is being satisfied for each step, if applicable
- Identified the input/test data that should be entered for each transaction
- Identifies the expected results for each step, if applicable
- Should demonstrate how the system can support the HCA warehouse business processes

Test Script

- A test script in software testing is **a set of instructions that will be performed on the system under test to test that the system functions as expected.**

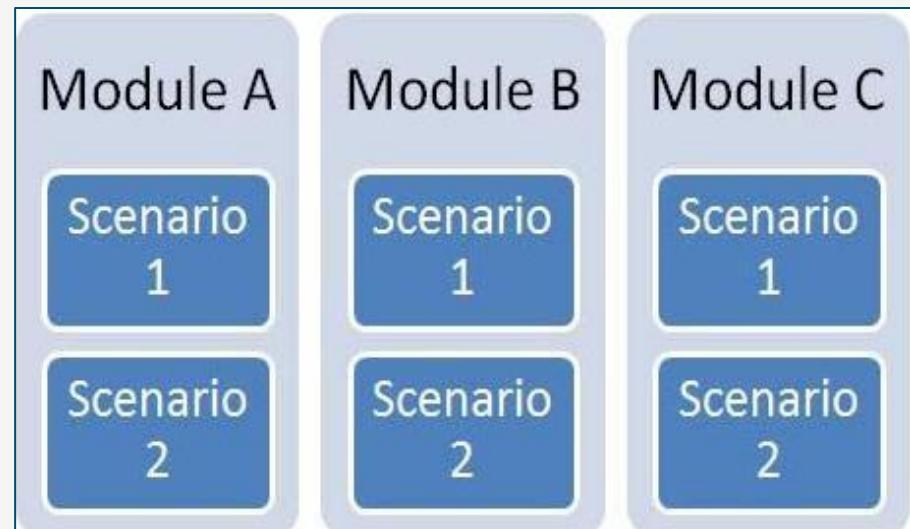
- There are various means for executing test scripts.

- **Manual Testing**
- **Automation Testing**

Test Scenario

A Scenario is any functionality that can be tested. It is also called Test Condition, or Test Possibility.

- Test Scenario is ‘What to be tested’
- Test scenario is nothing but test procedure.
- The scenarios are derived from use cases.
- Test Scenario represents a series of actions that are associated together.
- Scenario is thread of operations



Test Case

Test cases involve the set of steps, conditions and inputs which can be used while performing the testing tasks.

- Test Case is ‘How to be tested’
- Test case consist of set of input values, execution precondition, expected Results and executed post-condition developed to cover certain test Condition.
- Test cases are derived (or written) from test scenario.
- Test Case represents a single (low level) action by the user.
- Test cases are set of input and output given to the System.

Formate :<https://github.com/TopsCode/Software-Testing>

Test Case

Furthermore test cases are written to keep track of testing coverage of Software. Generally, there is no formal template which is used during the test case writing. However, following are the main components which are always available and included in every test case:

- Test case ID
- Product Module ID
- Product version (Optional)
- Revision history (Optional)
- Purpose/ Test Case Description
- Assumptions (Optional)
- Pre-Conditions(Optional)
- Test Steps
- Expected Outcome/Result
- Actual Outcome/Result
- Post Conditions(Pass/Fail)

Test Case(Cont...)

- The Step **# Identifies** the task sequence in the script

Action/Input Data

- The **Action Steps** details the task to be performed

- Write action steps in terms that support execution (action oriented)

- Level of detail should reflect core business function, not system navigation

- The **Input Data** describes what needs to be entered for a step (if applicable),
It's called **Test Data**.

- Include all search criteria (Ex. First name, last name)

Expected Results

- The Expected Results documents what results are anticipated for this step
- Include detail needed for business process (Ex. Required fields, external system interfaces)

Actual Results

- Where the “script executor” enters the result that occurred from this step
- Be specific

Test Case

- Developing test material can be split into two distinct stages:
 - Defining “what” needs to be tested
 - Defining “how” the system should be tested
- This process can vary from organisation to organisation, can be very formal or very informal with little documentation
- The more formal, the more repeatable the tests, but it does depend on the context of the testing being carried out
- The process of identifying test conditions and designing tests consists of the following steps:
 - **Identify and Defining Test Conditions**
 - **Specifying Test Cases**
 - **Specifying Test Procedures**
 - **Developing a Test Execution Schedule**

Test Procedures Specification (Test Script)

- The Test Procedures Specification **specifies the sequence of actions for a test**, i.e. one or more Test Cases

- It is also known as a **Test Script**

- The Test Script can be manual or automated

- Contents of a Test Procedure are:

- Test procedure specification identifier
- Purpose
- Special requirements
- Procedure steps

Tractability

Test conditions should be able to be linked back to their sources in the test basis, this is known as traceability.

Traceability can be horizontal through all the test documentation for a given test level (e.g. system testing, from test conditions through test cases to test scripts) or it can be vertical through the layers of development documentation (e.g. from requirements to components).

Tractability Matrix

- To protect against changes you should be able to **trace back from every system component** to the original requirement that caused its presence.

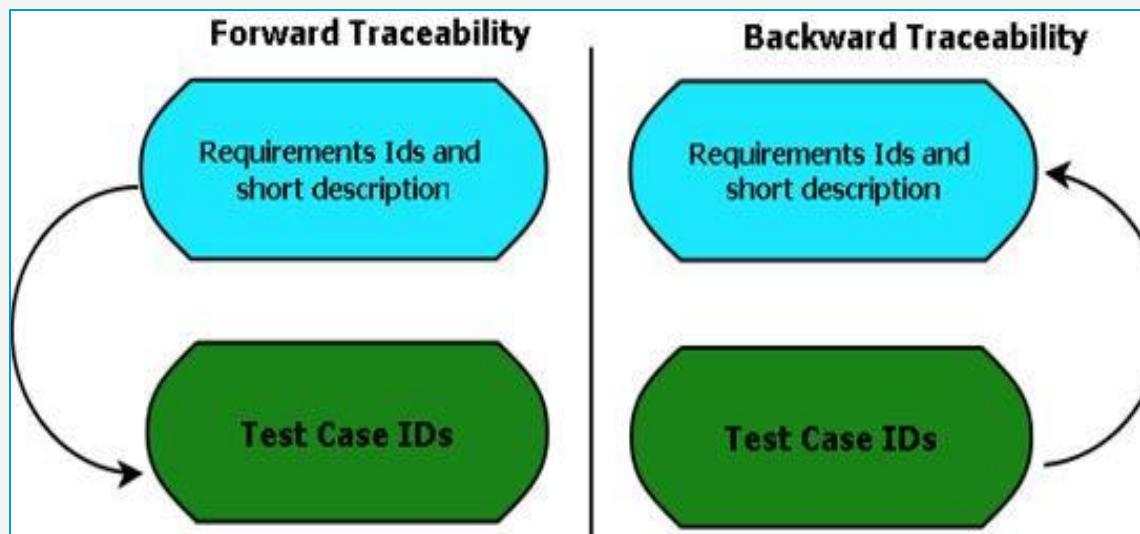
- A **software process** should help you keeping the virtual table up-to-date.

- Simple technique may be quite valuable (naming convention)

	Comp 1	Comp 2	:	:	:	:	:	:	Comp m
Req 1			x		x				
Req 2	x								x
...									
...		x			x		x		
...									
...		x							
...				x					
...								x	
Req n									

Types of Traceability Matrix

- Forward Traceability – Mapping of Requirements to Test cases
- Backward Traceability – Mapping of Test Cases to Requirements
- Bi-Directional Traceability - A Good Traceability matrix is the References from test cases to basis documentation and vice versa.



Pros of Traceability Matrix

- Make obvious to the client that the software is being developed as per the requirements.
- To make sure that all requirements included in the test cases
- To make sure that developers are not creating features that no one has requested
- Easy to identify the missing functionalities.
- If there is a change request for a requirement, then we can easily find out which test cases need to update.
- The completed system may have “Extra” functionality that may have not been specified in the design specification, resulting in wastage of manpower, time and effort.

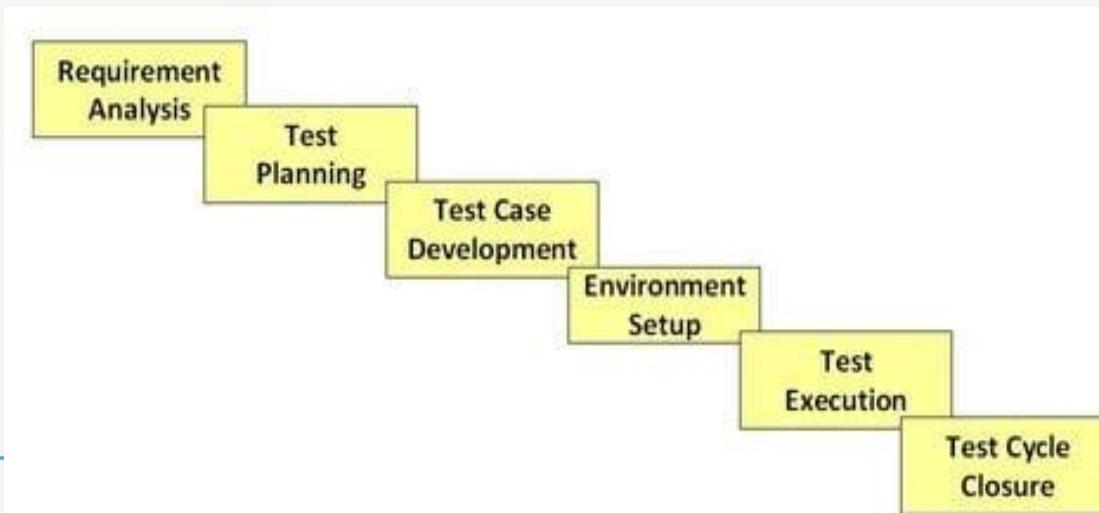
Cons of Traceability Matrix

- No traceability or Incomplete Traceability Results into:
- Poor or unknown test coverage, more defects found in production
- It will lead to miss some bugs in earlier test cycles which may arise in later test cycles. Then a lot of discussions arguments with other teams and managers before release.
- Difficult project planning and tracking, misunderstandings between different teams over project dependencies, delays, etc

STLC – Software Testing Life Cycle

STLC Phases

1. Requirement Analysis
2. Test Planning
3. Test case development
4. Test Environment setup
5. Test Execution
6. Test Cycle closure



What is Entry and Exit Criteria in STLC?

- **Entry Criteria:** Entry Criteria gives the prerequisite items that must be completed before testing can begin.
- **Exit Criteria:** Exit Criteria defines the items that must be completed before testing can be concluded
- **In an Ideal world, you will not enter the next stage until the exit criteria for the previous stage is met. But practically this is not always possible.**

Requirement Analysis

Requirement Phase Testing also known as Requirement Analysis in which test team studies the requirements from a testing point of view to identify testable requirements and the QA team may interact with various stakeholders to understand requirements in detail.

- Requirements could be either functional or non-functional.
- Automation feasibility for the testing project is also done in this stage.

Activities in Requirement Phase Testing

- Identify types of tests to be performed.
- Gather details about testing priorities and focus.
- Prepare Requirement Traceability Matrix (RTM).
- Identify test environment details where testing is supposed to be carried out.
- Automation feasibility analysis (if required).

Deliverables of Requirement Phase Testing

- RTM
- Automation feasibility report. (if applicable)

Test Planning

- **Test Planning in STLC** is a phase in which a Senior QA manager determines the test plan strategy along with efforts and cost estimates for the project.
- Moreover, the resources, test environment, test limitations and the testing schedule are also determined.
- The Test Plan gets prepared and finalized in the same phase.

Activities in Requirement Phase Testing

- Preparation of test plan/strategy document for various types of testing
- Test tool selection
- Test effort estimation
- Resource planning and determining roles and responsibilities.
- Training requirement

Deliverables of Requirement Phase Testing

- Test plan /strategy document.
- Effort estimation document.

Test Case Development

The **Test Case Development Phase** involves the creation, verification and rework of test cases & test scripts after the test plan is ready.

Initially, the Test data is identified then created and reviewed and then reworked based on the preconditions.

Then the QA team starts the development process of test cases for individual units.

Activities in Requirement Phase Testing

- Create test cases, automation scripts (if applicable)
- Review and baseline test cases and scripts
- Create test data (If Test Environment is available)

Deliverables of Requirement Phase Testing

- Test cases/scripts
- Test data

Test Environment Setup

- **Test Environment Setup** decides the software and hardware conditions under which a work product is tested.
- It is one of the critical aspects of the testing process and can be done in parallel with the Test Case Development Phase.
- Test team may not be involved in this activity if the development team provides the test environment.
- The test team is required to do a readiness check (smoke testing) of the given environment.

Activities in Requirement Phase Testing

- Understand the required architecture, environment set-up and prepare hardware and software requirement list for the Test Environment.
- Setup test Environment and test data
- Perform smoke test on the build

Deliverables of Requirement Phase Testing

- Environment ready with test data set up
- Smoke Test Results.

Test Execution

- **Test Execution Phase** is carried out by the testers in which testing of the software build is done based on test plans and test cases prepared.
- The process consists of test script execution, test script maintenance and bug reporting.
- If bugs are reported then it is reverted back to development team for correction and retesting will be performed.

Activities in Requirement Phase Testing

- Execute tests as per plan
- Document test results, and log defects for failed cases
- Map defects to test cases in RTM
- Retest the Defect fixes
- Track the defects to closure

Deliverables of Requirement Phase Testing

- Completed RTM with the execution status
- Test cases updated with results
- Defect reports

Test Cycle Closure

Test Cycle Closure phase is completion of test execution which involves several activities like test completion reporting, collection of test completion matrices and test results.

- Testing team members meet, discuss and analyze testing artifacts to identify strategies that have to be implemented in future, taking lessons from current test cycle.

- The idea is to remove process bottlenecks for future test cycles.

Activities in Requirement Phase Testing

- Evaluate cycle completion criteria based on Time, Test coverage, Cost, Software, Critical Business Objectives, Quality
- Prepare test metrics based on the above parameters.
- Document the learning out of the project
- Prepare Test closure report

Deliverables of Requirement Phase Testing

- Test Closure report
- Test metrics

Psychology of Testing

Relation Between Tester & Developer

- The testing and reviewing of the applications are different from the analyzing and developing of it.
- By this we mean to say that if we are building or developing applications we are working positively to solve the problems during the development process and to make the product according to the user specification.
- However while testing or reviewing a product we are looking for the defects or failures in the product.
- Thus building the software requires a different mindset from testing the software.

Relation Between Tester & Developer

It does not mean that the tester cannot be the programmer, or that the programmer cannot be the tester, although they often are separate roles. In fact programmers are the testers.

They always test their component which they built. While testing their own code they find many problems so the programmers, architect and the developers always test their own code before giving it to anyone.

Self Testing & Independent Testing

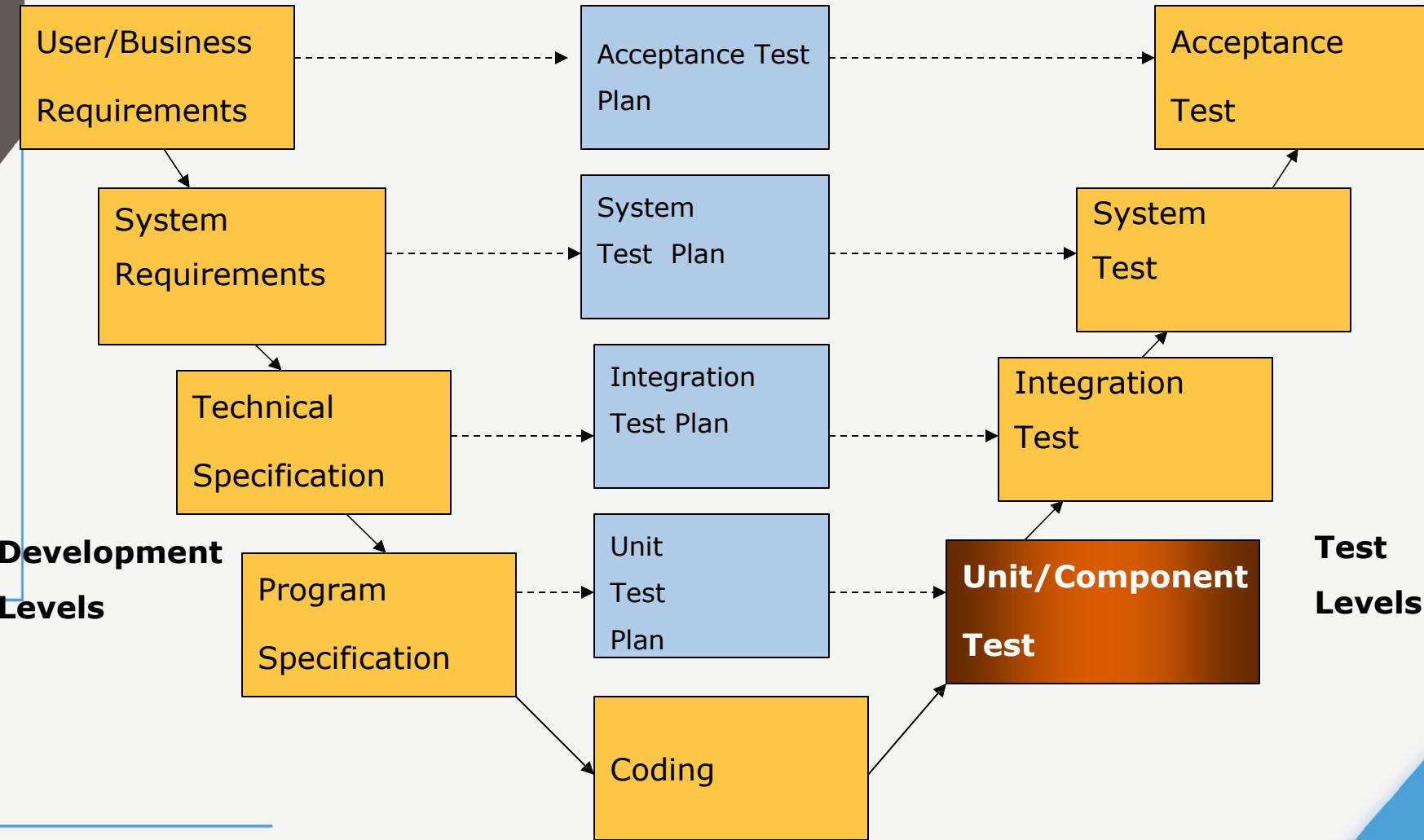
This degree of independence avoids author bias and is often more effective at finding defects and failures

There are several levels of independence in software testing which are listed here from the lowest level of independence to the highest:

- Tests by the person who wrote the item.
- Tests by another person within the same team, like another programmer.
- Tests by the person from some different group such as an independent test team.
- Tests by a person from a different organization or company, such as outsourced testing or certification by an external body.

Levels of Software Testing

Component Testing



Component (Unit) Testing

Component(Unit) – A minimal software item that can be tested in isolation. It means “A unit is the smallest testable part of software.”

Component Testing – The testing of individual software components.

Unit Testing is a level of the software testing process where **individual units/components of a software/system** are tested. The purpose is to validate that each unit of the software performs as designed.

Unit testing is the first level of testing and is performed prior to Integration Testing.

Sometimes known as **Unit Testing, Module Testing or Program Testing**

Component can be tested in isolation – stubs/drivers may be employed

Unit testing **frameworks, drivers, stubs and mock or fake objects** are used to assist in unit testing.

Functional and Non-Functional testing

Unit tests are typically written and run by software developers to ensure that code meets its design and behaves as intended with **debugging tool**.

Component Testing

- A unit is the smallest testable part of an application like functions/procedures, classes, interfaces.
- The goal of unit testing is to isolate each part of the program and show that the individual parts are correct.
- A unit test provides a strict, written contract that the piece of code must satisfy.
- As a result, it affords several benefits.
- Unit tests find problems early in the development cycle.
- **Unit testing is performed by using the White Box Testing method.**

Component Testing

Test Approach :

Test-First/Test-Driven approach – create the tests to drive the design and code construction!

Instead of creating a design to tell you how to structure your code, you create a test that defines how a small part of the system should function.

Three steps:

1. Design test that defines how you think a small part of the software should behave (Incremental development).
2. Make the test run as easily and quickly as you can. Don't be concerned about the design of code, just get it to work!
3. Clean up the code. Now that the code is working correctly, take a step back and re-factor to remove any duplication or any other problems that were introduced to get the test to run.

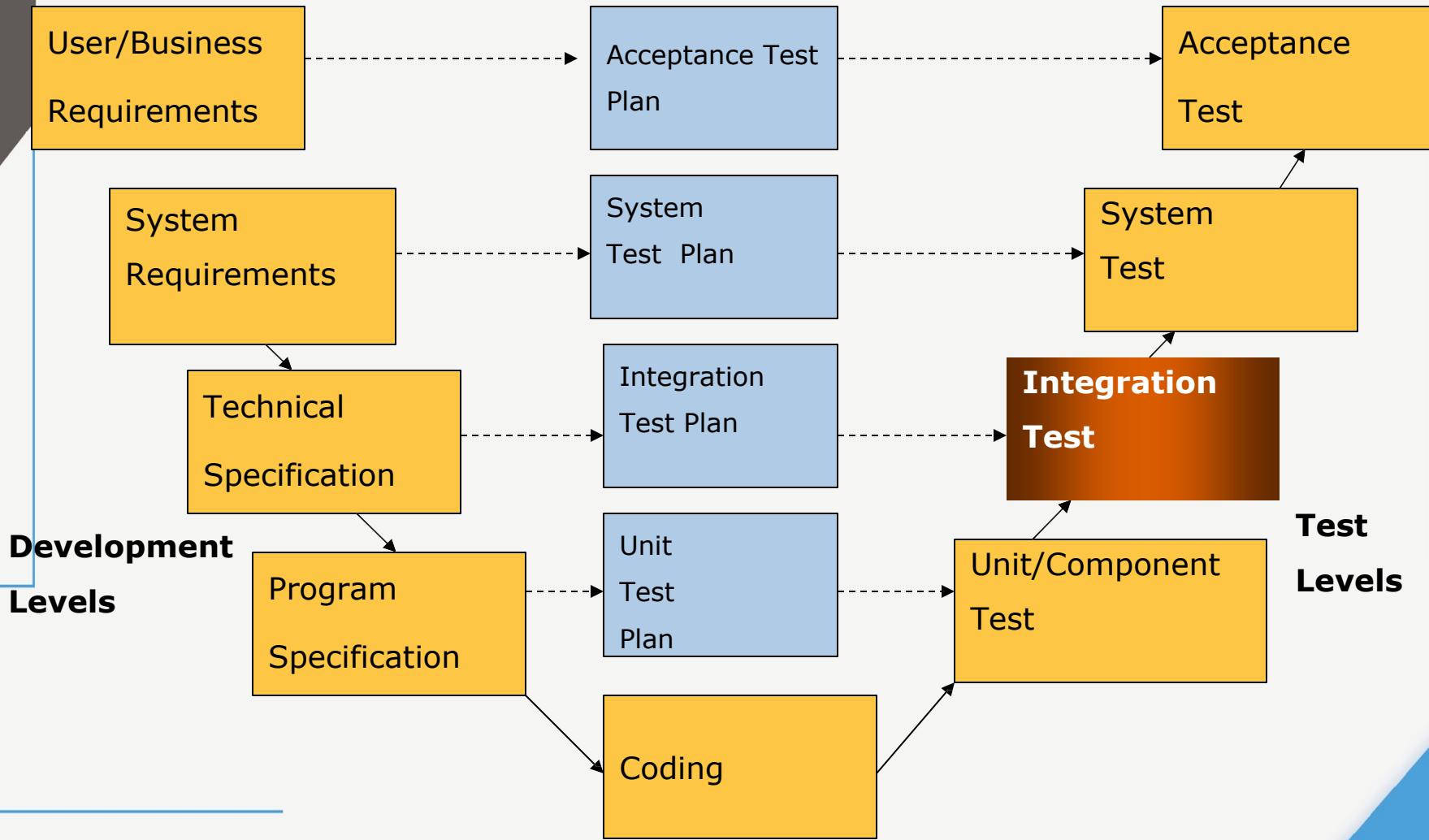
Component Testing

Unit testing in Extreme Programming involves the extensive use of testing frameworks. A unit test framework is used in order to create automated unit tests. Unit testing frameworks are not unique to extreme programming, but they are essential to it.

Below we look at some of what extreme programming brings to the world of unit testing:

- Tests are written before the code
- Rely heavily on testing frameworks
- All classes in the applications are tested
- Quick and easy integration is made possible

Integration Testing



Integration Testing

Integration Testing - Testing performed to expose defects in the interfaces and in the interactions between integrated components or systems

Integration Testing is a level of the software testing process where individual units are combined and tested as a group.

- The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing.

- Integration testing tests integration or interfaces between components, interactions to different parts of the system such as an operating system, file system and hardware or interfaces between systems.

- Integration testing is done by a specific integration tester or test team.

- Components may be code modules, operating systems, hardware and even complete systems

- There are 2 levels of Integration Testing

- Component Integration Testing**

- System Integration Testing**

Need of Integration Testing

- A Module in general is designed by an individual software developer who understanding and programming logic may differ from other programmers. Integration testing becomes necessary to verify the software modules work in unity

- At the time of module development, there wide chances of change in requirements by the clients. These new requirements may not be unit tested and hence integration testing becomes necessary.

- Interfaces of the software modules with the database could be erroneous

- External Hardware interfaces, if any, could be erroneous

- Inadequate exception handling could cause issues.

Component Integration Testing

Component Integration Testing: Testing performed to expose defects in the interfaces and interaction between integrated components

- Usually formal (records of test design and execution are kept)

- All individual components should be integration tested prior to system testing

- It tests the interactions between software components and is done after component testing.

- The software components themselves may be specified at different times by different specification groups, yet the integration of all of the pieces must work together.

- It is important to cover negative cases as well because components might make assumption with respect to the data.

- The following testing techniques are appropriate for Integration Testing:

- Functional Testing using** Black Box Testing techniques against the interfacing requirements for the component under test

- Non-functional Testing** (where appropriate, for *performance* or *reliability testing* of the component interfaces, for example)

System Integration Testing

- It tests the interactions between different systems and may be done after system testing.
- It verifies the proper execution of software components and proper interfacing between components within the solution.
- The objective of SIT Testing is to validate that all software module dependencies are functionally correct and that data integrity is maintained between separate modules for the entire solution.
- As testing for dependencies between different components is a primary function of SIT Testing, this area is often most subject to Regression Testing.

Integration Testing Methods

During the process of manufacturing a ballpoint pen, the cap, the body, the tail and clip, the ink cartridge and the ballpoint are produced separately and unit tested separately. When two or more units are ready, they are assembled and Integration Testing is performed. For example, whether the cap fits into the body or not.

Any of Black Box Testing, White Box Testing, and Gray Box Testing methods can be used. Normally, the method depends on your definition of ‘unit’.

There are two types of methods of Integration Testing:

- Bing Bang Integration Testing
- Incremental Integration Testing
 - Top Down Approach
 - Bottom Up Approach

When is Integration Testing performed?

● Integration Testing is performed after Unit Testing and before System Testing.

Who performs Integration Testing?

● Either Developers themselves or independent Testers perform Integration Testing.

Big Bang Integration Testing

In Big Bang integration testing all components or modules are integrated simultaneously, after which everything is tested as a whole.

Big Bang testing has the advantage that everything is finished before integration testing starts.

The major disadvantage is that in general it is time consuming and difficult to trace the cause of failures because of this late integration.

- Here all components are integrated together at **once**, and then tested.

Big Bang Integration Testing

Advantages:

- Convenient for small systems.

Disadvantages:

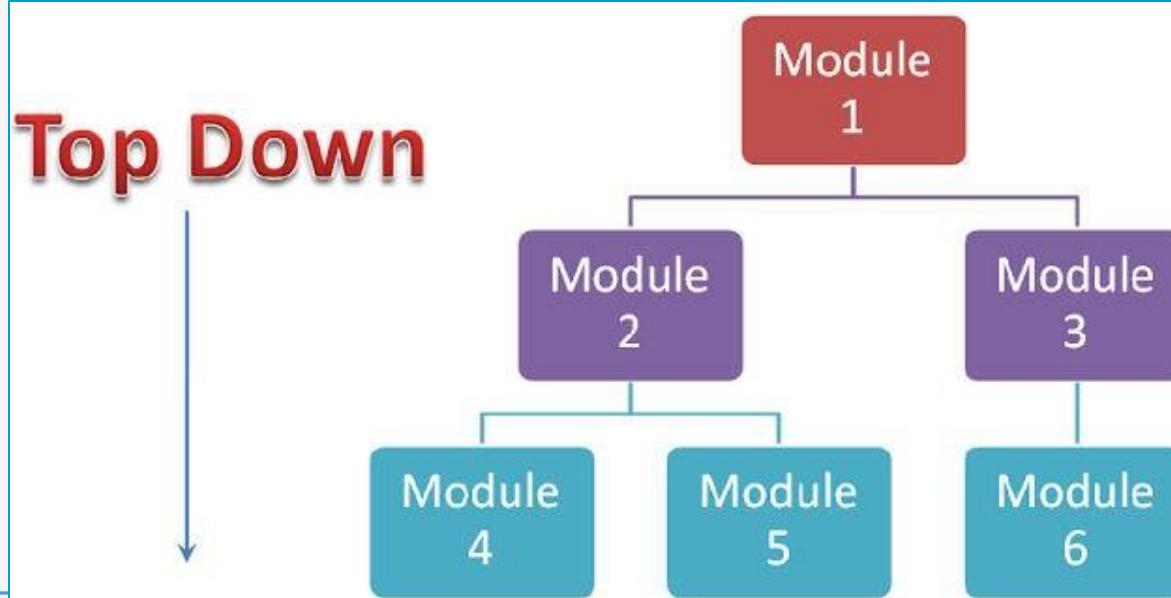
- Fault Localization is difficult.
- Given the sheer number of interfaces that need to be tested in this approach, some interfaces links to be tested could be missed easily.
- Since the integration testing can commence only after “all” the modules are designed, testing team will have less time for execution in the testing phase.
- Since all modules are tested at once, high risk critical modules are not isolated and tested on priority. Peripheral modules which deal with user interfaces are also not isolated and tested on priority.

Top Down Approach

Testing takes place from top to bottom, following the control flow or architectural structure (e.g. starting from the GUI or main menu). Components or systems are substituted by stubs.

In Top to down approach, testing takes place from top to down following the control flow of the software system.

Takes help of stubs for testing.



Top Down Approach

Advantages:

- Fault Localization is easier.
- Possibility to obtain an early prototype.
- Critical Modules are tested on priority; major design flaws could be found and fixed first.

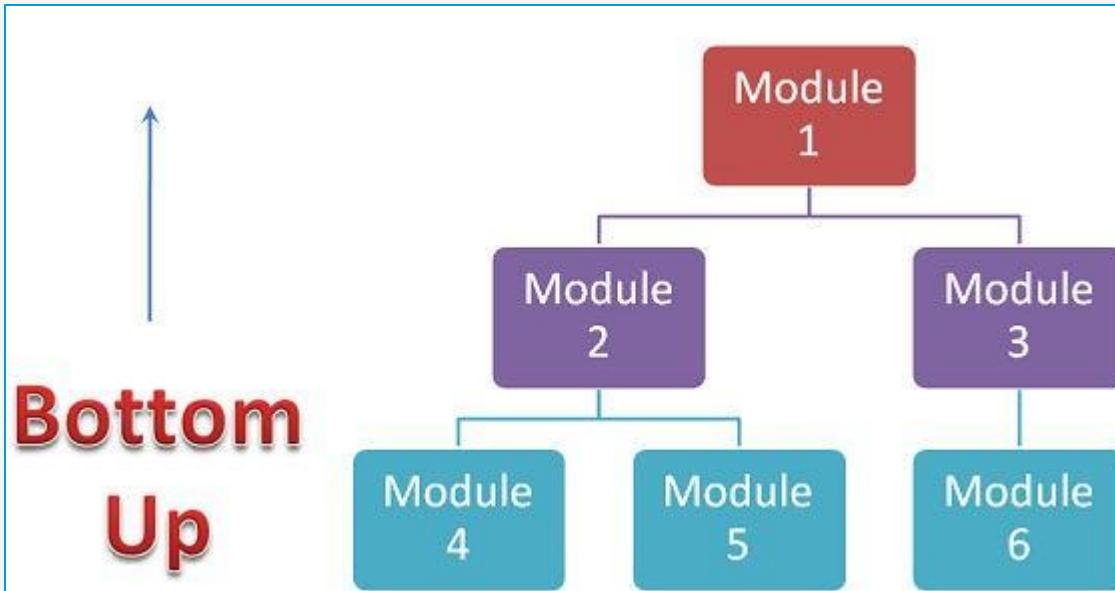
Disadvantages:

- Needs many Stubs.
- Modules at lower level are tested inadequately.

Bottom Up Approach

Testing takes place from the bottom of the control flow upwards.
Components or systems are substituted by drivers.

In the bottom up strategy, each module at lower levels is tested with higher modules until all modules are tested. It takes help of Drivers for testing



Bottom Up Approach

Advantages:

- Fault localization is easier.
- No time is wasted waiting for all modules to be developed unlike Big-bang approach

Disadvantages:

- Critical modules (at the top level of software architecture) which control the flow of application are tested last and may be prone to defects.
- Early prototype is not possible

Stub and Driver Approach

- **Stubs and Drivers** are the dummy programs in Integration testing used to facilitate the software testing activity.
 - These programs act as substitutes for the missing models in the testing.
 - They do not implement the entire programming logic of the software module but they simulate data communication with the calling module while testing.
-
- **Stub :** Is called by the Module under Test.
 - **Driver :** Calls the Module to be tested.

Continuous Integration Approach

- **Continuous Integration** is a software development method where team members integrate their work at least once a day.
- In this method, every integration is checked by an automated build to detect errors.
- This concept was first introduced over two decades ago to avoid “integration hell,” which happens when integration is put off till the end of a project.
- In Continuous Integration after a code commit, the software is built and tested immediately.
- In a large project with many developers, commits are made many times during a day. With each commit code is built and tested.
- If the test is passed, build is tested for deployment. If the deployment is a success, the code is pushed to Production.
- This commit, build, test, and deploy is a continuous process, and hence the name continuous integration/deployment.

Entry and Exit Criteria

Entry Criteria:

- Unit Tested Components/Modules
- All High prioritized bugs fixed and closed
- All Modules to be code completed and integrated successfully.
- Integration test Plan, test case, scenarios to be signed off and documented.
- Required Test Environment to be set up for Integration testing

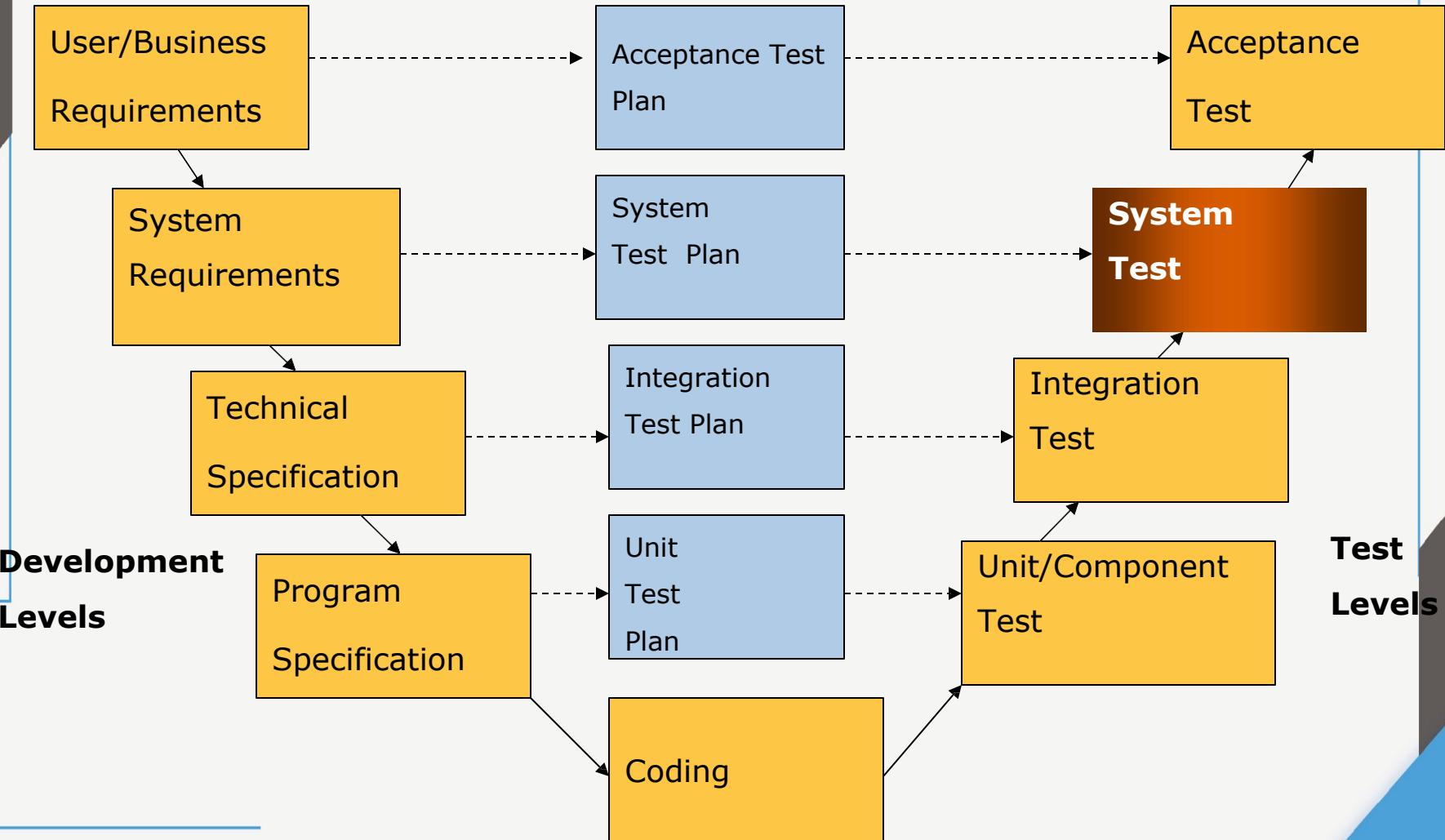
Exit Criteria:

- Successful Testing of Integrated Application.
- Executed Test Cases are documented
- All High prioritized bugs fixed and closed
- Technical documents to be submitted followed by release Notes.

Limitations

- Any condition not specified in integration tests, apart from the confirmation of the execution of the design items is usually not tested.

System Testing



System Testing

- **System Testing - process of testing an integrated system to verify that it meets specified requirements**

- In system testing the behavior of whole system/product is tested as defined by the scope of the development project or product.

- It may include tests based on risks and/or requirement specifications, business process, use cases, or other high level descriptions of system behavior, interactions with the operating systems, and system resources.

- System testing is most often the final test to verify that the system to be delivered meets the specification and its purpose.

- System testing is carried out by **specialists testers or independent testers**.

- System testing should investigate both functional and non-functional requirements of the testing.

- There are two types of System Testing which are:

- **Functional System Testing**
- **Non-Functional System Testing**

Functional System Testing

Functional System Testing : A requirement that specifies a function that a system or system component must perform

A Requirement may exist as a text document and/or a model

There is two types of Test Approach

- Requirement Based Functional Testing
- Process Based Testing

Functional System Testing Functionality As below:

Accuracy	Provision of right or agreed results or effects
Interoperability	Ability to interact with specified systems
Compliance	Adhere to applicable standards, conventions, regulations or laws
Auditability	Ability to provide adequate and accurate audit data
Suitability	Presence and appropriateness of functions for specified tasks

Requirement Based Testing

- Testing against requirements and specifications
- Test procedures and cases derived from:
 - detailed user requirements
 - system requirements functional specification
 - User documentation/instructions
 - high level System design
- Starts by using the most appropriate black-box testing techniques
- May support this with white-box techniques (e.g. menu structures, web page navigation)
- Risk based approach

Business Process Based Testing

Test procedures and cases derived from:

- Expected user profiles
- Business scenarios
- Use cases

Testing should reflect the business environment and processes in which the system will operate.

Therefore, test cases should be based on real business processes.

Functional and Non-Functional Testing

Functional Testing

Functional Testing: Testing based on an analysis of the specification of the functionality of a component or system.

'Specification' – E.g. Requirements specification, Use Cases, Functional specification or maybe undocumented.

'Function' – what the system does

Functional test – based on the Functions and features – may be applied at all Test levels (e.g. Component Test, System Test etc.)

Considers the external (not internal) behaviour of the software. Black-Box testing. What it does rather than how it does it. More on this later!

Functional testing verifies that each **function** of the software application operates in conformance with the requirement specification.

Functional Testing

- This testing mainly **involves black box testing** and it is not concerned about the source code of the application.
- Each & every functionality of the system is tested by providing appropriate input, verifying the output and comparing the actual results with the expected results.
- This testing involves checking of User Interface, APIs, Database, security, client/ server applications and functionality of the Application under Test. The testing can be done either manually or using automation

Functional Testing Examples

- **Web Based Testing :**

- Are you able to login to a system after entering correct credentials?
- Does your payment gateway prompt an error message when you enter incorrect card number?
- Does your “add a customer” screen adds a customer to your records successfully?

- **Desktop Based Testing :**

- Verifies Installation testing, Check for broken lines,
- Testing with different client accounts, Theme change and Print,
- Check for broken links. Warning messages. Resolution change effect on the **application**

- **Mobile Based Testing :**

- To validate whether the application works as per as requirement whenever the application starts/stops
- To validate whether the application goes into minimized mode whenever there is an incoming phone call. In order to validate the same we need to use a second phone, to call the device

- **Game Based Testing :**

- Takes more time to execute as testers look for game play issues, graphics issues, audio-visual issues, etc.
- Validates whether installation goes smoothly, the app works in minimized mode, the app allows social networking options, supports payment gateways, and many more.

Non-Functional Testing

Non-Functional Testing: Testing the attributes of a component or system that do not relate to functionality, e.g. reliability, efficiency, usability, interoperability, maintainability and portability

- May be performed at all Test levels (not just Non Functional Systems Testing)
- Measuring the **characteristics** of the system/software that can be quantified on a varying scale- e.g. performance test scaling
- Non-functional testing includes, but is not limited to, performance testing, load testing, stress testing, usability testing, maintainability testing, reliability testing and portability testing.

Non-Functional Testing

- It is the testing of “**how**” the system works. Non-functional testing may be performed at all test levels.

- The term non-functional testing describes the tests required to measure characteristics of systems and software that can be quantified on a varying scale, such as response times for performance testing.

- To address this issue, **performance testing is carried out to check & fine tune system response times**. The goal of performance testing is to reduce response time to an acceptable level

- Hence **load testing is carried out to check systems performance at different loads i.e. number of users accessing the system**

Non - Functional Testing Examples

- **Web Based Testing :**
 - Identify the software processes that directly influence the overall performance of the system.
 - **In website number of user/customer will increase , how the website will handled to every customer/user.**
- **Desktop Based Testing :**
 - Numerous other such GUI test cases, the desktop application tester must view
 - Guarantee that error messages are instructive and helpful for the client
 - Memory, and different other issues
- **Mobile Based Testing :**
 - In mobile , automatically will switch off without any reason.
 - To stop the application which is not in our hand.
- **Game Based Testing :**
 - Confirms workability and stability of the software.
 - Validate whether the user interface of the app is as per the screen size of the device and ensure high quality

Functional vs Non-Functional

Functional Testing	Non-Functional Testing
Functional testing is performed using the functional specification provided by the client and verifies the system against the functional requirements.	Non-functional testing checks the performance, reliability, scalability and other non-functional aspects of the software system.
Functional testing is executed first	Non functional testing should be performed after functional testing
Manual testing or automation tools can be used for functional testing	Using tools will be effective for this testing
Business requirements are the inputs to functional testing	Performance parameters like speed , scalability are inputs to non-functional testing.
Functional testing describes what the product does	Nonfunctional testing describes how good the product works
Easy to do manual testing	Tough to do manual testing
Types of Functional testing are <ul style="list-style-type: none"> • Unit Testing • Smoke Testing • Sanity Testing • Integration Testing • White box testing • Black Box testing • User Acceptance testing • Regression Testing 	Types of Nonfunctional testing are <ul style="list-style-type: none"> • Performance Testing • Load Testing • Volume Testing • Stress Testing • Security Testing • Installation Testing • Penetration Testing • Compatibility Testing • Migration Testing

Functional Testing

- **Black Box Testing**
- **White Box Testing**
- **Experience Based Testing**
- **Smoke Testing**
- **Sanity Testing**
- **End to End Testing**

Black box Testing & Technique

Introduction

Black-box testing: Testing, either functional or non-functional, without reference to the internal structure of the component or system.

Specification-based testing technique is also known as ‘black-box’ or input/output driven testing techniques because they view the software as a black-box with inputs and outputs.

The testers **have no knowledge of how the system or component is structured inside the box**. In black-box testing the tester is concentrating on what the software does, not how it does it.

Specification-based techniques are appropriate at all levels of testing (component testing through to acceptance testing) where a specification exists.

- For example, when performing system or acceptance testing, the requirements specification or functional specification may form the basis of the tests.

Introduction(Cont...)

- The technique of testing without having **any knowledge of the interior workings of the application** is Black Box testing.
- What a system does, rather than HOW it does it
- Typically used at System Test phase, although can be useful throughout the test lifecycle
- The tester is oblivious to the **system architecture and does not have access to the source code**.
- Typically, when performing a black box test, **a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon**.



If Output = Expected result then pass

Introduction(Cont...)

Advantages

- Well suited and efficient for large code segments.
- Code Access not required.
- Clearly separates user's perspective from the developer's perspective through visibly defined roles.
- Large numbers of moderately skilled testers can test the application with no knowledge of implementation, programming language or operating systems.

Disadvantage

- Limited Coverage since only a selected number of test scenarios are actually performed.
- Inefficient testing, due to the fact that the tester only has limited knowledge about an application.
- Blind Coverage, since the tester cannot target specific code segments or error prone areas.
- The test cases are difficult to design.

Black Box Testing Examples

Web Based Testing :

- Takes more time to execute as testers look for game play issues, graphics issues, audio-visual issues, etc.
- Validates whether installation goes smoothly, the app works in minimized mode, the app allows social networking options, supports payment gateways, and many more.
- Login by the user is must for accessing the sensitive information.

Desktop Based Testing :

- Resolution change effect on the application
- Installation Testing (Upgrade/Downgrade)

Mobile Based Testing :

- In mobile , automatically will switch off without any reason.
- To stop the application which is not in our hand.

Game Based Testing :

- the game tester must know how to play the game, utilization of the gamepad, know the game flow and the rules.

Techniques of Black Box Testing

- There are four specification-based or black-box technique:

- Equivalence partitioning
- Boundary value analysis
- Decision tables
- State transition testing
- Use-case Testing
- Other Black Box Testing
 - Syntax or Pattern Testing

Equivalence Partitioning(E.P.)

- Aim is to treat groups of inputs as equivalent and to select one representative input to test them all

- EP can be used for all Levels of Testing

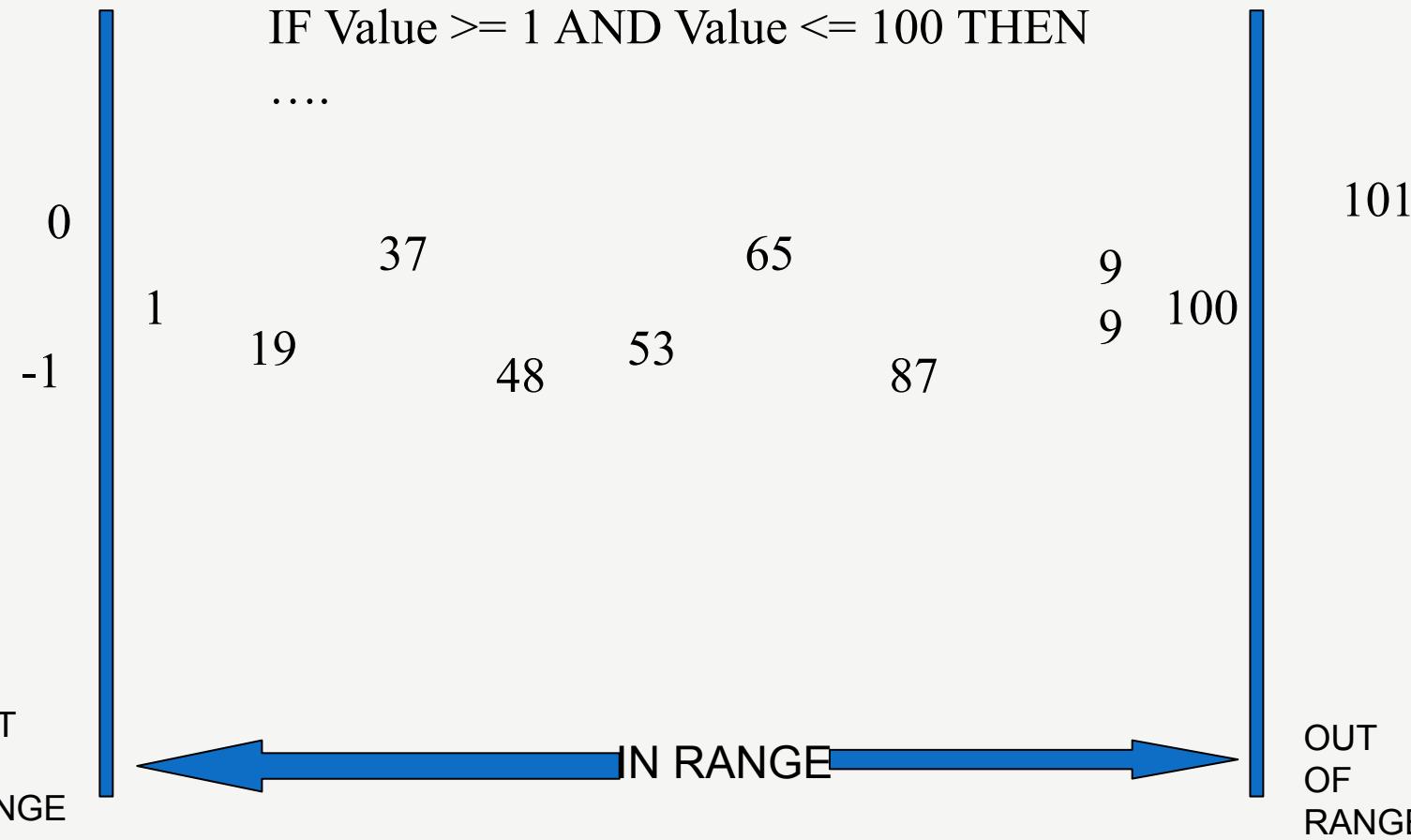
- Equivalence partitioning is the process of defining the optimum number of tests by:

- Reviewing documents such as the Functional Design Specification and Detailed Design Specification, and identifying each input condition within a function,
- Selecting input data that is representative of all other data that would likely invoke the same process for that particular condition.

- If we want to test the following IF statement: “If value is between 1 and 100 (inclusive) (e.g value ≥ 1 and value ≤ 100) Then...”

- We could put a range of numbers as shown in the below figure.

Equivalence Partitioning(E.P.)



Equivalence Partitioning(E.P.)

The numbers fall into a partition where each would have the same, or equivalent, result i.e. an Equivalence Partition (EP) or Equivalence Class

EP says that by testing just one value we have tested the partition (typically a mid-point value is used). It assumes that:

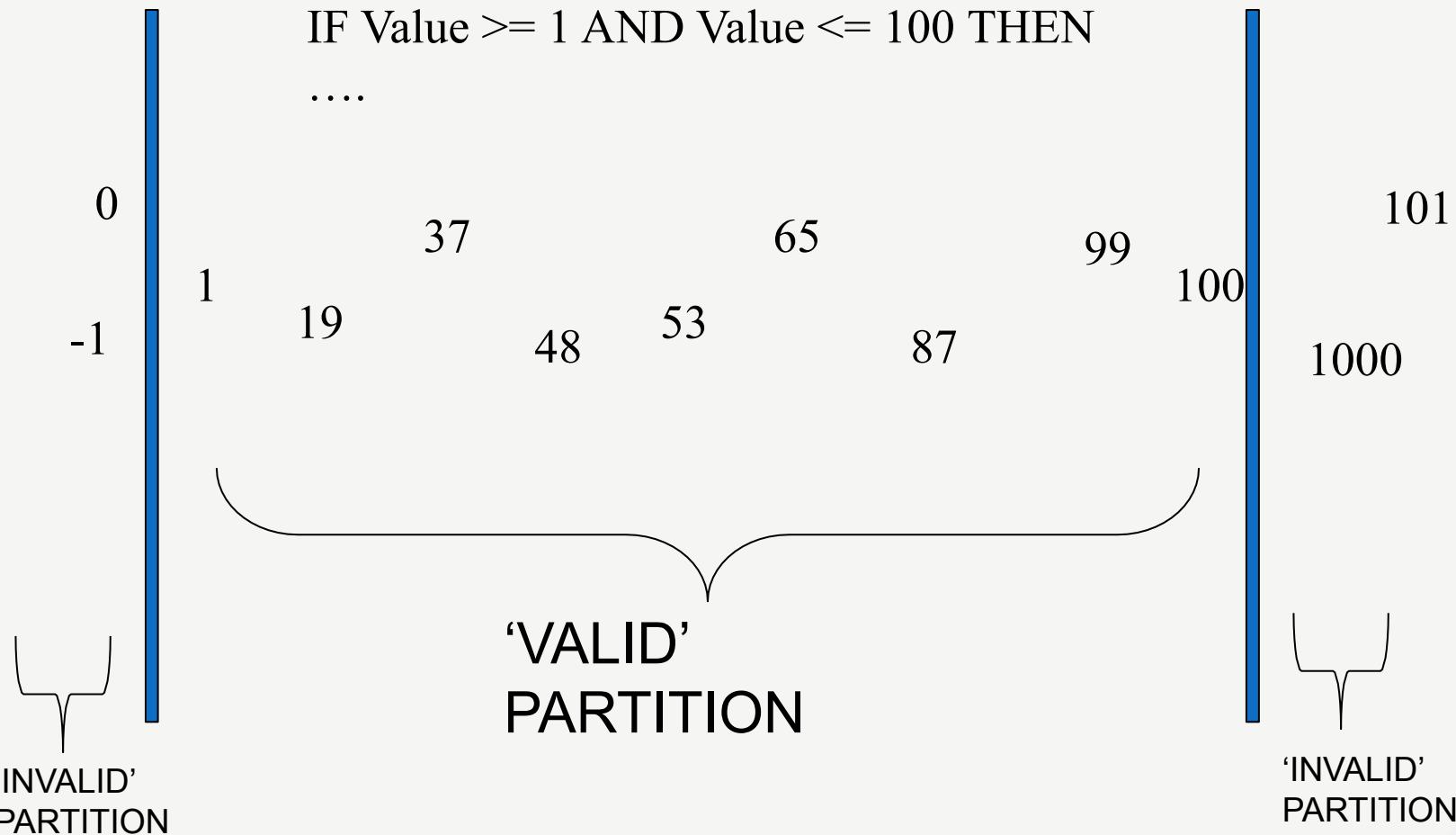
- If one value finds a bug, the others probably will too
- If one doesn't find a bug, the others probably won't either

In EP we must identify Valid Equivalence partitions and Invalid Equivalence partitions where applicable (typically in range tests)

The Valid partition is bounded by the values 1 and 100

Plus there are 2 Invalid partitions

Equivalence Partitioning(E.P.)



Boundary Value Analysis(B.V.A.)

- Boundary value analysis is a methodology for designing test cases that concentrates software testing effort on cases near **the limits of valid ranges**

- Boundary value analysis is a method which **refines** equivalence partitioning.

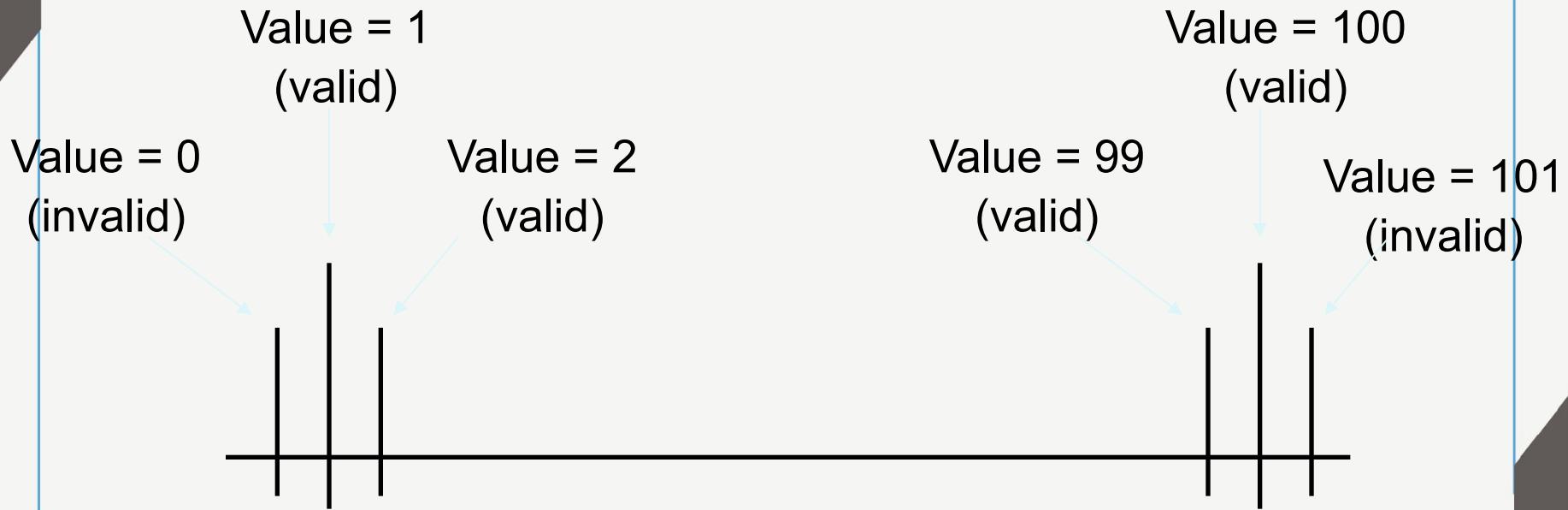
- Boundary value analysis generates test cases that highlight errors better than equivalence partitioning.

- The trick is to concentrate software testing efforts at the extreme ends of the equivalence classes.

- At those points when input values change from valid to invalid errors are most likely to occur.

- Boundary Value Analysis (BVA) uses the same analysis of partitions as EP and is usually used in conjunction with EP in test case design

Boundary Value Analysis(B.V.A.)



Decision Table

- The techniques of equivalence partitioning and boundary value analysis are often applied to specific situations or inputs.
- However, if different combinations of inputs result in different actions being taken, this can be more difficult to show using equivalence partitioning and boundary value analysis, which tend to be more focused on the user interface.
- The other two specification-based software testing techniques, decision tables and state transition testing are more focused on business logic or business rules.**
- A decision table is a good way to deal with combinations of things (e.g. inputs).**
- This technique is sometimes also referred to as a '**cause-effect**' table. The reason for this is that there is an associated logic diagramming technique called '**cause-effect graphing**' which was sometimes used to help derive the decision table (Myers describes this as a combinatorial logic network [Myers, 1979]). However, most people find it more useful just to use the table described in [Copeland, 2003].

Decision Table

- Table based technique where

- Inputs to the system are recorded
- Outputs to the system are defined

Inputs are usually defined in terms of actions which are Boolean (true or false)

- Outputs are recorded against each unique combination of inputs

Using the **Decision Table the relationships between the inputs and the possible outputs are mapped together**

- As with State Transition testing, an excellent tool to capture certain types of system requirements and to document internal system design.

- As such can be used for a number of test levels

- Especially useful for complex business rules

Decision Table

- Each column of the table corresponds to a business rule that defines a unique combination of conditions that result in the execution of the actions associated with that rule
- The strength of Decision Table testing is that it creates combinations of conditions that might not otherwise have been exercised during testing

	Test 1	Test 2	Test 3
Inputs / Actions	Input 1	T	T
	Input 2	T	T
	Input 3	T	DON'T CARE
	Input 4	T	F
Output / Response	Response 1	Y	N
	Response 2	Y	Y
	Response 3	N	Y

Decision Table

What will be the outcome of the following Scenarios?

Joe is a 22 year old non smoker who goes to the gym 4 times / week and has no history of heart attacks in his family

Kevin is 62 year old non smoker who swims twice a week and plays tennis. He has no history of heart attacks in his family

	Test 1	Test 2	Test 3
> 55 yrs old	F	T	T
Smoker	F	T	F
Exercises 3 times a week +	T	F	T
History of Heart Attacks	F	T	F
Insure	Y	N	Y
Offer 10% Discount	N	N	Y
Offer 30% Discount	Y	N	N

State Transaction Testing

State Transition Testing uses the following terms:

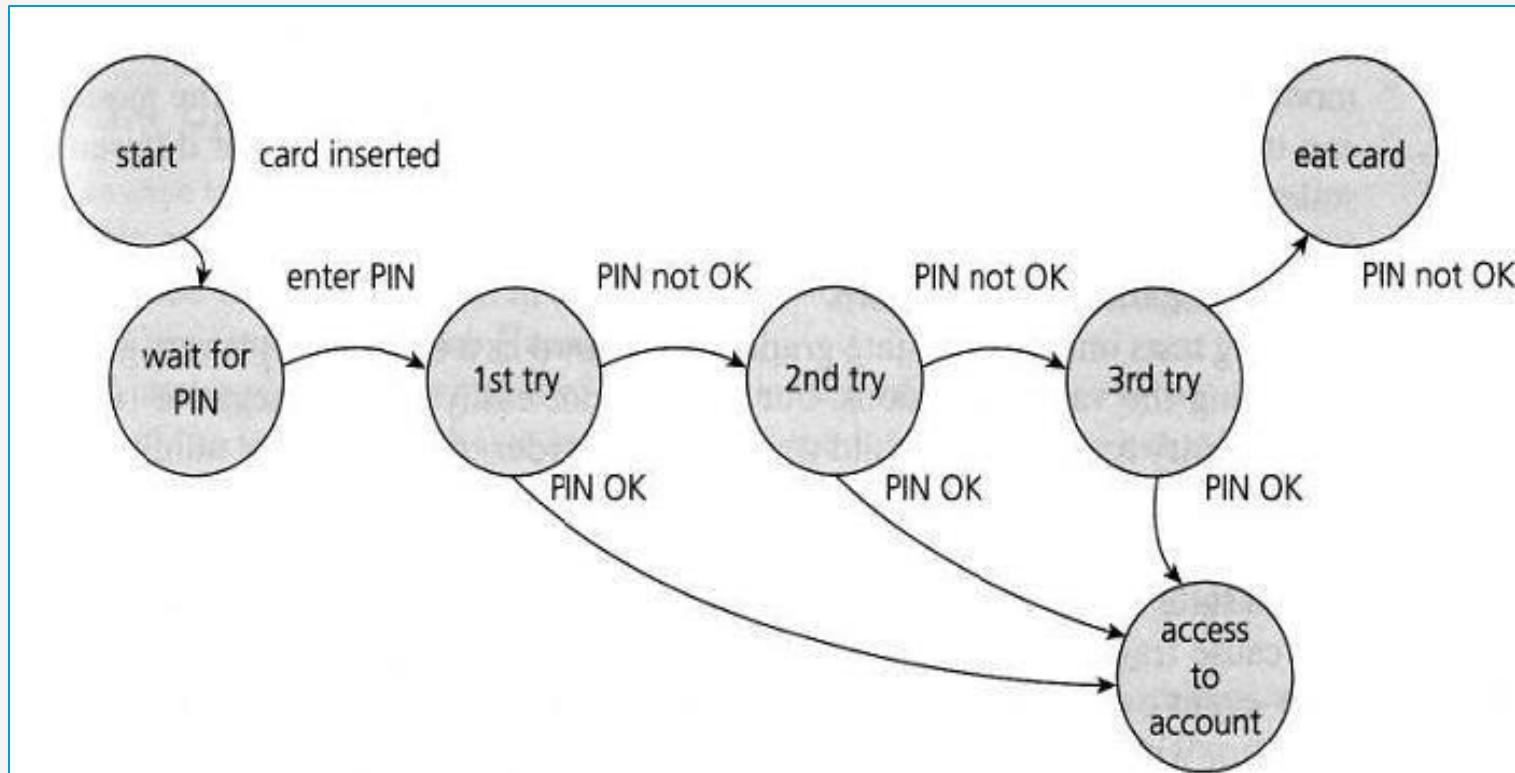
- **State Diagram:** A diagram that depicts the states that a component or system can assume, and shows the events or circumstances that cause and/or result from a change from one state to another. [IEEE 610]
- **State Table:** A grid showing the resulting transitions for each state combined with each possible event, showing both valid and invalid transitions.
- **State Transition:** A transition between two states of a component or system.
- **State Transition Testing:** A black box test design technique in which test cases are designed to execute valid and invalid state transitions. Also known as N-switch testing.
- An excellent tool to capture certain types of system requirements and to document internal system design. As such can be used for a number of test levels
- Often used in testing:
 - Screen dialogues

State Transaction Testing

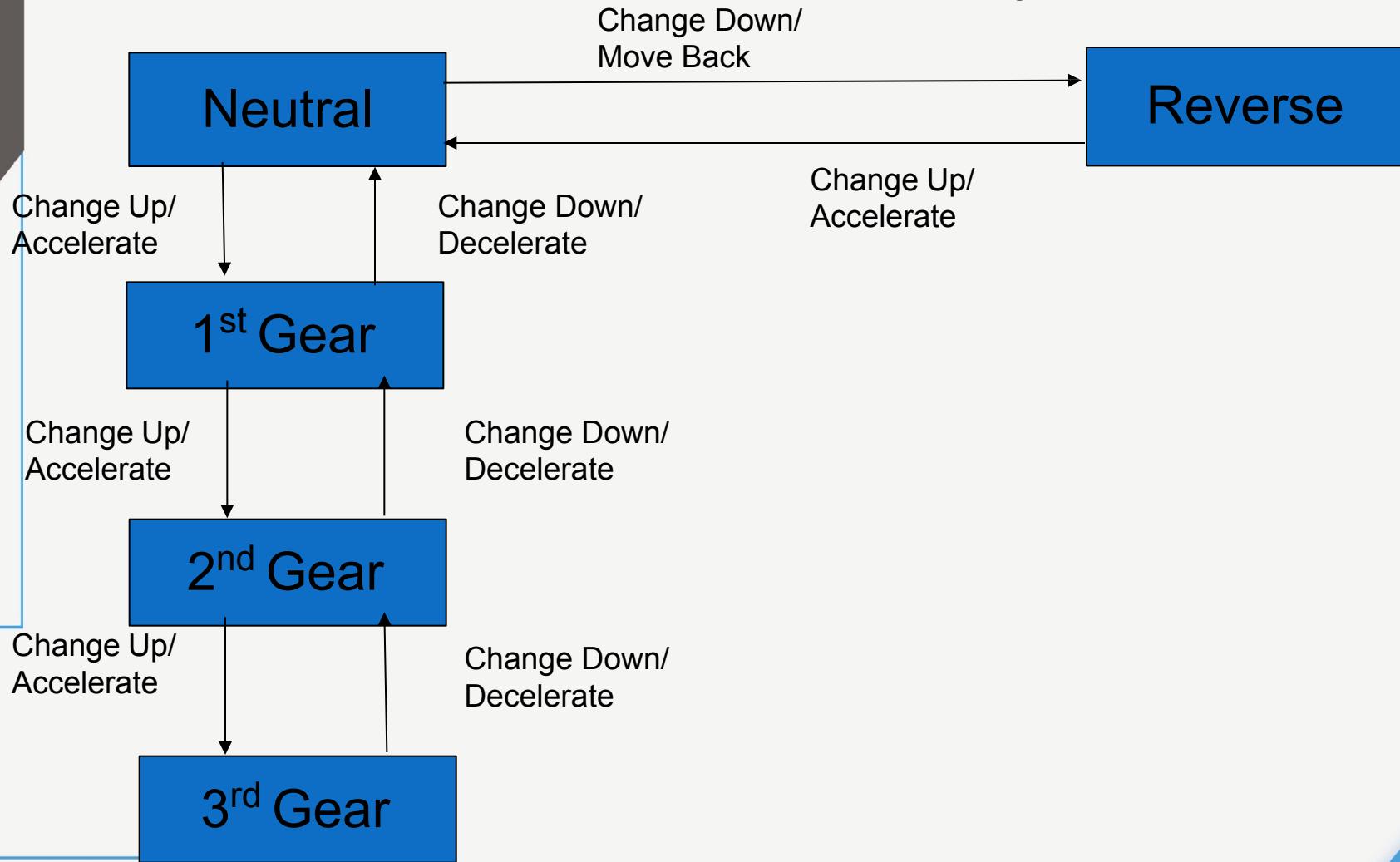
State transition testing uses the same principles as the State Transition Diagramming design technique.

- State transition testing is used where some aspect of the system can be described in what is called a '**finite state machine**'. This simply means that the system can be in a (finite) number of different states, and the transitions from one state to another are determined by the rules of the '**machine**'. This is the model on which the system and the tests are based.
- Any system where you get a **different output for the same input**, depending on what has happened before, **is a finite state system**.
- **A finite state system is often shown as a state diagram.(next slide given example)**
- One of the advantages of the state transition technique is that the model can be as detailed or as abstract as you need it to be.
 - Where a part of the system is **more important** (that is, requires more testing) a greater depth of detail can be modeled.
 - Where the system is **less important** (requires less testing), the model can use a single state to signify what would otherwise be a series of different states.

State Transaction Example



State Transaction Examples



Switch Coverage of State Transaction

- Switch Coverage is a method of determining the number tests based on the number of “hops” between transitions. Some times known as Chow
- These hops can be used to determine the VALID tests

0-Switch Coverage (1 hop)

R□N
N□R
N□1
1□N
1□2
2□1
2□3
3□2

1-Switch Coverage (2 hops)

R□N□1
R□N□R
N□R□N
N□1□2
N□1□N
1□N□R
Etc.

WhiteBox Testing & Technique

Introduction

White Box Testing: *Testing based on an analysis of the internal structure of the component or system.*

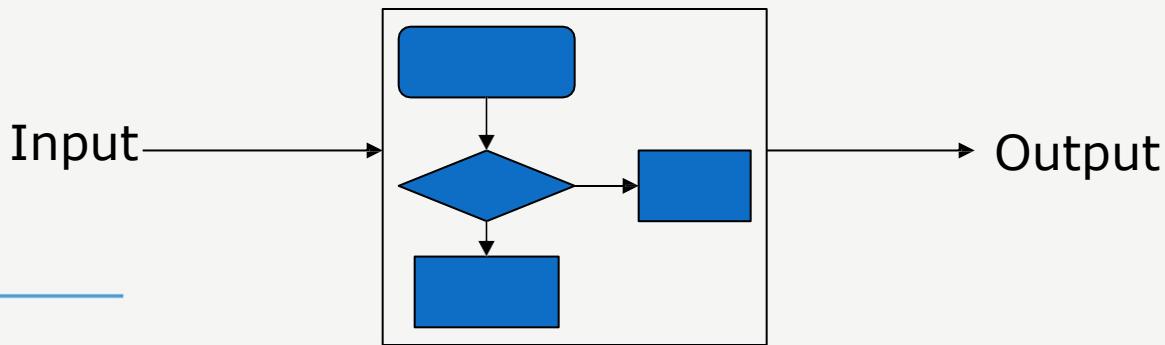
Structure-based testing technique is also known as '**white-box**' or '**glass-box**' testing technique because here **the testers require knowledge of how the software is implemented, how it works.**

In white-box testing the tester is concentrating on how the software does it.

- For example, a structural technique may be concerned with exercising loops in the software.
- Different test cases may be derived to exercise the loop once, twice, and many times. This may be done regardless of the functionality of the software.
- Structure-based techniques are also used in system and acceptance testing, but the structures are different.
- For example, the coverage of menu options or major business transactions could be the structural element in system or acceptance testing.

Introduction(Cont...)

- Testing based upon the structure of the code
- Typically undertaken at Component and Component Integration Test phases by development teams
- White box testing is the detailed investigation of internal logic and structure of the code.
- White box testing is also called **glass testing or open box testing**. In order to perform white box testing on an application, the tester needs to possess knowledge of the internal working of the code.
- The tester needs to have a look inside the source code and find out which unit/chunk of the code is behaving inappropriately.



White Box Testing Examples

- **Web Based Testing :**
 - Analyze the logic by reading the code
 - Code optimization Suggest if any optimization can be done in the code which is better than the existing one
 - Deploy the code using different web servers which could be a case study even if the product is not supporting other web servers
- **Desktop Based Testing :**
 - When we debug the code when we writing
- **Mobile Based Testing :**
 - The Android SDK and related plugin for Eclipse
 - Android devices enabled for development and debugging, with appropriate USB drivers as necessary
- **Game Based Testing :**
 - When we connect with remote device , so which device we connect will check in code
 - When some debug the code and play at that time game.

Test/Code Coverage

Test coverage measures the amount of testing performed by a set of test. Wherever we can count things and can tell whether or not each of those things has been tested by some test, then we can measure coverage and is known as test coverage.

The basic coverage measure is where the ‘coverage item’ is whatever we have been able to count and see whether a test has exercised or used this item

$$\text{Coverage} = \frac{\text{Number of coverage items exercised}}{\text{Total number of coverage items}} \times 100\%$$

There is danger in using a coverage measure. But, 100% coverage does *not* mean 100% tested. Coverage techniques measure only one dimension of a multi-dimensional concept. Two different test cases may achieve exactly the same coverage but the input data of one may find an error that the input data of the other doesn't.

Benefit & Drawback of Test/Code Coverage

BENEFIT:

- It creates additional test cases to increase coverage
- It helps in finding areas of a program not exercised by a set of test cases
- It helps in determining a quantitative measure of code coverage, which indirectly measures the quality of the application or product.

DRAWBACK :

- One drawback of code coverage measurement is that it measures coverage of what has been written, i.e. the code itself; it cannot say anything about the software that has not been written.
- If a specified function has not been implemented or a function was omitted from the specification, then structure-based techniques cannot say anything about them it only looks at a structure which is already there.

Types of Coverage

The different types of coverage
are:

- Statement coverage
- Decision coverage
- Condition coverage

Statement/Segment Coverage

- The statement coverage is also known as line **coverage or segment coverage**.
- The statement coverage **covers only the true conditions**.
- Through statement coverage we can identify the statements executed and where the code is not executed because of blockage.
- In this process each and every line of code needs to be checked and executed.
- Aim is to display that all executable statements have been run at least once

The statement coverage can be calculated as shown below:

$$\text{Statement coverage} = \frac{\text{Number of statements exercised}}{\text{Total number of statements}} \times 100\%$$

Statement/Segment Coverage

ADVANTAGE:

- It verifies what the written code is expected to do and not to do
- It measures the quality of code written
- It checks the flow of different paths in the program and it also ensure that whether those path are tested or not.

DISADVANTAGE:

- It cannot test the false conditions.
- It does not report that whether the loop reaches its termination condition.
- It does not understand the logical operators.

Decision/Branch Coverage

Decision coverage **also known as branch coverage** or all-edges coverage.

It **covers both the true and false conditions** unlikely the statement coverage.

A branch is the outcome of a decision, so branch coverage simply measures which decision outcomes have been tested.

Aim is to demonstrate that all Decisions have been run at least once

Decision and Branch Test coverage for a piece of code is often the same, but not always

With an IF statement, the exit can either be TRUE or FALSE, depending on the value of the logical condition that comes after IF.

The decision coverage can be calculated as shown below:

$$\text{Decision coverage} = \frac{\text{Number of decision outcomes exercised}}{\text{Total number of decision outcomes}} \times 100\%$$

Decision/Branch Coverage

A decision is an IF statement, a loop control statement (e.g. DO-WHILE or REPEAT-UNTIL, JUMP, GO TO), or a CASE statement, where there are two or more outcomes from the statement.

ADVANTAGES:

- To validate that all the branches in the code are reached
- To ensure that no branches lead to any abnormality of the program's operation
- It eliminate problems that occur with statement coverage testing

DISADVANTAGES:

- This metric ignores branches within Boolean expressions which occur due to short-circuit operators.

NOTE:

- Branch Coverage Testing \geq Statement Coverage Testing

Condition Coverage

- This is closely related to decision coverage but has better sensitivity to the control flow.

- However, **full condition coverage does not guarantee full decision coverage.**

- Condition coverage reports the true or false outcome of each condition.

- Condition coverage measures the conditions independently of each other.

Statement Coverage Example

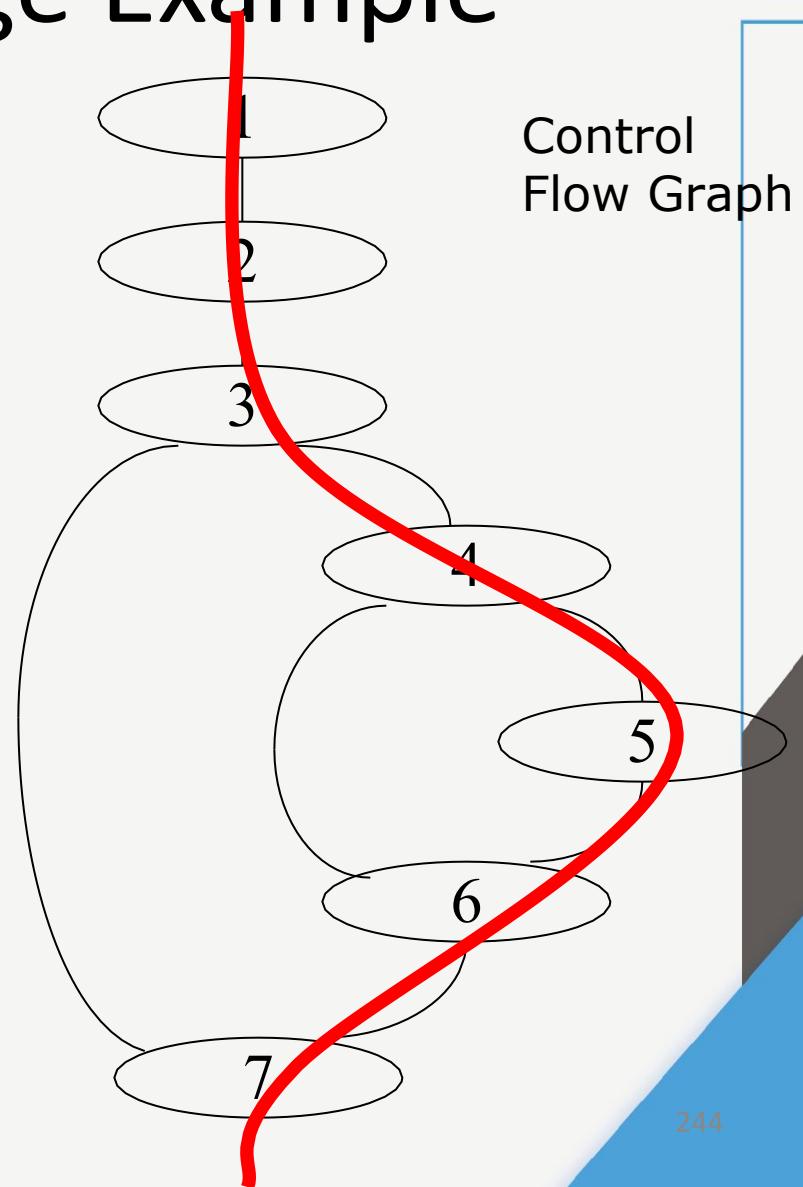
1. Read vehicle
2. Read colour
3. If vehicle = 'Car' Then
4. If colour = 'Red' Then
5. Print "Fast"
6. End If
7. End If

Answer :

- Statement Coverage is 1.

Reason:

- Because of we need to take single input to cover the single statement
- Consider Input is:
- Vehicle = car and color = red



Decision Coverage Example

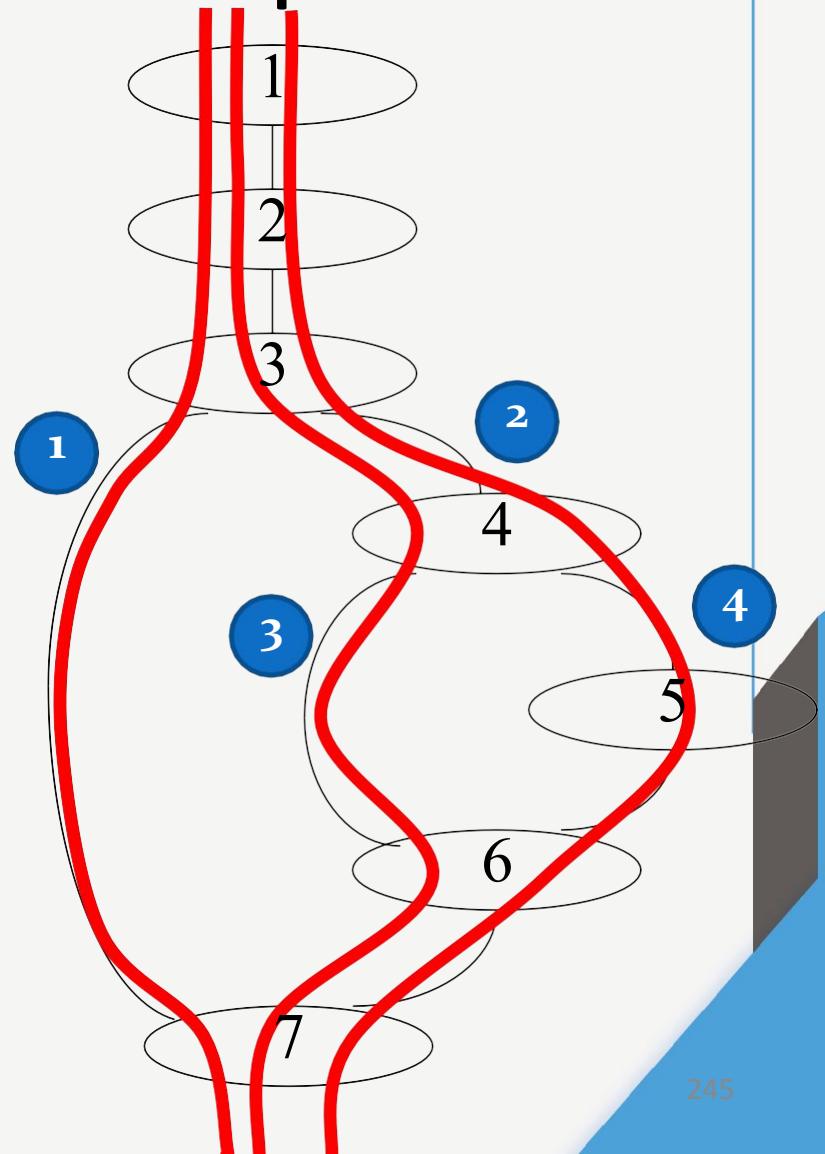
1. Read vehicle
2. Read colour
3. If vehicle = 'Car' Then
4. If colour = 'Red' Then
5. Print "Fast"
6. End If
7. End If

Answer :

- Decision/Branch Coverage is 3.

Reason:

- Because of we need to take three input to cover the four branches
- Consider Input is:
 - (Vehicle = car and color = red) cover branches 2 and 4
 - (Vehicle = scooter and color = black) cover branch 1
 - (Vehicle = car and color = black) cover branch 2 and 3.



Code Cover in Flow Chart

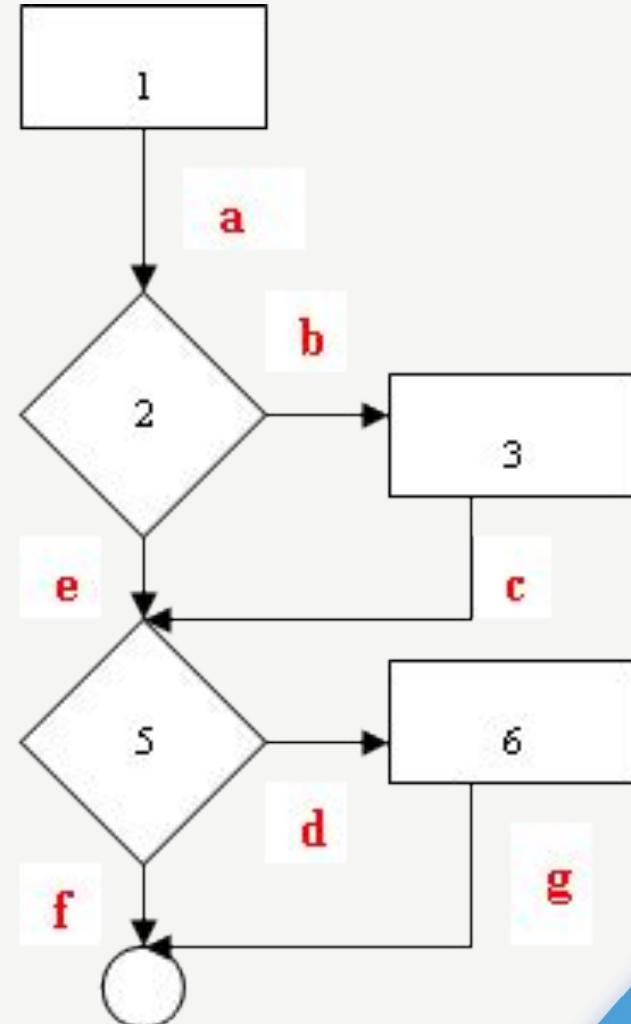
1. Read A
2. If A > 40 Then
 3. $A = A * 2$
4. End If
5. If A > 100 Then
 6. $A = A - 10$
7. End If

Answer :

- Decision/Branch Coverage is 2.

Reason:

- Because of we need to take two input to cover the four branches
- Consider Input is:
 - $A = 60$ cover branch no 2 and 4
 - $A = 10$ cover branch no 1 and 3



Statement Coverage Example

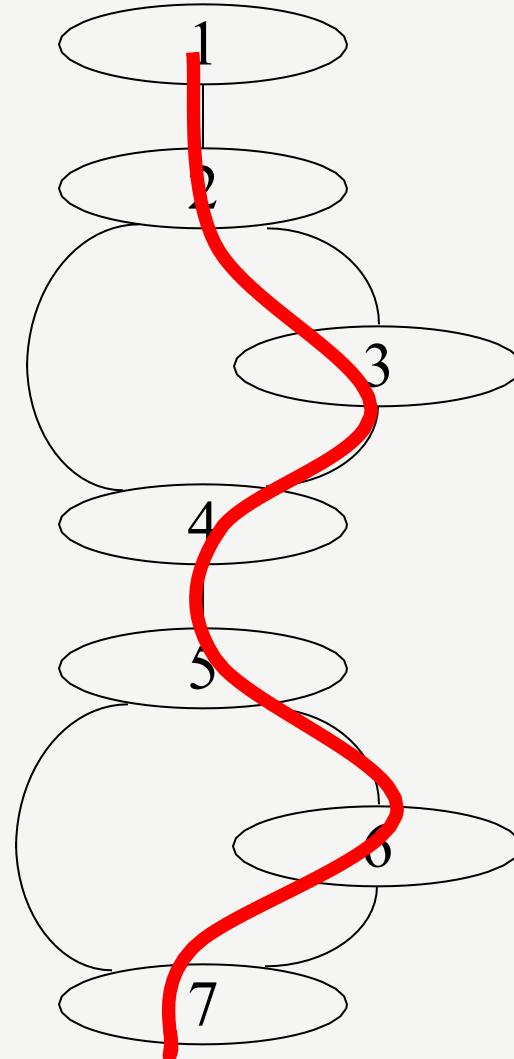
1. Read A
2. If A > 40 Then
3. A = A * 2
4. End If
5. If A > 100 Then
6. A = A – 10
7. End If

Answer :

- Statement Coverage is 1.

Reason:

- Because of we need to take single input to cover the two statement
- Consider Input is:
- A = 60



Decision Coverage Example

1. Read A
2. If $A > 40$ Then
3. $A = A * 2$
4. End If
5. If $A > 100$ Then
6. $A = A - 10$
7. End If

Answer :

- Decision/Branch Coverage is 2.

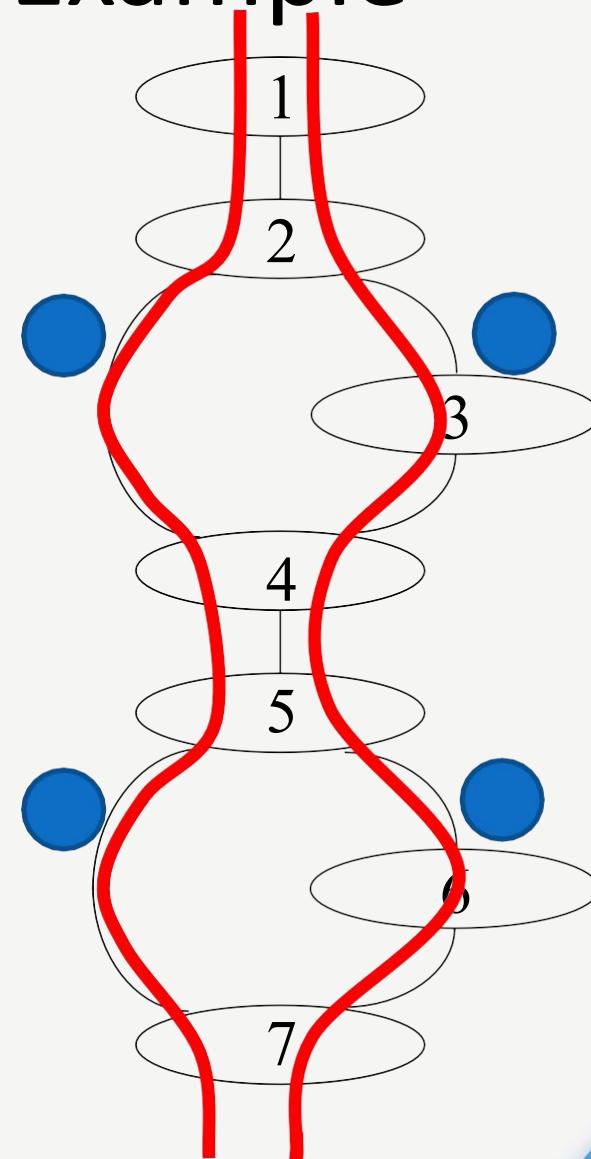
Reason:

- Because of we need to take two input to cover the four branches

- Consider Input is:

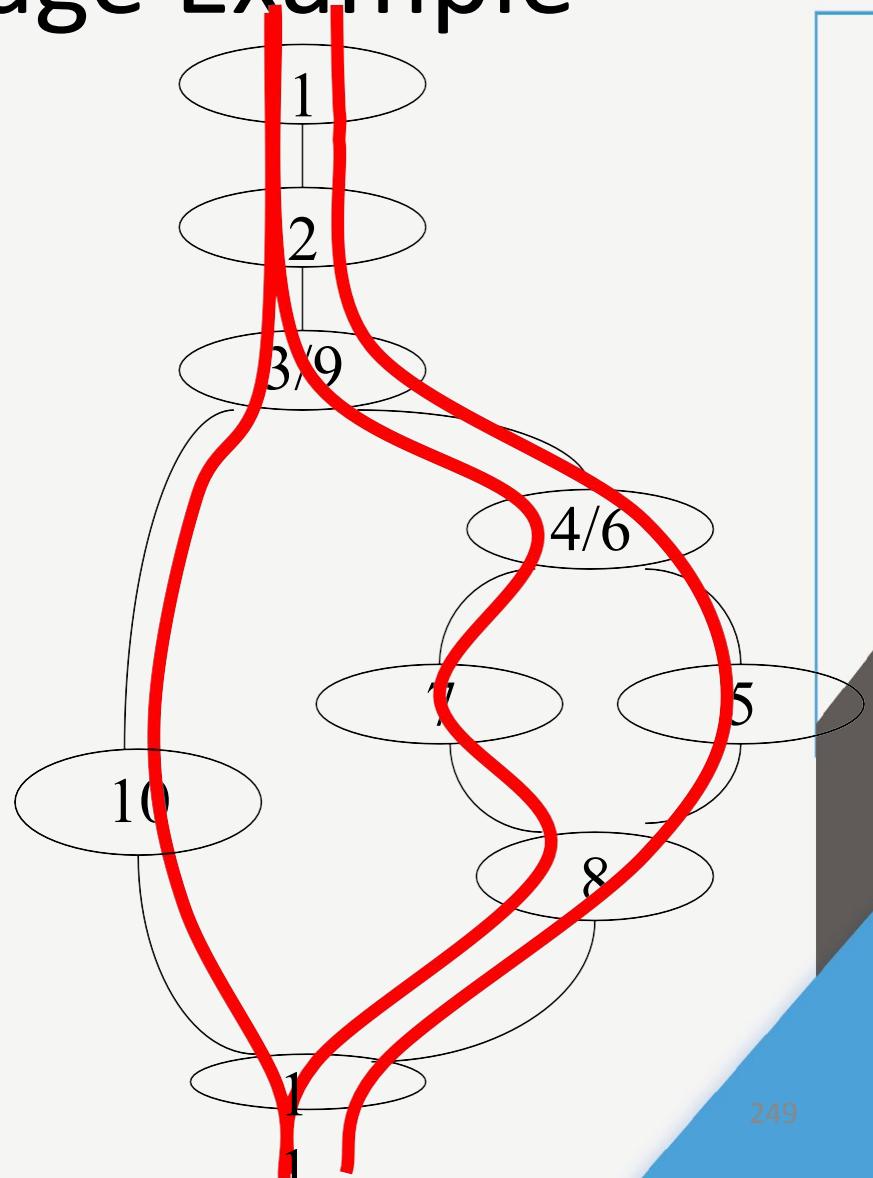
- $A = 60$ cover branch no 2 and 4

- $A = 10$ cover branch no 1 and 3



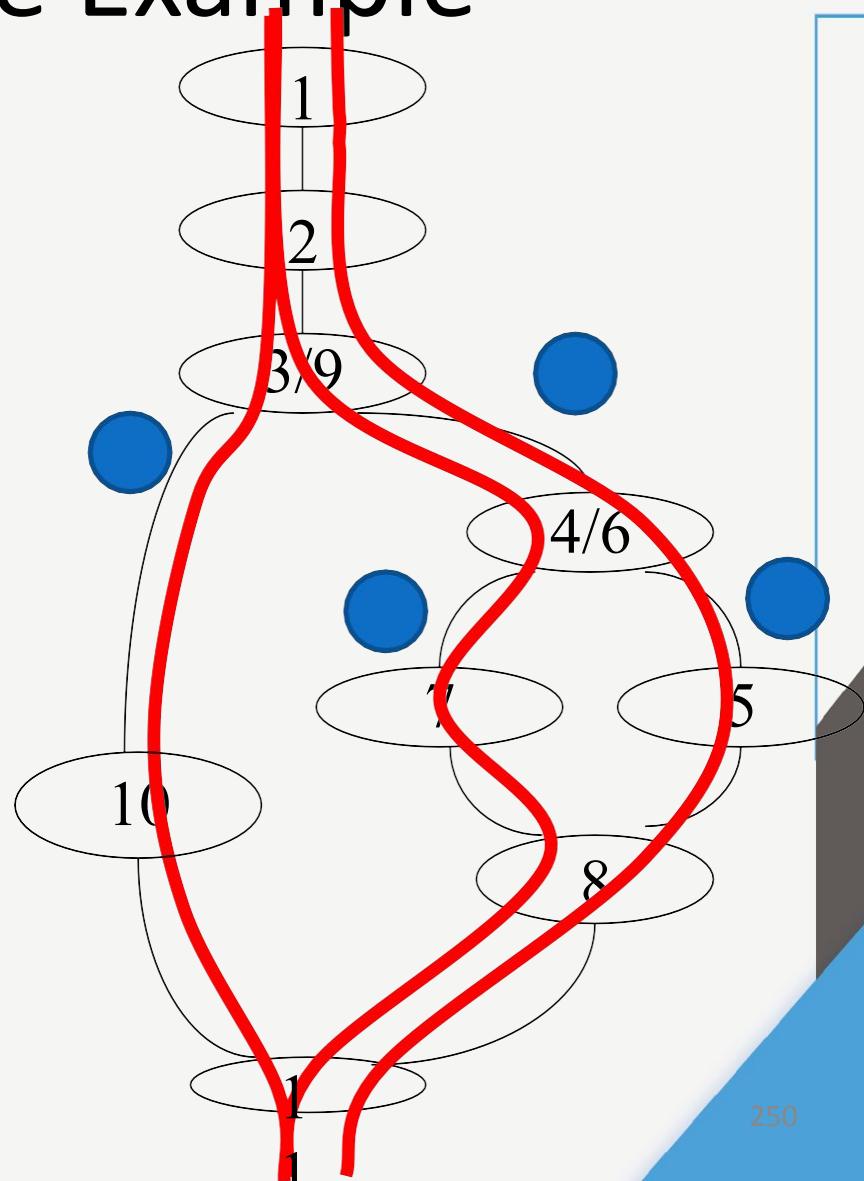
Statement Coverage Example

1. Read bread
2. Read filling
3. If bread = 'Roll' Then
4. If filling = 'Tuna' Then
5. Price = 1.50
6. Else
7. Price = 1.00
8. End If
9. Else
10. Price = 0.75
11. End If



Decision Coverage Example

1. Read bread
2. Read filling
3. If bread = 'Roll' Then
4. If filling = 'Tuna' Then
5. Price = 1.50
6. Else
7. Price = 1.00
8. End If
9. Else
10. Price = 0.75
11. End If



Coverage Answer of Above Ex.

Statement Coverage to achieve 100%

- **Answer is : 3**
- **Reason : Consider 3 inputs for covering 3 statements**
 - Bread = Roll and filling = Tuna, which cover price = 1.50
 - Bread = Roll and filling = Other, which cover price = 1.00
 - Bread = Swiss and filling = Other, which cover price = 0.75

Decision/Branch Coverage to achieve 100%

- **Answer = 3**
- **Reason : Consider 3 inputs for covering 4 branches.**
 - Bread = Roll and filling = Tuna, which cover branch 2 and 4
 - Bread = Roll and filling = Other, which cover branch 2 and 3
 - Bread = Swiss and filling = Other, which cover branch 1.

No of Branches : 4 (T and F of each IF...ELSE)

No of Executable Statements : 7

Other White Box Techniques

Branch Condition testing

- Branch Condition Testing requires that the True and False of each Boolean operand is tested (**Boolean Operands** in this example: *If A > 30 and B >= 5*)

Branch Condition Combination testing

- Branch Condition Combination Coverage would require all **combinations** of Boolean operands to be evaluated

Modified Condition Decision testing

- Modified Condition Decision Coverage requires test cases to show that each Boolean operand can independently affect the outcome of the decision

Dataflow testing

- Data flow testing aims to execute sub-paths from points where each variable in a component is defined to points where it is referenced.

Linear Code Sequence And Jump (LCSAJ) testing

- LCSAJ testing requires a model of the source code which identifies control flow jumps (where control flow does not pass to a sequential statement).

Experience Based Testing

Introduction

- Experience based techniques are non structured and do not rely on specification documents. This makes them unable to be measured in terms of coverage

- In **experience-based techniques**, people's knowledge, skills and background are of prime importance to the test conditions and test cases.

- The experience of both technical and business people is required, as they bring different perspectives to the **test analysis and design process**.

- This may be the only type of technique used for **low-risk systems**, but this approach may be particularly useful under extreme time pressure – in fact this is **one of the factors leading to exploratory testing**.

Introduction(Cont...)

- Uses the knowledge of people and the experience of past projects to determine likely errors based on the knowledge

- Testers
- Users
- Stakeholders

- Good to identify tests which may not be explicitly described in the specification

- Most beneficial when applied after more formal measures

Grey Box Testing

Grey Box testing is a technique to test the application with limited knowledge of the internal workings of an application.

In software testing, **the term the more you know the better** carries a lot of weight when testing an application.

Mastering the domain of a system always gives the tester an edge over someone with limited domain knowledge.

Unlike black box testing, where the tester only tests the application's user interface, in grey box testing, the tester has access to design documents and the database.

Having this knowledge, **the tester is able to better prepare test data and test scenarios when making the test plan.**

Adhoc Testing(Error Guessing)

- Adhoc testing is an informal testing type with an **aim to break the system.**
- It does not follow any test design techniques to create test cases.
- In fact it does not create test cases altogether!**
- This testing is primarily performed **if the knowledge of testers in the system under test is very high.**
- Testers randomly test the application without any test cases or any business requirement document.
- Adhoc Testing does not follow any structured way of testing and it is randomly done on any part of application.
- Main aim of this testing is to find defects by random checking.**
- Adhoc testing can be achieved with the testing technique called Error Guessing.**
- Error guessing can be done by the people having enough experience on the system to “**guess**” the most likely source of errors.

Adhoc Testing(Error Guessing)

The Error guessing is a technique where the experienced and good testers are encouraged to think of situations in which the software may not be able to cope.

Some people seem to be naturally good at testing and others are good testers because they have a **lot of experience** either as a **tester or working with a particular system** and so are able to find out its weaknesses.

This is why an error guessing approach, used after **more formal techniques** have been applied to some extent, can be very effective.

It also saves a lot of time because of the assumptions and guessing made by the experienced testers to find out the defects which otherwise won't be able to find.

Using experience to postulate errors.

Use Error Guessing to Complement Test Design Techniques.

Types of Adhoc Testing

There are different types of Adhoc testing and they are listed as below:

1. Buddy Testing

- Two buddies mutually work on identifying defects in the same module. Mostly one buddy will be from development team and another person will be from testing team. Buddy testing helps the testers develop better test cases and development team can also make design changes early. This testing usually happens after unit testing completion.

2. Pair testing

- Two testers are assigned modules, share ideas and work on the same machines to find defects. One person can execute the tests and another person can take notes on the findings. Roles of the persons can be a tester and scribe during testing.

3. Monkey Testing

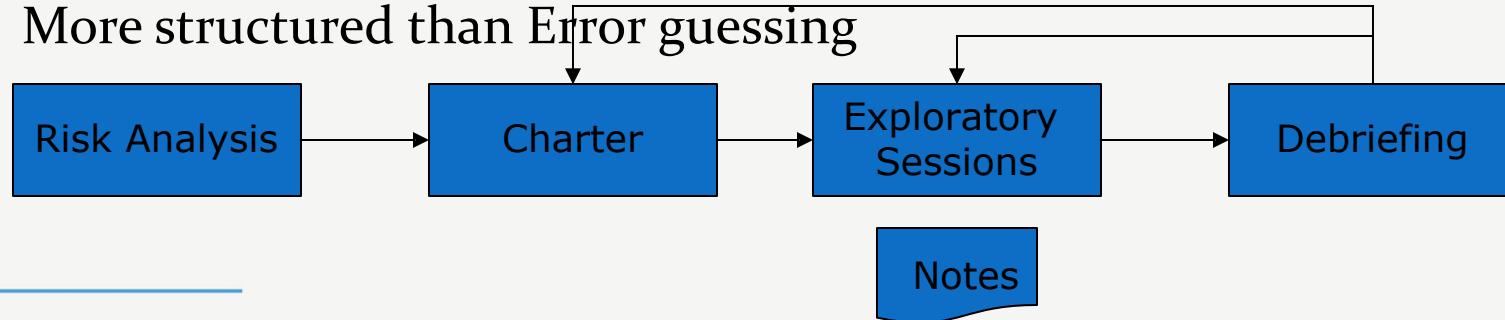
- Randomly test the product or application without test cases **with a goal to break the system**.

Exploratory Testing

Exploratory testing is a concurrent process where

- Test design, execution and logging happen simultaneously
- Testing is often not recorded
- Makes use of experience, heuristics and test patterns
- Testing is based on a test charter that may include
 - Scope of the testing (in and out)
 - The focus of exploratory testing is more on testing as a “thinking” activity.
 - A brief description of how tests will be performed
 - Expected problems
- Is carried out in time boxed intervals

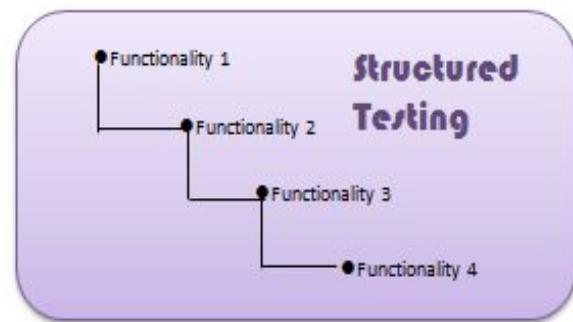
More structured than Error guessing



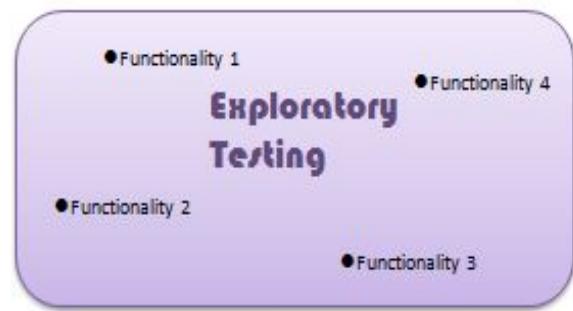
Exploratory Testing

Though the current trend in testing is to push for automation, exploratory testing is a new way of thinking. **Automation has its limits**

- Is not random testing but it is Adhoc testing with purpose of find bugs
- Is structured and rigorous
- Is cognitively (thinking) structured as compared to procedural structure of scripted testing. This structure comes from Charter, time boxing etc.
- Is highly teachable and manageable
- Is not a technique but it is an approach. What actions you perform next is governed by what you are doing currently



**Functionalities are checked
in a structured manner**



**Functionalities are checked
in a ad-hoc manner**

Black Box
(Specification Based)

- Based on requirements
- From the requirements, tests are created
- Specification Models can be used for systematic test case design

Techniques

- Equivalence Partitioning
- Boundary Value Analysis
- Decision Tables
- State Transition Testing
- Use Case Testing

White
(Structure Based)

- Based on code and the design of the system
- The tests provide the ability to derive the extent of coverage of the whole application

Techniques

- Statement coverage
- Branch Coverage
- Decision Coverage

Experience Based

- Based on the knowledge of the tester
- Using past experienced use & intuition to “guess” where errors may occur

Techniques

- Grey Box
- Error Guessing
- Exploratory Testing

Smoke and Sanity Testing

Introductions

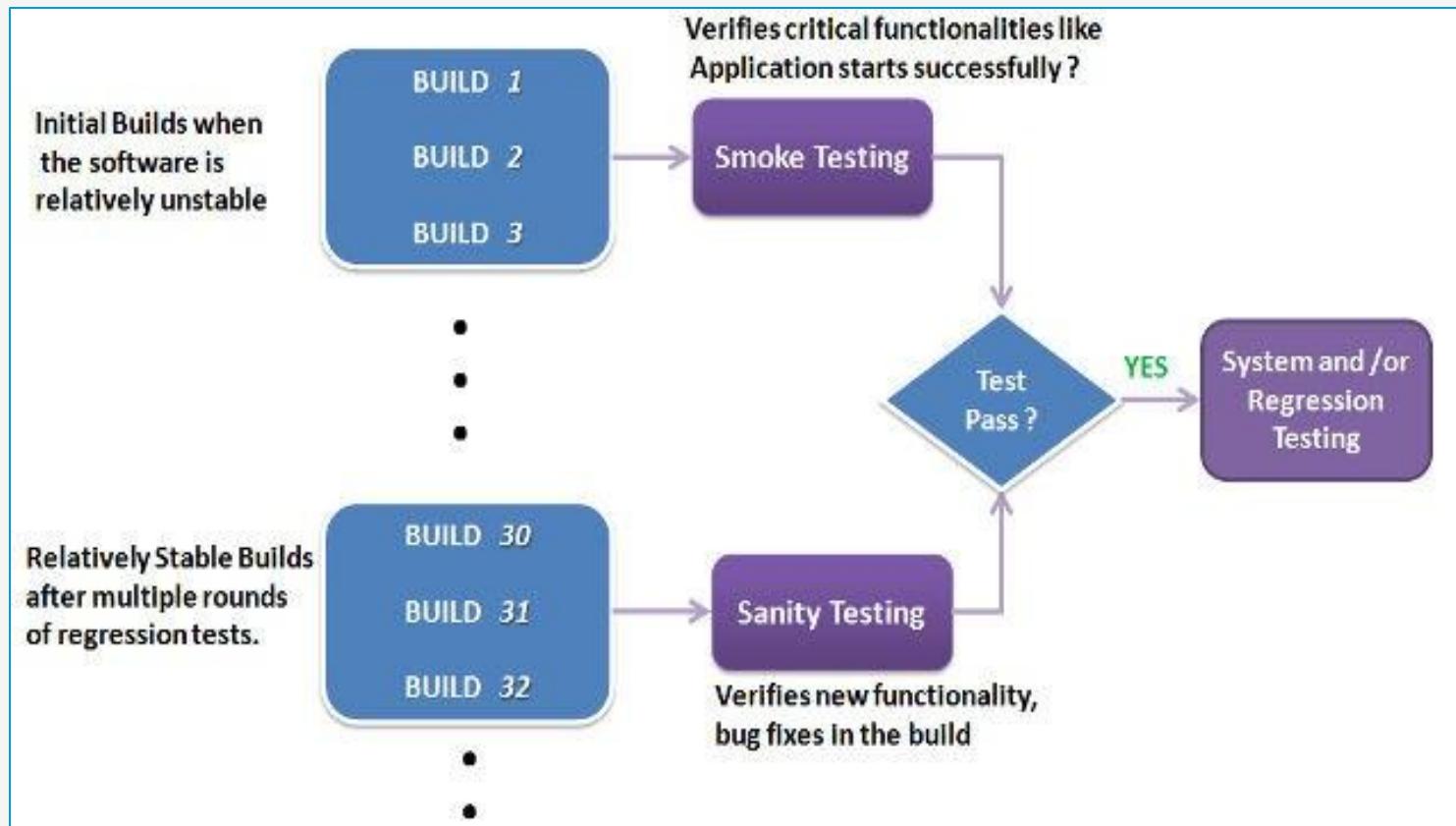
Smoke and Sanity testing are the most misunderstood topics in Software Testing. There is enormous amount of literature on the subject, but most of them are confusing. The following article makes an attempt to address the confusion.

Software Build

- If you are developing a simple computer program which consists of only one source code file, you merely need to compile and link this one file, to produce an executable file. This process is very simple.
- Usually this is not the case. A typical Software Project consists of hundreds or even thousands of source code files. Creating an executable program from these source files is a complicated and time-consuming task.
- You need to use "build" software to create an executable program and the process is called "**Software Build**"

Introductions

The key differences between Smoke and Sanity Testing can be learned with the help of following diagram –



Smoke Testing

- Smoke Testing is performed after software build to **ascertain that the critical functionalities of the program is working fine.**
- It is executed "**before**" any detailed functional or regression tests are executed on the software build.
- The **purpose is to reject a badly broken application**, so that the QA team does not waste time installing and testing the software application.
- In Smoke Testing, the **test cases chosen cover the most important functionality** or component of the system.
- The objective is not to perform exhaustive testing, but to verify that the critical functionalities of the system are working fine.
- For Example a typical smoke test would be – Verify that the application launches successfully, Check that the GUI is responsive ... etc.

Smoke Testing Examples

- **Web Based Testing :**

- New registration button is added in the login window and build is deployed with the new code. We perform smoke testing on a new build.

- **Desktop Based Testing :**

- If applicable in your SUT (System Under Test), as part of the smoke test you should try to successfully login with old and newly created credentials. Also, verify that you are able to successfully log out of the system without any errors

- **Mobile Based Testing :**

- If mobile is in 4g but now new mobile are you buying and new mobile is 5g so we check mobile 4g is working properly or not.

- **Game Based Testing :**

- When we play the game and make it score will be added in score but when we update the application but score not be changed

Sanity Testing

- After receiving a **software build**, with minor changes in code, or functionality, **Sanity testing is performed to ascertain that the bugs have been fixed and no further issues are introduced due to these changes.**
- The goal is to determine that the proposed functionality works roughly as expected.
- If sanity test fails, the build is rejected to save the time and costs involved in a more rigorous testing.**
- The **objective is "not" to verify thoroughly the new functionality**, but to determine that the developer has applied some rationality (sanity) while producing the software.
- For instance, if your scientific calculator gives the result of $2 + 2 = 5!$ Then, there is no point testing the advanced functionalities like $\sin 30 + \cos 50$.

Sanity Testing Examples

- **Web Based Testing :**

- In an e-commerce project, main modules are login page, home page, user profile page, user registration etc. There is a defect in the login page when the password field accepts less than four alpha numeric characters and the requirement mentions that this password field should not be below eight characters.

- **Desktop Based Testing :**

- If applicable in your SUT (System Under Test), as part of the smoke test you should try to successfully login with old and newly created credentials. Also, verify that you are able to successfully log out of the system without any errors

- **Mobile Based Testing :**

- Verify the device in different available networks like 2G, 3G, 4G or WIFI.
- Interrupt testing- Able to receive the calls while running the application

- **Game Based Testing :**

- When we play the game so new functionality not affect the other mobile functionality d

Smoke Testing vs Sanity Testing

Smoke Testing Sanity Testing

Smoke Testing is performed to ascertain that the critical functionalities of the program is working fine

The objective of this testing is to verify "stability" of the system in order to proceed with more rigorous testing

This testing is performed by the developers or testers

Smoke testing is usually documented or scripted

Smoke testing is a subset of Regression testing

Smoke testing exercises the entire system from end to end

Smoke testing is like General Health Check Up

Sanity Testing is done to check the new functionality / bugs have been fixed

The objective of the testing is to verify the "rationality" of the system in order to proceed with more rigorous testing

Sanity testing is usually performed by testers

Sanity testing is usually not documented and

is unscripted

Sanity testing is a subset of Acceptance testing

Sanity testing exercises only the particular component of the entire system

Sanity Testing is like specialized health check up

End to End Testing

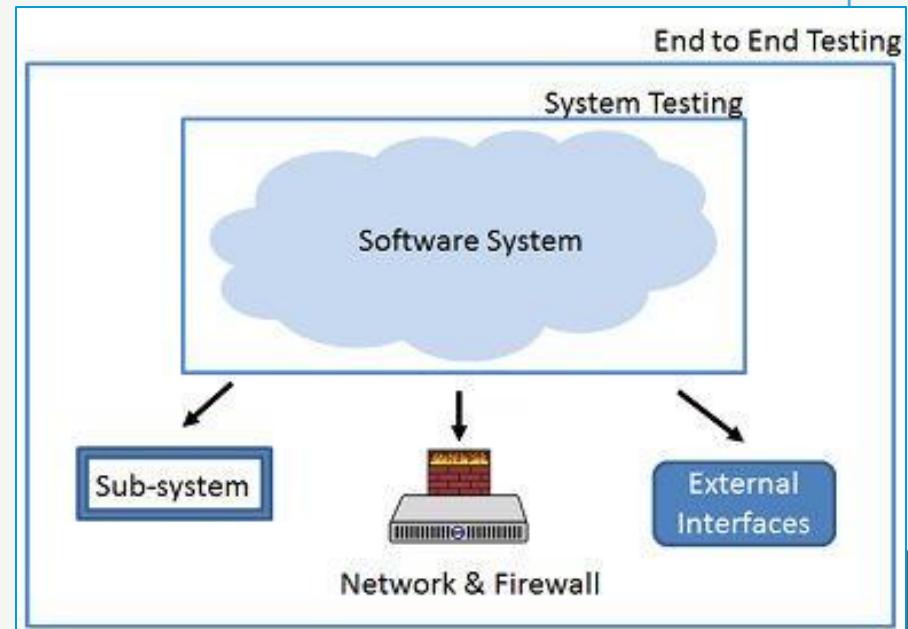
Introduction

- Unlike System Testing, End-to-End Testing not only validates the software system under test but also checks its integration with external interfaces.

- Hence, the name "**End-to-End**". The purpose of End-to-End Testing is to exercise a complete production-like scenario.

- Along with the software system, it also validates batch/data processing from other upstream/downstream systems.

- End to End Testing is usually executed after functional and system testing. It uses actual production like data and test environment to simulate real-time settings. End-to-End testing is also called **Chain Testing**

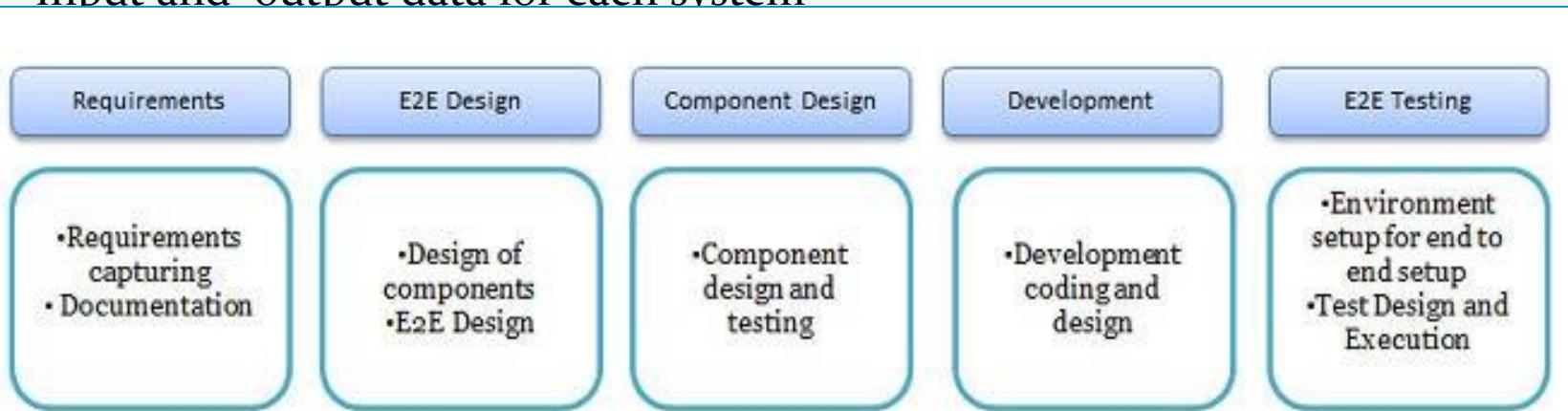


Why End-to End Testing?

- Modern software systems are complex and are interconnected with multiple sub-systems
- A sub-system may be different from the current system or may be owned by another organization.
- **If any one of the sub-system fails, the whole software system could collapse.**
- This is major risk and can be avoided by End-to-End testing.
- End-to-End testing verifies the complete system flow. It increase test coverage of various sub-systems.
- It helps detect issues with sub-systems and increases confidence in the overall software product.

End-to End Testing Process

- Study of end to end testing requirements
- Test Environment setup and hardware/software requirements
- Describe all the systems and its subsystems processes.
- Description of roles and responsibilities for all the systems
- Testing methodology and standards
- End to end requirements tracking and designing of test cases
- Input and output data for each system



End-to End Testing

End to End Testing Design

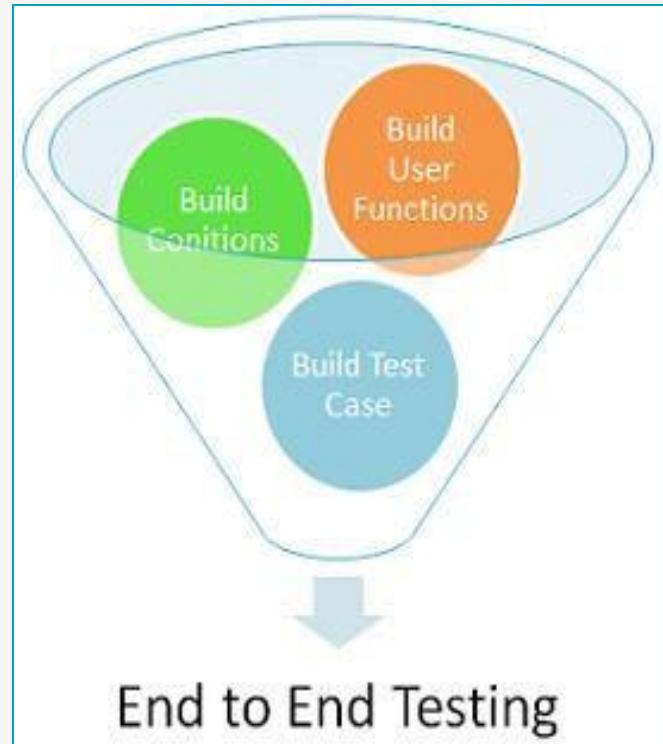
framework

consists of three parts

- Build user functions
- Build Conditions
- Build Test Cases

Metrics used for End to End Testing.

- Test Case preparation status
- Weekly Test Progress
- Defects Status & Details
- Environment Availability



Build User Functions

Following activities should be done as a part of build user functions:

- List down the features of the system and their interconnected components
- List the input data, action and the output data for each feature or function
- Identify the relationships between the functions
- Determine whether the function can be reusable or independent
- For example –Consider a scenario where you login into your bank account and transfer some money to another account from some other bank (3rdparty sub-system)
 - Login into the banking system
 - Check for the balance amount in the account
 - Transfer some amount from your account to some other bank account (3rdparty sub-system)
 - Check the your latest account balance
 - Logout of the application

Build Conditions

Following activities are performed as a part of build conditions:

- Building a set of conditions for each user function defined
- Conditions include sequence, timing and data conditions
- For example –Checking of more conditions like
 - **LOGIN PAGE**
 - Invalid User Name and Password
 - Checking with valid user name and password
 - Password strength checking
 - Checking of error messages
 - **BALANCE AMOUNT**
 - Check the current balance after 24 hours.(If the transfer is sent to different bank)
 - Check for the error message if the transfer amount is greater than the current balance amount
 - **BUILD TEST SCENARIO**
 - Login into the system
 - Check of bank balance amount
 - Transfer the bank balance amount

Build Test Case

- Build one or more test cases for each scenario defined .Test cases may include each condition as single test case.

End to End Testing Examples

- **Web Based Testing :**
 - Type URL into the address bar to launch the Gmail login page.
 - Log into account with valid credentials.
- **Desktop Based Testing :**
 - When punch machine connect with your desktop PC at that time your intime punch will be show in your system
- **Mobile Based Testing :**
 - A proper end-to-end mobile test plan ensures that the tested applications will function as planned on various devices with different screen sizes, resolutions, operating systems and internal hardware, as well as on different carrier networks.
- **Game Based Testing :**
 - :when remote will be connect with your game machine/ play station
 - When AR with connect with any Image or VR box connect with mobile Device

End-to End vs System Testing

End to End Testing

Validates the software system as well as interconnected sub-systems

Validates just the software system as per the requirements specifications.
It checks the complete end-to-end process and flow.

Features

All interfaces, backend systems will be Functional and Non-Functional

Testing considered for testing will be considered for testing

It's executed once system testing is completed.

It's executed after integration testing.

End to End testing involves checking external interfaces which can be complex to Both Manual and Automation can be automate. Hence Manual Testing is performed for system testing preferred.

System Testing

It checks system functionalities

Functionalities

Non - Functional Testing

- Usability Testing
- Compatibility Testing
- GUI Testing
- Security Testing
- Performance Testing
- Stress Testing
- Load Testing

Usability Testing

Need For Usability Testing

- Aesthetics and design are important. How well a product looks usually determines how well it works.
- There are many software applications / websites, which miserably fail, once launched, due to following reasons –

- Where do I click next?
- Which page needs to be navigated?
- Which Icon or Jargon represents what?
- Error messages are not consistent or effectively displayed
- Session time not sufficient.

Usability Testing identifies usability errors in the system early in development cycle and can save a product from failure.

Usability Testing Example

- **Web Based Testing , Desktop Based , Mobile based & Game based Testing :**
 - All fields on a page (For Example, text box, radio options, drop-down lists) should be aligned properly.
 - The user should not be able to type in drop-down select lists.
 - Tab and Shift+Tab order should work properly.
 - All buttons on a page should be accessible by keyboard shortcuts and the user should be able to perform all operations using a keyboard.
 - All buttons on a page should be accessible by keyboard shortcuts and the user should be able to perform all operations using a keyboard...
 - All pages should have a title.
 - Confirmation messages should be displayed before performing any update or delete operation.
 - Hourglass should be displayed when the application is busy.
 - Page text should be left-justified.
 - The user should be able to select only one radio option and any combination for checkboxes.

Goal of Usability Testing

- Effectiveness of the system
 - Efficiency
 - Accuracy
 - User Friendliness
-
- HOW MANY USERS DO YOU NEED?
 - 5(five) users are enough to uncover 80% of usability problems. Some researchers suggest other numbers. The truth is, the actual number of user required depends on the complexity of the given application and your usability goals. Increase in usability participant's results into increased cost, planning, participant management and data analysis.
 - But as a general guideline, if you on a small budget and interested in DIY usability testing 5 is a good number to start with. If budget is not a constraint its best consult experienced professionals to determine number of users.

Pros & Cons Usability Testing

Pros

- It helps uncover usability issues before the product is marketed.
- It helps improve end user satisfaction
- It makes your system highly effective and efficient
- It helps gather true feedback from your target audience who actually use your system during usability test. You do not need to rely on “opinions” from random people.

Cons

- Cost is a major consideration in usability testing. It takes lots of resources to set up a Usability Test Lab. Recruiting and management of usability testers can also be expensive
- However, these costs pay themselves up in form of higher customer satisfaction, retention and repeat business. Usability testing is therefore highly recommended.

Compatibility Testing

Introduction

- In computer world, compatibility is to check whether your software is capable of running on different hardware, operating systems, applications, network environments or mobile devices.

- Compatibility Testing is a type of the **Non-functional testing**.

Types of Compatibility Testing

- Hardware
- Operating Systems
- Software
- Network
- Browser
- Devices
- Mobile
- Versions of the software
 - Backward compatibility Testing
 - Forward compatibility Testing

Compatibility Testing Example

- **Web Based Testing & Desktop Based Testing :**
 - The application is compatible with different sizes such as RAM, hard disk, processor, and the graphic card, etc.
 - Check that the application is compatible with mobile platforms such as iOS, Android, etc.
 - Checking the compatibility of the software in the different network parameters such as operating speed, bandwidth, and capacity.
- **Mobile Based Testing :**
 - Define the platforms on which mobile app is likely to be used
 - Create the device compatibility library
 - Make a drawing of various environments, their hardware's, and software to figure out the behavior of the application in different configurations
 - Again perform the testing by following the same process, till no bugs can be found.
- **Game Based Testing :**
 - Validate whether the user interface of the app is as per the screen size of the device and ensure high quality
 - Ensures real compatibility between different testing environments with android phone and ipone

GUI Testing

Introduction

Graphical User Interface (GUI) testing is the process of testing the system's GUI of the System under Test. GUI testing involves checking the screens with the controls like menus, buttons, icons, and all types of bars – tool bar, menu bar, dialog boxes and windows etc.

WHAT DO YOU CHECK IN GUI TESTING?

- Check all the GUI elements for size, position, width, length and acceptance of characters or numbers. For instance, you must be able to provide inputs to the input fields.
- Check you can execute the intended functionality of the application using the GUI
- Check Error Messages are displayed correctly
- Check for Clear demarcation of different sections on screen
- Check Font used in application is readable
- Check the alignment of the text is proper
- Check the Color of the font and warning messages is aesthetically pleasing
- Check that the images have good clarity
- Check that the images are properly aligned
- Check the positioning of GUI elements for different screen resolution.

Approach of GUI Testing

MANUAL BASED TESTING

Under this approach, graphical screens are checked manually by testers in conformance with the requirements stated in business requirements document.

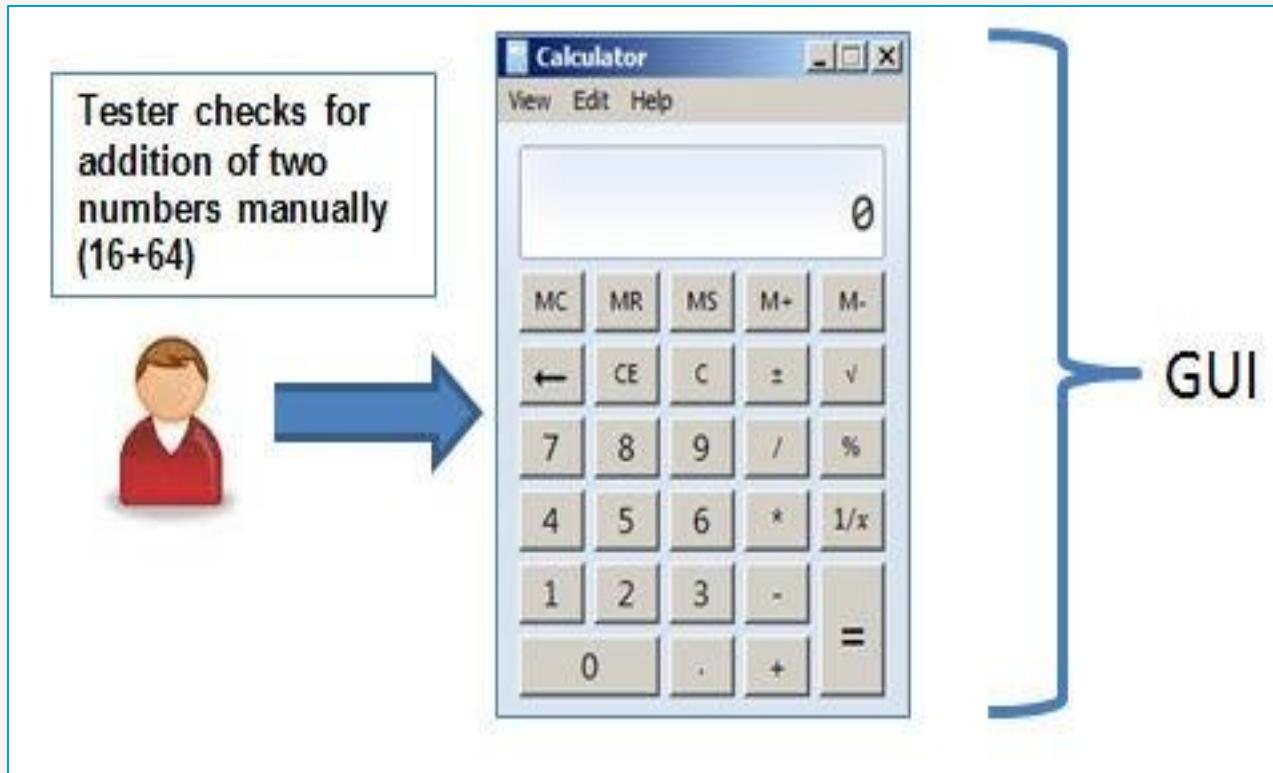
RECORD AND REPLAY

GUI testing can be done using automation tools. This is done in 2 parts. During Record , test steps are captured into the automation tool. During playback, the recorded test steps are executed on the Application under Test. Example of such tools - QTP.

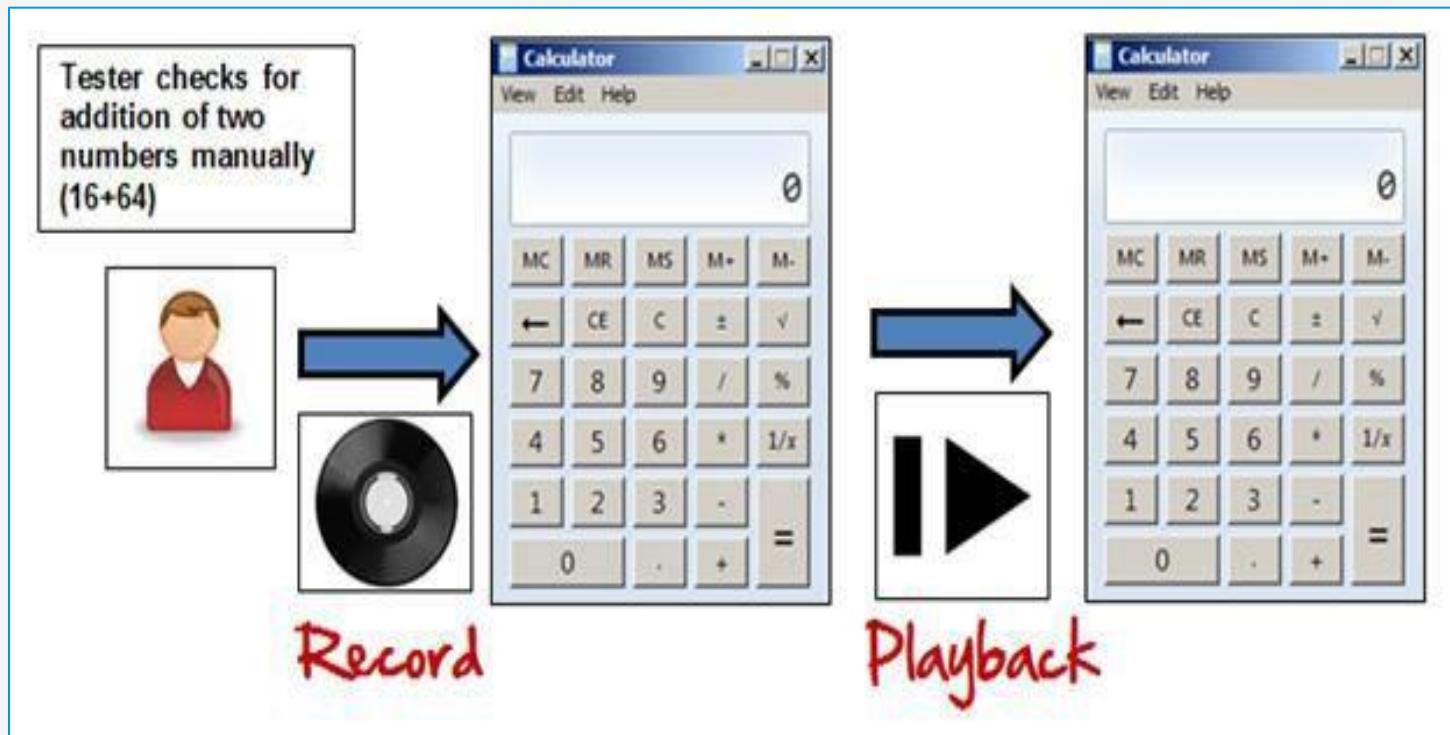
MODEL BASED TESTING

A model is a graphical description of system's behavior. It helps us to understand and predict the system behavior. Models help in a generation of efficient test cases using the system requirements.

Manual Based Testing in GUI



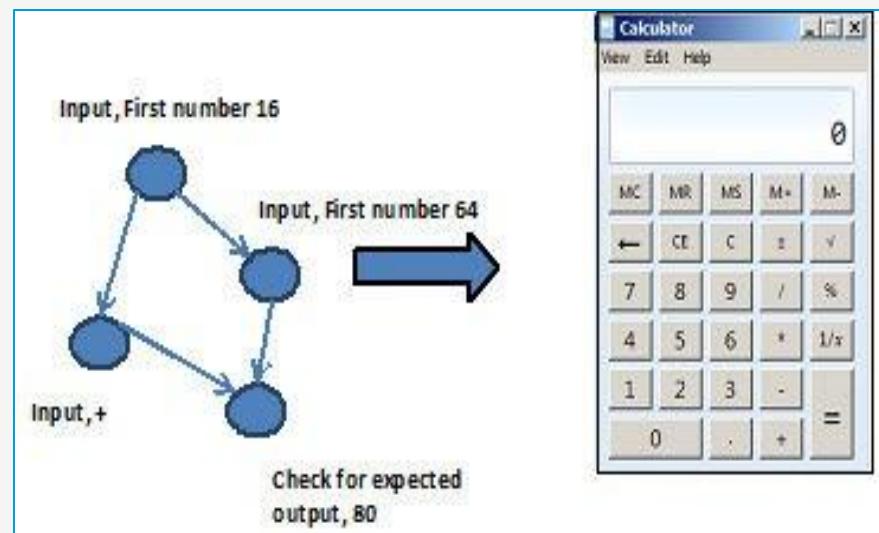
Record & Play in GUI



Model Based Testing in GUI

- Build the model
- Determine Inputs for the model
- Calculate expected output for the model
- Run the Tests
- Compare the actual output with the expected output
- Decision on further action on the model
- Some of the modeling techniques from which test cases can be derived:
 - Charts – Depicts the state of a system and checks the state after some input.
 - Decision Tables – Tables used to determine results for each input applied

- Model based Testing is an evolving technique for the generating the test cases from the requirements.
- Its main advantage, compared to above two methods, is that it can determine undesirable states that your GUI can attain.



GUI Testing Examples

- **Web Based Testing & Desktop Based Testing :**
 - The scrollbar should be enabled only when necessary.
 - Font size, style, and color for headline, description text, labels, infield data, and grid info should be standard as specified in SRS.
 - The description text box should be multi-lined.
 - Enough space should be provided between field labels, columns, rows, error messages, etc.
- **Mobile Based Testing :**
 - If mobile is in every orientation mode so display image , video properly.
 - Every app will display in responsive type .
 - Alignment should be apply properly of every field.
- **Game Based Testing :**
 - Game infra design will showing properly
 - Game points or score will display proper with its background color.
 - Game sound manage with its background effect
 - Can be also conducted in advance of designing page layouts or navigation menus

Security Testing

Introduction

- You need to test how secure your web application is from both external and internal threats.
- The security of your web application should be planned for and verified by qualified security specialists.
- Security is set of measures to protect an application against unforeseen actions that cause it to stop functioning or being exploited.
- Security Testing ensures that system and applications in an organization are free from any loopholes that may cause a big loss.
- Security testing of any system is about finding all possible loopholes and weaknesses of the system which might result into loss of information at the hands of the employees or outsiders of the Organization.
- **The goal of security testing is to identify the threats in the system and measure its potential vulnerabilities.**
- It also helps in detecting all possible security risks in the system and help developers in fixing these problems through coding.

Types of Security Testing

- **Vulnerability Scanning**
- **Security Scanning**
- **Risk Assessment**
- **Security Auditing**
- **Ethical hacking**
- **Posture Assessment**

ROLES YOU MUST KNOW!

- **Hackers** - Access computer system or network without authorization
- **Crackers** – Break into the systems to steal or destroy data
- **Ethical Hacker** – Performs most of the breaking activities but with permission from owner
- **Script Kiddies or packet monkeys** – Inexperienced Hackers with programming language skill

Security with SDLC

SDLC Phases	Security Processes
Requirements	Security analysis for requirements and check abuse/misuse cases
Design	Security risk analysis for designing. Development of test plan including security tests
Coding and Unit Testing	Static and Dynamic Testing and Security white box testing
Integration Testing	Black Box Testing
System Testing	Black Box Testing and Vulnerability scanning
Implementation	Penetration Testing, Vulnerability Scanning
Support	Impact analysis of Patches

Security Testing Techniques

- In security testing, different methodologies are followed, and they are as follows:
 - **Tiger Box:** This hacking is usually done on a laptop which has a collection of OSs and hacking tools. This testing helps penetration testers and security testers to conduct vulnerabilities assessment and attacks.
 - **Black Box:** Tester is authorized to do testing on everything about the
 - network topology and the technology.
 - **Grey Box:** Partial information is given to the tester about the system, and it is hybrid of white and black box models.

Security Testing Examples

- **Web Based Testing & Desktop Based Testing :**
 - Secure pages should use the HTTPS protocol.
 - Sensitive fields like passwords and credit card information should not have to autocomplete enabled.
 - Verify CAPTCHA functionality.
- **Mobile Based Testing :**
 - Every payment gateway app used security code or OTP
 - Must be used mobile tracking mode on
- **Game Based Testing :**
 - Secure your rewards which you get from your game playing
 - Without current round completing , you can not going in next round.

Performance Testing

Introduction

Software performance testing is a means of quality assurance (QA). It involves testing software applications to ensure they will perform well under their expected workload.

Features and Functionality supported by a software system is not the only concern. A software application's performance like its response time, do matter. The goal of performance testing is not to find bugs but to eliminate performance bottlenecks

The focus of Performance testing is checking a software programs

- **Speed** – Determines whether the application responds quickly
- **Scalability** – Determines maximum user load the software application can handle.
- **Stability** – Determines if the application is stable under varying loads

Types of Performance Testing

- Load testing
- Stress testing
- Endurance testing
- Spike testing
- Volume testing
- Scalability testing

Performance Problems

Long Load time -

- Load time is normally the initial time it takes an application to start.
- This should generally be kept to a minimum.
- While some applications are impossible to make load in under a minute, Load time should be kept under a few seconds if possible.

Poor response time -

- Response time is the time it takes from when a user inputs data into the application until the application outputs a response to that input.
- Generally this should be very quick.
- Again if a user has to wait too long, they lose interest.

Poor scalability -

- A software product suffers from poor scalability when it cannot handle the expected number of users or when it does not accommodate a wide enough range of users.
- Load testing should be done to be certain the application can handle the anticipated number of users.

Performance Problems

Bottlenecking –

- Bottlenecks are obstructions in system which degrade overall system performance.
- Bottlenecking is when either coding errors or hardware issues cause a decrease of throughput under certain loads.
- **Bottlenecking is often caused by one faulty section of code.**
- The key to fixing a bottlenecking issue is to find the section of code that is causing the slowdown and try to fix it there.
- Bottlenecking is generally fixed by either fixing poor running processes or adding additional Hardware.
- Some **common performance bottlenecks** are
 - CPU utilization
 - Memory utilization
 - Network utilization
 - Operating System limitations
 - Disk usage

Performance Parameters

Processor Usage

- Bandwidth
- Private bytes
- Committed memory
- Memory pages/second
- Page faults/second
- CPU interrupts per second
- Disk queue length
- Network output queue length
- Network bytes total per second
- Response time

- Throughput
- Amount of connection pooling
- Maximum active sessions
- Hit ratios
- Hits per second
- Rollback segment
- Database locks
- Top waits
- Thread counts
- Garbage collection

Performance Test Tools

HP Load runner –

- is the most popular performance testing tools on the market today.
- This tool is capable of simulating hundreds of thousands of users, putting applications under real life loads to determine their behavior under expected loads.
- Load runner features a virtual user generator which simulates the actions of live human users.

HTTP Load –

- a throughput testing tool aimed at testing web servers by running several http or https fetches simultaneously to determine how a server handles the workload.

Proxy Sniffer –

- one of the leading tools used for load testing of web and application servers.
- It is a cloud based tool that's capable of simulating thousands of users.

Performance Testing Examples

- **Web Based Testing & Desktop Based Testing :**
 - Check the page load on slow connections.
 - Check the database query execution time.
 - Check the performance of database stored procedures and triggers.
- **Mobile Based Testing :**
 - Check the database query execution time.
 - Check the response time for any action under a light, normal, moderate, and heavy load conditions.
 - Check CPU and memory usage under peak load conditions
- **Game Based Testing :**
 - Determine whether the current infrastructure is sufficient for the smooth running of the game
 - Determine the number of users an app can support and its scalability rate to support more users
 - Accommodates strategies for performance management.

Stress Testing

Introduction

- **Stress testing** - System is stressed beyond its specifications to check how and when it fails. Performed under heavy load like putting large number beyond storage capacity, complex database queries, continuous input to system or database load.
- Stress testing is used to test the stability & reliability of the system. This test mainly determines the system on its robustness and error handling under extremely heavy load conditions.
- **It even tests beyond the normal operating point and evaluates how the system works under those extreme conditions.**
- **Stress Testing is done to make sure that the system would not crash under crunch situations.**
- **Stress testing is also known as endurance testing.**

Introduction

- Under Stress Testing, AUT is stressed for a short period of time to know its withstanding capacity.
- Most prominent use **of stress testing is to determine the limit, at which the system or software or hardware breaks.**
- It also checks whether system demonstrates effective error management under extreme conditions.
- The application under testing will be stressed when 5GB data is copied from the website and pasted in notepad.
- Notepad is under stress and gives ‘Not Responded’ error message.
- Examples of stress conditions include:
 - Excessive volume in terms of either users or data; examples might include a denial of service (DoS) attack or a situation where a widely viewed news item prompts a large number of users to visit a Web site during a three-minute period.
 - Resource reduction such as a disk drive failure.
 - Application components fail to respond.

Need For Stress Testing

- During festival time, an online shopping site may witness a spike in traffic, or when it announces a sale.
- When a blog is mentioned in a leading newspaper, it experiences a sudden surge in traffic.
- To check whether the system works under abnormal conditions.
- Displaying appropriate error message when the system is under stress.
- System failure under extreme conditions could result in enormous revenue loss
- It is better to be prepared for extreme conditions by executing Stress Testing.

Goal of Stress Testing

- The goal of stress testing is to analyze the behavior of the system after failure. For stress testing to be successful, system should display appropriate error message while it is under extreme conditions.
- To conduct Stress Testing, sometimes, massive data sets may be used which may get lost during Stress Testing.
- Testers should not lose this security related data while doing stress testing.
- The main purpose of stress testing is to make sure that the system recovers after failure which is called as **recoverability**.

Types of Stress Testing

- Application Stress Testing:
- Transactional Stress Testing:
- Systemic Stress Testing:
- Exploratory Stress Testing:

Stress Testing

- Stress Tester
- Neo Load
- App Perfect

Metrics for Stress Testing

Measuring Scalability & Performance

- **Pages per Second:** Measures how many pages have been requested / Second
- **Throughput:** Basic Metric – Response data size/Second
- **Rounds:** Number of times test scenarios has been planned Versus Number of times client has executed

Application Response

- **Hit time:** Average time to retrieve an image or a page
- **Time to the first byte:** Time taken to return the first byte of data or information
- **Page Time:** Time taken to retrieve all the information in a page

Failures

- **Failed Connections:** Number of failed connections refused by the client
- **Failed Rounds:** Number of rounds it gets failed
- **Failed Hits:** Number of failed attempts done by the system

Load Testing

Introduction

Load testing - Its a performance testing to check system behavior under load. Testing an application under heavy loads, such as testing of a web site under a range of loads to determine at what point the system's response time degrades or fails.

Load testing is a kind of performance testing which determines a system's performance under real-life load conditions. This testing helps determine how the application behaves when multiple users access it simultaneously.

This testing usually identifies –

- The maximum operating capacity of an application
- Determine whether current infrastructure is sufficient to run the application
- Sustainability of application with respect to peak user load
- Number of concurrent users that an application can support, and scalability to allow more users to access it.
- It is a type of non-functional testing. Load testing is commonly used for the Client/Server, Web based applications – both Intranet and Internet.

Need For Load Testing

Some extremely popular sites have suffered serious downtimes when they get massive traffic volumes. E-commerce websites invest heavily in advertising campaigns, but not in Load Testing to ensure optimal system performance, when that marketing brings in traffic.

For Example

- Popular toy store **Toysrus.com**, could not handle the increased traffic generated by their advertising campaign resulting in loss of both marketing dollars, and potential toy sales.
- An Airline website was not able to handle 10000+ users during a festival offer.
- Encyclopedia Britannica declared free access to their online database as a promotional offer. They were not able to keep up with the onslaught of traffic for weeks.
- Facebook(FB)

Why Load Testing?

- Load testing gives confidence in the system & its reliability and performance.
- Load Testing helps identify the bottlenecks in the system under heavy user stress scenarios before they happen in a production environment.
- Load testing gives excellent protection against poor performance and accommodates complementary strategies for performance management and monitoring of a production environment.

Goals of Load Testing

● Loading testing identifies the following problems before moving the application to market or Production:

- Response time for each transaction
- Performance of System components under various loads
- Performance of Database components under different loads
- Network delay between the client and the server
- Software design issues
- Server configuration issues like Web server, application server, database server etc.
- Hardware limitation issues like CPU maximization, memory limitations, network bottleneck, etc.

● Load testing will determine whether system needs to be fine-tuned or modification of hardware and software is required to improve performance.

Pre-Requisites for Load Testing

The chief metric for load testing is response time. Before you begin load testing, you must determine -

- Whether the response time is already measured and compared – Quantitative
- Whether the response time is applicable to the business process – Relevant
- Whether the response time is justifiable – Realistic
- Whether the response time is achievable – Achievable
- Whether the response time is measurable using a tool or stopwatch – Measurable

Hardware Platform Software Configuration

- | | |
|--|--|
| <ul style="list-style-type: none">· Server Machines· Processors· Memory· Disk Storage· Load Machines configuration· Network configuration | <ul style="list-style-type: none">· Operating System· Server Software |
|--|--|

Strategies of Load Testing

- Manual Load Testing
- In house(Organization) developed load testing tools
- Open source load testing tools
- Enterprise(Record and Play) load testing tools

Load Testing

- Loadrunner
- Web Load
- Astra Load Test
- Review's Web Load
- Studio, Rational Site Load
- Silk Performer

Pros and Cons of Load Testing

Pros:

- Performance bottlenecks identification before production
- Improves the scalability of the system
- Minimize risk related to system down time
- Reduced costs of failure
- Increase customer satisfaction

Cons:

- Need programming knowledge to use load testing tools.
- Tools can be expensive as pricing depends on the number of virtual users supported.

Load Testing vs Stress Testing

Load Testing Stress Testing

Load Testing is to test the system behavior under normal workload conditions, and it is just testing or simulating with the actual workload.

Load testing identifies the bottlenecks in Stress testing determines the breaking the system under various workloads and point of the system to reveal the maximum checks how the system reacts when the point after which it breaks. load is gradually increased

Load testing does not break the system

Stress testing is to test the system behavior under extreme conditions and is carried out till the system failure.

Stress testing tries to break the system by testing with overwhelming data or resources.

Re-Testing and Regression Testing

Introduction

- The purpose of regression testing is to confirm that a recent program or code change has not adversely affected existing features.

- Regression testing is nothing but full or partial selection of already executed test cases which are re-executed to ensure existing functionalities work fine.

- This testing is done to make sure that new code changes should not have side effects on the existing functionalities. It ensures that old code still works once the new code changes are done.

Confirmation Testing (Re-Testing)

- **Re-testing: Testing that runs test cases that failed the last time they were run, in order to verify the success of corrective actions**
- Whenever a fault is detected and fixed then the software should be re-tested to show that the original fault has been fixed. This is known as Re-Testing.
- It is important that the test case is repeatable.
- In order to support this the test identifier should be included on the fault report.
- It is important that the environment and test data used are as close as possible as those used during the original test.

Regression Testing

Regression Testing: Testing of a previously tested program following modification to ensure that defects have not been introduced or uncovered in unchanged areas of the software, as a result of the changes made. It is performed when the software or its environment is changed.

- If the test is re-run and passes you cannot necessarily say the fault has been resolved because ..
- You also need to ensure that the modifications have not caused unintended side-effects elsewhere and that the modified system still meets its requirements – Regression Testing

Regression Testing

Regression testing should be carried out:

- when the system is stable and the system or the environment changes
- when testing bug-fix releases as part of the maintenance phase
- It should be applied at all Test Levels
- It should be considered complete when agreed completion criteria for regression testing have been met
- Regression test suites evolve over time and given that they are run frequently are ideal candidates for automation

Need of Regression Testing

- Change in requirements and code is modified according to the requirement
- New feature is added to the software
- Defect fixing
- Performance issue fix

Difference between Re-Testing and Regression Testing

- Retesting means testing the functionality or bug again to ensure the code is fixed. If it is not fixed, defect needs to be re-opened. If fixed, defect is closed.
- Regression testing means testing your software application when it undergoes a code change to ensure that the new code has not affected other parts of the software.

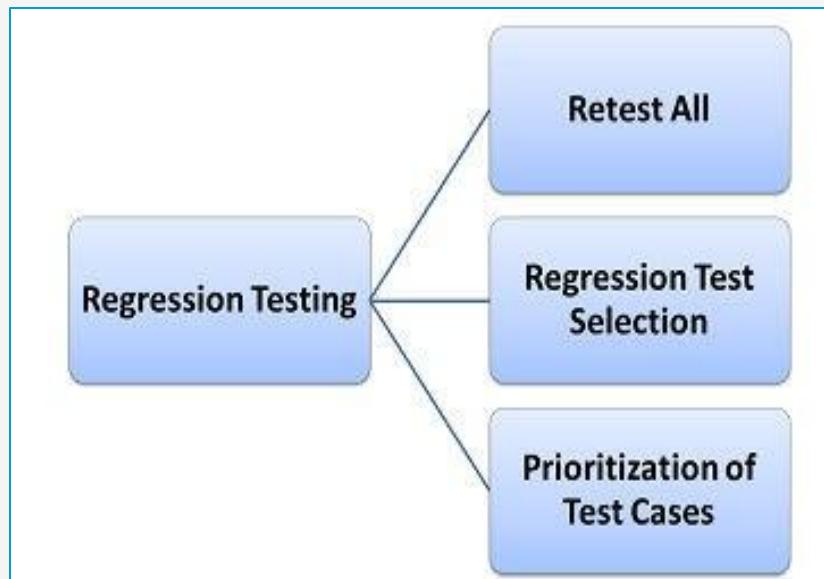
Regression Testing Techniques

Software maintenance is an activity which includes enhancements, error corrections, optimization and deletion of existing features.

These modifications may cause the system to work incorrectly.

Therefore, Regression Testing becomes necessary.

Regression Testing can be carried out using following techniques:



Regression Testing Techniques

Retest All

- This is one of the methods for regression testing in which all the tests in the existing test bucket or suite should be re-executed. This is very expensive as it requires huge time and resources.

Regression Test Selection

- Instead of re-executing the entire test suite, it is better to select part of test suite to be run
- Test cases selected can be categorized as 1) Reusable Test Cases 2) Obsolete Test Cases.
- Re-usable Test cases can be used in succeeding regression cycles.
- Obsolete Test Cases can't be used in succeeding cycles.

Prioritization Of Test Cases

- Prioritize the test cases depending on business impact, critical & frequently used functionalities. Selection of test cases based on priority will greatly reduce the regression test suite.

Challenges Regression Testing

- With successive regression runs, test suites become fairly large. Due to time and budget constraints, the entire regression test suite cannot be executed
- Minimizing test suite while achieving maximum test coverage remains a challenge
- Determination of frequency of Regression Tests, i.e., after every modification or every build update or after a bunch of bug fixes, is a challenge.



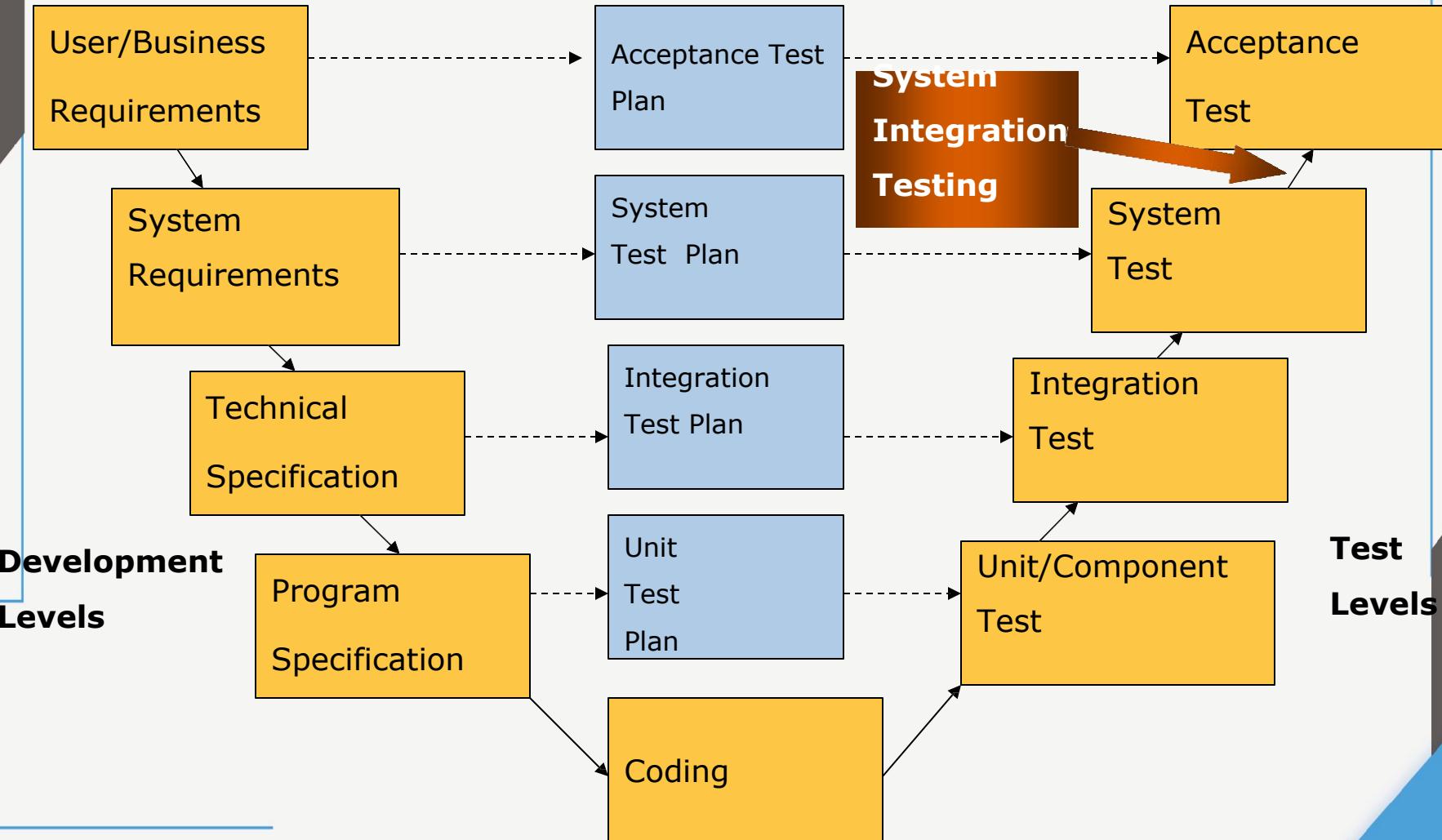
Regression Testing Tools

- **Quick Test Professional (QTP)**
- **Rational Functional Tester (RFT)**
- **Selenium**

When use the testing tools?

- If your software undergoes frequent changes, regression testing costs will escalate.
- In such cases, Manual execution of test cases increases test execution time as well as costs.
- Automation of regression test cases is the smart choice in such cases.
- Extent of automation depends on the number of test cases that remain reusable for successive regression cycles.

System Integration Testing



System Integration Testing

System Integration Testing is testing between the ‘System’ and ‘Acceptance’ phases.

The System has already proven to be functionally correct, what remains to be tested is how the system reacts to other systems and/or organisations.

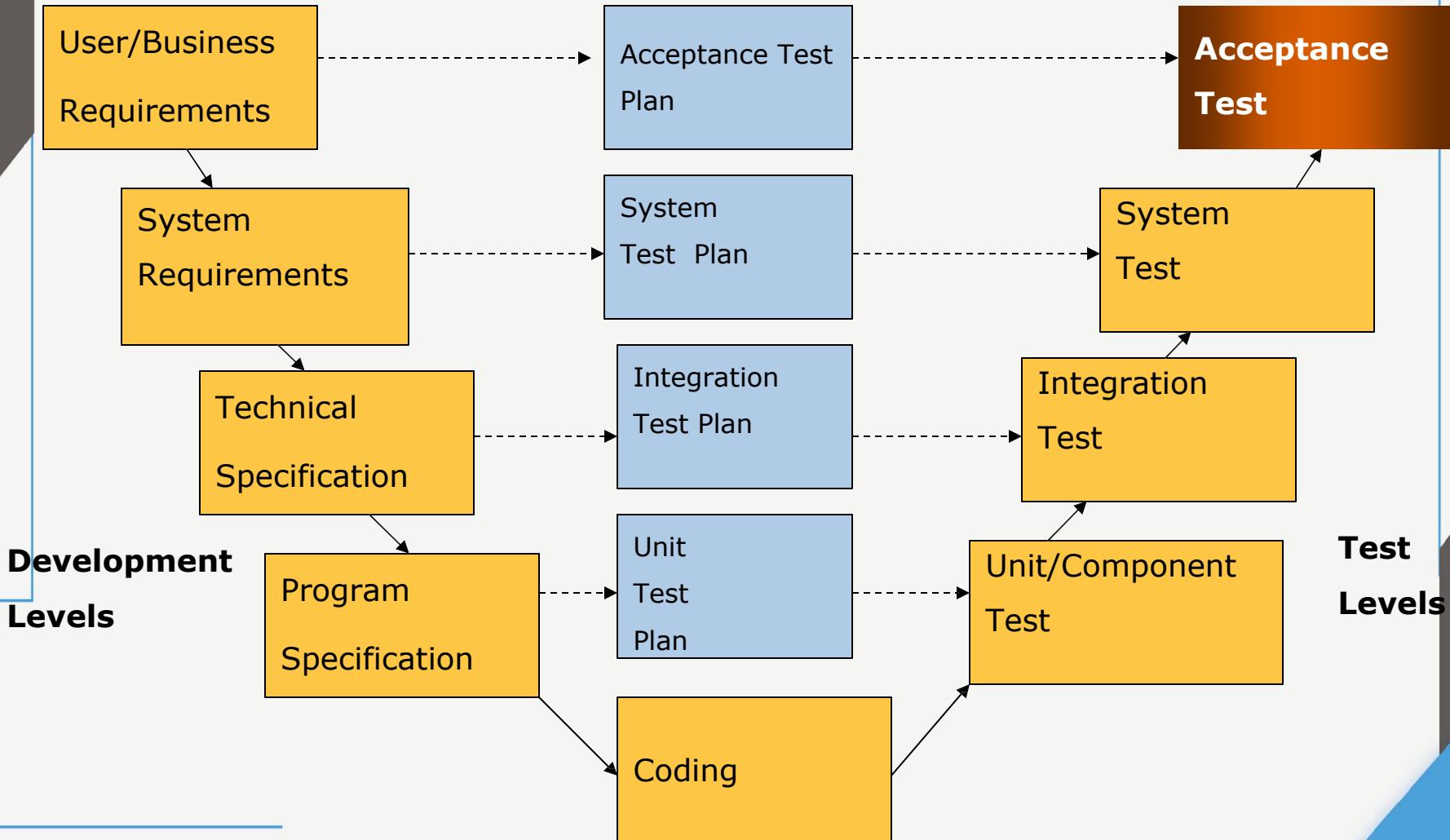
The **objective** of System Integration Testing is to provide confidence that the **system or application** is able to interoperate successfully with other specified software systems and does not have an adverse affect on other systems that may also be present in the **live environment**, or vice versa

System Integration Testing

It is possible that the testing tasks performed during System Integration Testing may be combined with **System Testing**, particularly if the system or application has little or no requirement to interoperate with other systems

- In terms of the *V Model*, Systems Integration Testing corresponds to the Functional and Technical Specification phases of the software development lifecycle
- Having completed Component integration testing and Systems testing, one must execute the plan for system-to-system integration
- Infrastructure may need to be transformed in order to feed to an external system
- Black Box testing techniques used

Acceptance Testing



User Acceptance Testing

Acceptance testing: Formal testing with respect to user needs, requirements, and business processes conducted to determine whether or not a system satisfies the acceptance criteria and to enable the user, customers or other authorized entity to determine whether or not to accept the system.

● **Acceptance Testing** is a level of the software testing process where a system is tested for acceptability.

● The purpose of this test is to evaluate the system's compliance with the business requirements and assess whether it is acceptable for delivery.

● After the system test has corrected all or most defects, the system will be delivered to the user or customer for acceptance testing.

● Acceptance testing is basically done by the user or customer although other stakeholders may be involved as well.

● The **goal** of acceptance testing is to **establish confidence in the system**.

● Acceptance testing is most often **focused on a validation type testing**.

● Usually, **Black Box Testing method is used in Acceptance Testing**.

● Testing does not usually follow a strict procedure and is not scripted but is rather **ad-hoc**.

Acceptance(Thread) Testing

- Often uses the “Thread Testing” approach:

- A testing technique used to test the business functionality or business logic of the application in an end-to-end manner, in much the same way a User or an operator might interact with the system during its normal use.’*

- This approach is also often used for functional system test

- The same Threads server both test activates

- Often use big bang approach

- Regression testing to ensure changes have not regressed other areas of the system.

Alpha Testing

- It is always performed by the developers at the software development site.
- Sometimes it is also performed by Independent Testing Team.
- Alpha Testing is not open to the market and public
- It is conducted for the software application and project.
- It is always performed in **Virtual Environment**.
- It is always performed within the organization.
- It is the form of Acceptance Testing.
- Alpha Testing is definitely performed and carried out at the developing organizations location with the involvement of developers.
- It comes under the category of both White Box Testing and Black Box Testing.

Beta Testing(Field Testing)

- It is always performed by the customers at their own site.
- It is not performed by Independent Testing Team.
- Beta Testing is always open to the market and public.
- It is usually conducted for software product.
- It is performed in **Real Time Environment**.
- It is always performed outside the organization.
- It is also the form of Acceptance Testing.
- Beta Testing (field testing) is performed and carried out by users or you can say people at their own locations and site using customer data.
- It is only a kind of Black Box Testing.

Beta Testing(Field Testing)

- Beta Testing is always performed at the time when software product and project are marketed.
- It is always performed at the user's premises in the absence of the development team.
- It is also considered as the User Acceptance Testing (UAT) which is done at customers or users area.
- Beta testing can be considered “**pre-release**” testing.
- **Pilot Testing** is testing to product on real world as well as collect data on the use of product in the classroom.

Testing Definitions as per ISTQB

- **Unit testing or Component testing:** The testing of individual software components.
- **Integration testing:** Testing performed to expose defects in the interfaces and in the interactions between integrated components or systems. See also component integration testing, system integration testing.
- **Component integration testing:** Testing performed to expose defects in the interfaces and interaction between integrated components.
- **System integration testing:** Testing the integration of systems and packages; testing interfaces to external organizations (e.g. Electronic Data Interchange, Internet).
- **System testing:** The process of testing an integrated system to verify that it meets specified requirements.
- **Acceptance testing:** Formal testing with respect to user needs, requirements, and business processes conducted to determine whether or not a system satisfies the acceptance criteria and to enable the user, customers or other authorized entity to determine whether or not to accept the system.

Maintenance Testing

Maintenance Testing

Maintenance testing: Testing the changes to an operational system or the impact of a changed environment to an operational system

- testing changes to a Live System

- Triggered by, for example,

- Modification**

- software upgrades
 - Operating system changes
 - system tuning
 - emergency fixes

- Software Retirement** (may necessitate data archiving tests)

- Migration**

- System migration (including operational tests of new environment plus changed software)
 - database migration

Objectives of Maintenance

Testing

- Develop tests to detect problems prior to placing the change into production
- Correct problems identified in the live environment
- Test the completeness of needed training material
- Involve users in the testing of software changes

Types of Maintenance Testing

- **Corrective maintenance:** identifying and repairing defects
- **Adaptive maintenance:** adapting the existing solution to the new platforms.
- **Perfective Maintenance:** implementing the new requirements

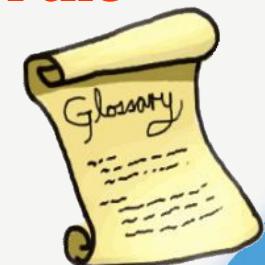
Test case Authoring Or Static Testing

Static Testing

Static Testing: Static testing techniques involve examination of the project's documentation, software and other information about the software products without executing them

- Static Testing Includes both Reviews (e.g. of documentation) and Static Analysis of code
- Reviews, Static Analysis and dynamic testing have the same objective – identifying defects
- Static Testing and Dynamic Testing are complementary each technique can find different types of defects effectively and efficiently

Dynamic testing: Testing that involves the execution of the software of a component or system.



Static Testing(Cont...)

Static testing techniques rely on the manual examination (reviews) and automated analysis (static analysis) of the code or other project documentation.

Reviews are a way of testing software work products (including code) and can be performed well before dynamic test execution.

Defects detected during reviews early in the life cycle are often much cheaper to remove than those detected while running tests (e.g. defects found in requirements).

A review could be done entirely as a manual activity, but there is also tool support available.

The main manual activity is to examine a work product and make comments about it.

Use of Static Testing

Since static testing can start early in the life cycle so early feedback on quality issues can be established.

As the defects are getting detected at an early stage so the rework cost most often relatively low.

Development productivity is likely to increase because of the less rework effort.

Types of the defects that are easier to find during the static testing are:

- deviation from standards,
- missing requirements,
- design defects,
- non-maintainable code,
- inconsistent interface specifications.

Informal Reviews

- Informal reviews are applied many times during the early stages of the life cycle of the document.
- A two person team can conduct an informal review. In later stages these reviews often involve more people and a meeting.
- The goal is to keep the author and to improve the quality of the document.
- The most important thing to keep in mind about the informal reviews is that they are **not documented**.

Formal Reviews

- Formal reviews follow a formal process. It is well structured and regulated.

- A formal review process consists of six main steps:

- Planning
- Kick-off
- Preparation
- Review meeting
- Rework
- Follow-up

Formal Review - Planning

- The first phase of the formal review is the Planning phase.
- In this phase the review process begins with a request for review by the author to the moderator (or inspection leader).
- A moderator has to take care of the scheduling like date, time, place and invitation of the review.
- For the formal reviews the moderator performs the entry check and also defines the formal exit criteria.
- The **entry check** is done to ensure that the reviewer's time is not wasted on a document that is not ready for review.
- After doing the entry check if the document is found to have very little defects
then it's ready to go for the reviews.

Formal Review - Planning

So, the **entry criteria** are to check that whether the document is ready to enter the formal review process or not. Hence the entry criteria for any document to go for the reviews are:

- The documents should not reveal a large number of major defects.
- The documents to be reviewed should be with line numbers.
- The documents should be cleaned up by running any automated checks that apply.
- The author should feel confident about the quality of the document so that he can join the review team with that document.

Once, the document clear the entry check the moderator and author decides that which part of the document is to be reviewed.

Since the human mind can understand only a limited set of pages at one time so in a review the maximum size is between 10 and 20 pages.

Hence checking the documents improves the moderator ability to lead the meeting because it ensures the better understanding.

Formal Review – Kick Off

- This kick-off meeting is an optional step in a review procedure.
- The goal of this step is to give a short introduction on the objectives of the review and the documents to everyone in the meeting.
- The relationships between the document under review and the other documents are also explained, especially if the numbers of related documents are high.
- At customer sites, we have measured results up to 70% more major defects found per page as a result of performing a kick-off.

Formal Review – Preparation

- In this step the reviewers review the document individually using the related documents, procedures, rules and checklists provided.
- Each participant while reviewing individually identifies the defects, questions and comments according to their understanding of the document and role.
- After that all issues are recorded using a logging form.
- The success factor for a thorough preparation is the number of pages checked per hour. This is called the **checking rate**.
- Usually the checking rate is in the range of 5 to 10 pages per hour.

Formal Review – Review Meeting

The review meeting consists of **three phases**:

- **Logging phase:**
 - In this phase the issues and the defects that have been identified during the preparation step are logged page by page.
 - The logging is basically done by the author or by a **scribe**.
 - Scribe is a separate person to do the logging and is especially useful for the formal review types such as an inspection.
 - Every defects and its severity should be logged in any of the three severity classes given below:
 - **Critical:** The defects **will cause** downstream damage.
 - **Major:** The defects **could cause** a downstream damage.
 - **Minor:** The defects are **highly unlikely to cause** the downstream damage.
 - During the logging phase the moderator focuses on logging as many defects as possible within a certain time frame and tries to keep a good logging rate (number of defects logged per minute).
 - In formal review meeting the good logging rate should be between one and two defects logged per minute.

Formal Review – Review Meeting

The review meeting consists of **three phases**:

Discussion phase:

- If any issue needs discussion then the item is logged and then handled in the discussion phase.
- As chairman of the discussion meeting, the moderator takes care of the people issues and prevents discussion from getting too personal and calls for a break to cool down the heated discussion.
- The outcome of the discussions is documented for the future reference.

Decision phase:

- At the end of the meeting a decision on the document under review has to be made by the participants, sometimes based on formal **exit criteria**.
- **Exit criteria** are the average number of critical and/or major defects found per page (for example no more than three critical/major defects per page).
- If the number of defects found per page is more than a certain level then the document must be reviewed again, after it has been reworked.

Formal Review – Follow Up

- In this step the moderator check to make sure that the author has taken action on all known defects.
- If it is decided that all participants will check the updated documents then the moderator takes care of the distribution and collects the feedback.
- It is the responsibility of the moderator to ensure that the information is correct and stored for future analysis.

Formal Review –

- In this step if the number of defects found per page exceeds the certain level then the document has to be reworked.
- Not every defect that is found leads to rework.
- It is the author's responsibility to judge whether the defect has to be fixed.
- If nothing can be done about an issue then at least it should be indicated that the author has considered the issue.

Roles & Tasks During Review Process

The moderator:

- Also known as review leader
- Performs entry check
- Follow-up on the rework
- Schedules the meeting
- Coaches other team
- Leads the possible discussion and stores the data that is collected

The author:

- Illuminate the unclear areas and understand the defects found
- Basic goal should be to learn as much as possible with regard to improving the quality of the document.

Roles & Tasks During Review Process

The scribe:

- Scribe is a separate person to do the logging of the defects found during the review.

The reviewers:

- Also known as checkers or inspectors
- Check any material for defects, mostly prior to the meeting
- The manager can also be involved in the review depending on his or her background.

The managers:

- Manager decides on the execution of reviews
- Allocates time in project schedules and determines whether review process objectives have been met

Types of Review - Walkthrough

Walkthrough:

- It is not a formal process
- It is led by the authors
- Author guide the participants through the document according to his or her thought process to achieve a common understanding and to gather feedback.
- Useful for the people if they are not from the software discipline, who are not used to
- Is especially useful for higher level documents like requirement specification, etc.

The goals of a walkthrough:

- To present the documents both within and outside the software discipline in order to gather the information regarding the topic under documentation.
- To explain or do the knowledge transfer and evaluate the contents of the document
- To achieve a common understanding and to gather feedback.
- To examine and discuss the validity of the proposed solutions

Types of Review – Technical/Peer

Technical Review:

- It is less formal review
- It is led by the trained moderator but can also be led by a technical expert
- It is often performed as a peer review without management participation
- Defects are found by the experts (such as architects, designers, key users) who focus on the content of the document.
- In practice, technical reviews vary from quite informal to very formal

The goals of the Technical Review are:

- To ensure that at an early stage the technical concepts are used correctly
- To assess the value of technical concepts and alternatives in the product
- To have consistency in the use and representation of technical concepts
- To inform participants about the technical content of the document

Types of Review – Inspection

Inspection:

- It is the most formal review type
- It is led by the trained moderators
- During inspection the documents are prepared and checked thoroughly by the reviewers before the meeting
- It involves peers to examine the product
- A separate preparation is carried out during which the product is examined and the defects are found
- The defects found are documented in a logging list or issue log
- A formal follow-up is carried out by the moderator applying exit criteria

The goals of inspection are:

- It helps the author to improve the quality of the document under inspection
- It removes defects efficiently and as early as possible
- It improves product quality
- It creates common understanding by exchanging information
- It learns from defects found and prevent the occurrence of similar defects

Build Release Process

What is Build in testing?

- After developing the software module, developers convert the source codes into a standalone form or an executable code.
- Then the development team hands over the build to the testing team to perform testing.
- Build is in the testing phase; it may have already undergone testing or not. Software testing team checks this build.
- If it consists of multiple bugs and if it does not meet the requirements, then the software testing team rejects that build.
- Builds occur prior to the release, and they are generated more frequently..

What is Release in testing?

- The release is the final application after completing development and testing.
- After testing the build, the testing team certifies that software and delivers it to the customer.
- It is possible for one release to have several builds.
- Therefore, it is the software delivered to the customer after completing the development and testing phases.
- Moreover, the release is based on builds, and it can have several builds.

- The **main difference** between Build and Release in Software Testing is that **Build is a version of a software the development team hands over to the testing team for testing purposes while Release is a software the testing team hands over to the customer**.



BUILD IN SOFTWARE BUILD IN SOFTWARE TESTING

VERSUS

RELEASE IN SOFTWARE TESTING

Is a stand-alone artifact generated after converting the source code to an executable code that can be run on a computer

Still in testing or not yet tested

Development team hands over a build to the testing team

Build occurs frequently

Still in development

Developed over a build

Build occurs frequently

It is the distribution of the final version of an application

No longer requires testing

Testing team offers the release to their customers

Release occurs occasionally

Distribution of the final version of an application

Requires testing

Offers the release to their customers

Release occurs occasionally

Standalone Application

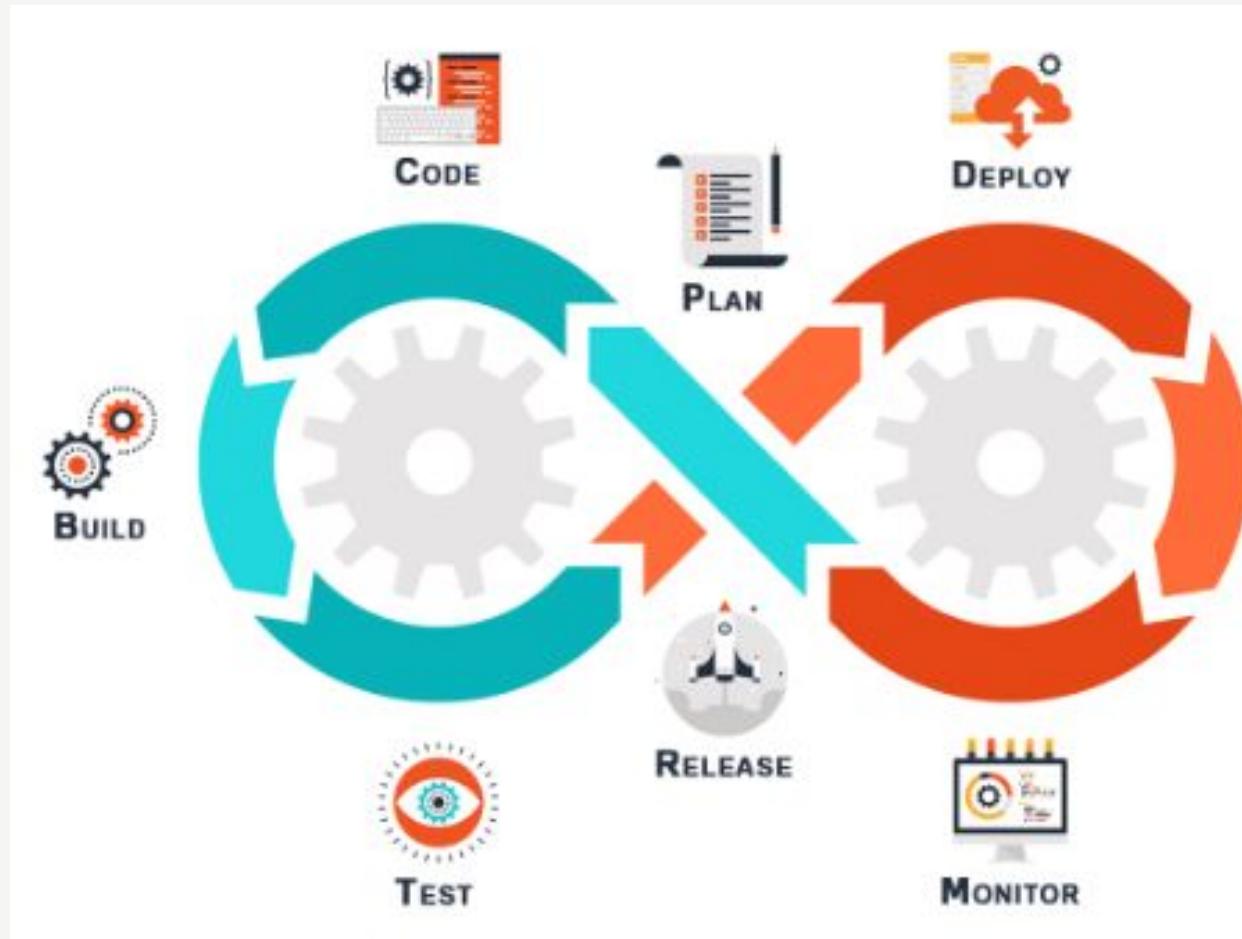
Software Build :

- After developing the software module, developers convert the source codes into a standalone form or an executable code.
- Then the development team hands over the build to the testing team to perform testing.
- Build is in the testing phase; it may have already undergone testing or not. Software testing team checks this build.
- If it consists of multiple bugs and if it does not meet the requirements, then the software testing team rejects that build.
- Builds occur prior to the release, and they are generated more frequently

Standalone Application

Software Release:

- The release is the final application after completing development and testing.
- After testing the build, the testing team certifies that software and delivers it to the customer.
- It is possible for one release to have several builds.
- Therefore, it is the software delivered to the customer after completing the development and testing phases.
- Moreover, the release is based on builds, and it can have several builds.



Defect Management and Tracking

Introduction

- Defect is the variance from a desired product attribute (it can be a wrong, missing or extra data).
- It can be of two types –
 - Defect from the **product or a variance from customer/user expectations.**
 - It is a flaw in the software system and has **no impact until it affects the user/customer and operational system.**
- With the knowledge of testing so far gained, you can now be able to categorize the defects you have found.
- Defects can be categorized into different types basing on the core issues they address.
- Some defects address security or database issues while others may refer to functionality or UI issues.

Types of Defect

Data Quality/Database Defects: Deals with improper handling of data in the database.

Examples:

- Values not deleted/inserted into the database properly
- Improper/wrong/null values inserted in place of the actual values

Critical Functionality Defects: The occurrence of these bugs hampers the crucial functionality of the application. Examples: - Exceptions

Functionality Defects: These defects affect the functionality of the application.

Examples:

- All JavaScript errors
- Buttons like Save, Delete, Cancel not performing their intended functions
- A missing functionality (or) a feature not functioning the way it is intended to
- Continuous execution of loops

Types of Defect

Security Defects: Application security defects generally involve improper handling of data sent from the user to the application. These defects are the most severe and given highest priority for a fix.

Examples:

- Authentication: Accepting an invalid username/password
- Authorization: Accessibility to pages though permission not given

User Interface Defects: As the name suggests, the bugs deal with problems related to UI are usually considered less severe.

Examples:

- Improper error/warning/UI messages
- Spelling mistakes
- Alignment problems

Bug(Defect) Life Cycle

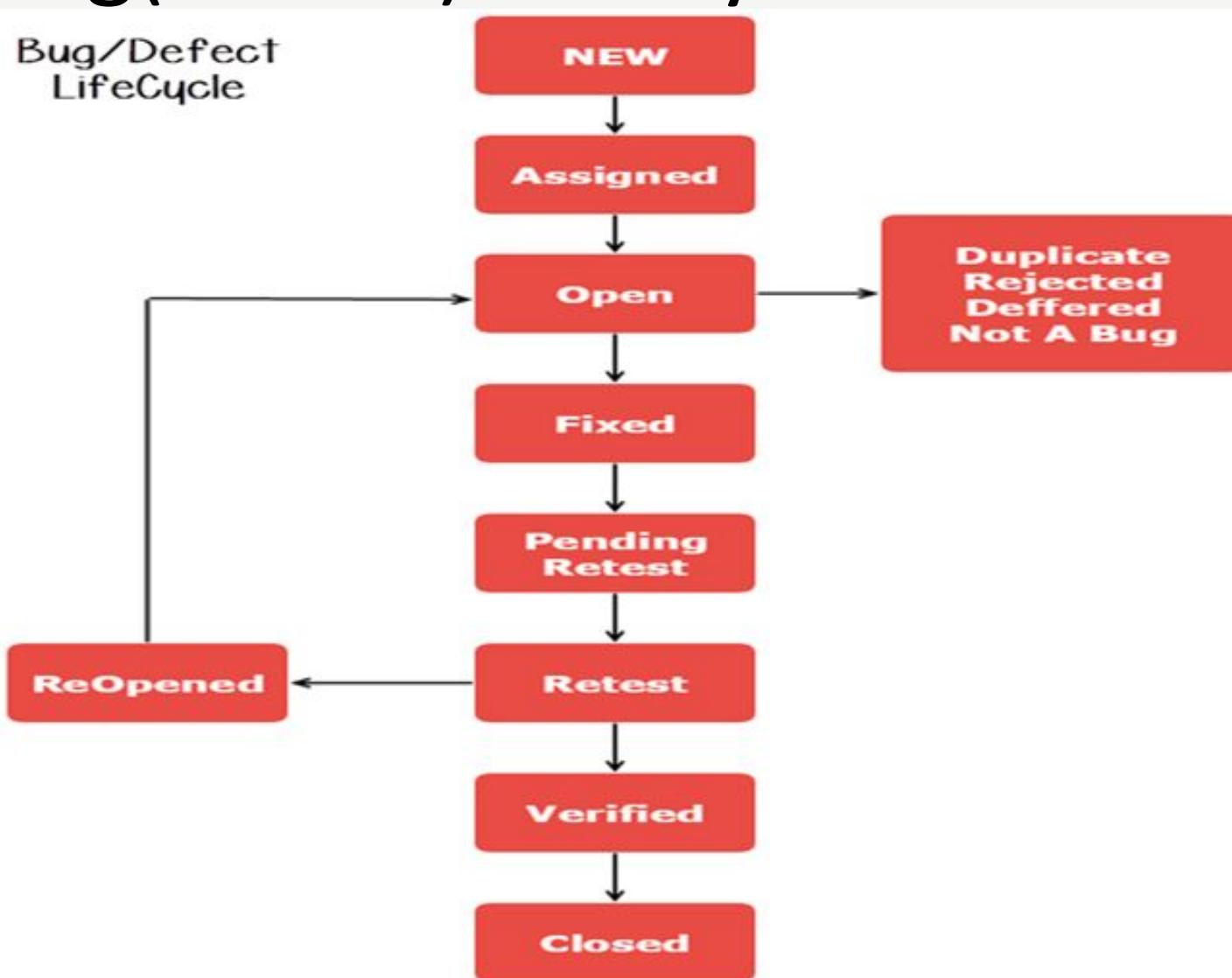
“A computer bug is an error, flaw, mistake, failure, or fault in a computer program that prevents it from working correctly or produces an incorrect result. Bugs arise from mistakes and errors, made by people, in either a program’s source code or its design.”

The duration or time span between the first time defects is found and the time that it is closed successfully, rejected, postponed or deferred is called as ‘Defect Life Cycle’.

When a bug is discovered, it goes through several states and eventually reaches one of the terminal states, where it becomes inactive and closed.

The process by which the defect moves through the life cycle is depicted next slide.

Bug(Defect) Life Cycle



Bug(Defect) Life Cycle

- As you can see from above diagram, a defect's state can be divided into Open or Closed.

- When a bug reaches one of the Closed or Terminal states, its lifecycle ends. Each state has one or more valid states to move to.

- This is to ensure that all necessary steps are taken to resolve or investigate that defect. For example, a bug should not move from Submitted state to resolved state without having it open.

- In a typical scenario, as soon as a bug is identified, it is logged into the bug tracking system with status as Submitted. After ascertaining the validity of the defect, it is given the “Open” Status.

Defect Stages

- **New:** When a new defect is logged and posted for the first time. It is assigned a status as NEW.
- **Assigned:** Once the bug is posted by the tester, the lead of the tester approves the bug and assigns the bug to the developer team
- **Open:** The developer starts analyzing and works on the defect fix
- **Fixed:** When a developer makes a necessary code change and verifies the change, he or she can make bug status as “Fixed.”
- **Pending retest:** Once the defect is fixed the developer gives a particular code for retesting the code to the tester. Since the software testing remains pending from the testers end, the status assigned is “pending retest.”
- **Retest:** Tester does the retesting of the code at this stage to check whether the defect is fixed by the developer or not and changes the status to “Re-test.”

Defect Stages(Cont...)

- **Verified:** The tester re-tests the bug after it got fixed by the developer. If there is no bug detected in the software, then the bug is fixed and the status assigned is “verified.”
- **Reopen:** If the bug persists even after the developer has fixed the bug, the tester changes the status to “reopened”. Once again the bug goes through the life cycle.
- **Closed:** If the bug is no longer exists then tester assigns the status “Closed.”
- **Duplicate:** If the defect is repeated twice or the defect corresponds to the same concept of the bug, the status is changed to “duplicate.”
- **Rejected:** If the developer feels the defect is not a genuine defect then it changes the defect to “rejected.”
- **Deferred:** If the present bug is not of a prime priority and if it is expected to get fixed in the next release, then status “Deferred” is assigned to such bugs
- **Not a bug:** If it does not affect the functionality of the application then the status assigned to a bug is “Not a bug”.

Defect Report Fields

You should provide enough detail while reporting the bug keeping in mind the people who will use it – test lead, developer, project manager, other testers, new testers assigned etc.

This means that the report you will write should be concise, straight and clear.

Following are the details your report should contain:

- Bug Title
- Bug identifier (number, ID, etc.)
- The application name or identifier and version / Test Case Id or Description
- The function, module, feature, object, screen, etc. where the bug occurred
- Environment (OS, Browser and its version)
- Bug Type or Category/Severity/Priority
- Bug status (Open, Pending, Fixed, Closed, Re-Open)
- Test case name/number/identifier
- Bug description
- Steps to Reproduce
- Actual Result
- Tester Comments

Defect Report Fields

- Bug Category: Security, Database, Functionality (Critical/General), UI
- Bug Severity: Severity with which the bug affects the application – Very High, High, Medium, Low, Very Low
- Bug Priority: Recommended priority to be given for a fix of this bug – Po, P₁, P₂, P₃, P₄, P₅ (Po-Highest, P₅-Lowest)

WHAT DOES THE TESTER DO WHEN THE DEFECT IS FIXED?

- Once the reported defect is fixed, the tester needs to re-test to confirm the fix. This is usually done by executing the possible scenarios where the bug can occur. Once retesting is completed, the fix can be confirmed and the bug can be closed. This marks the end of the bug life cycle.

Formate :<https://github.com/TopsCode/Software-Testing>

Defect Severity

Severity is absolute and Customer-Focused. It is the extent to which the defect can affect the software. In other words it defines the impact that a given defect has on the system.

- **For example:** If an application or web page crashes when a remote link is clicked, in this case clicking the remote link by an user is rare but the impact of application crashing is severe. So the severity is high but priority is low.

Severity can be of following types:

- **Critical:** The defect that results in the termination of the complete system or one or more component of the system and causes extensive corruption of the data. The failed function is unusable and there is no acceptable alternative method to achieve the required results then the severity will be stated as critical.

Defect Severity(Cont...)

Severity can be of following types:

- **Major (High):** The defect that results in the termination of the complete system or one or more component of the system and causes extensive corruption of the data. The failed function is unusable but there exists an acceptable alternative method to achieve the required results then the severity will be stated as major.
- **Moderate (Medium):** The defect that does not result in the termination, but causes the system to produce incorrect, incomplete or inconsistent results then the severity will be stated as moderate.
- **Minor (Low):** The defect that does not result in the termination and does not damage the usability of the system and the desired results can be easily obtained by working around the defects then the severity is stated as minor.
- **Cosmetic:** The defect that is related to the enhancement of the system where the changes are related to the look and field of the application then the severity is stated as cosmetic.

Defect Priority

Priority is Relative and Business-Focused. Priority defines the order in which we should resolve a defect. Should we fix it now, or can it wait? This priority status is set by the tester to the developer mentioning the time frame to fix the defect. If high priority is mentioned then the developer has to fix it at the earliest. The priority status is set based on the customer requirements.

- **For example:** If the company name is misspelled in the home page of the website, then the priority is high and severity is low to fix it.

Priority can be of following types:

- **Low:** The defect is an irritant which should be repaired, but repair can be deferred until after more serious defect has been fixed.
- **Medium:** The defect should be resolved in the normal course of development activities. It can wait until a new build or version is created.
- **High:** The defect must be resolved as soon as possible because the defect is affecting the application or the product severely. The system cannot be used until the repair has been done.
- **Critical:** Extremely urgent, resolve immediately

Scenario of Defect Priority - Severity

- **High Priority & High Severity:** An error which occurs on the basic functionality of the application and will not allow the user to use the system.
(Eg. A site maintaining the student details, on saving record if it, doesn't allow to save the record then this is high priority and high severity bug.)
- **High Priority & Low Severity:** The spelling mistakes that happens on the cover page or heading or title of an application.
- **High Severity & Low Priority:** An error which occurs on the functionality of the application (for which there is no workaround) and will not allow the user to use the system but on click of link which is rarely used by the end user.
- **Low Priority and Low Severity:** Any cosmetic or spelling issues which is within a paragraph or in the report (Not on cover page, heading, title).

Agile Testing

What is not AGILE?

Conducting meetings

The team conducts frequent meetings for 10-15 minutes daily, and they think that conducting frequent meetings will be Agile. However, only the following meetings will not be Agile.

Requirements changing anytime

Requirements can be changed at any time, then it is not Agile. For example, a client wants to add some new features and want the changes to be updated at the same time, then this will not be Agile.

Unstructured development

Suppose you are not following any plan and you are working on Adhoc basis then it is not Agile wherein Adhoc testing, testers randomly test the application without following any documents and test design techniques.

No documentation

If the company does not make the documentation, then it is not Agile.

What is Agile?

The **Agile methodology** is a way to manage a project by breaking it up into several phases. It involves constant collaboration with stakeholders and continuous improvement at every stage. Once the **work** begins, teams cycle through a process of planning, executing, and evaluating.

- Agile is a philosophy, i.e., a set of values and principles to make a decision for developing software.
- Agile is based on the iterative-incremental model. In an incremental model, we create the system in increments, where each increment is developed and tested individually.

What are values?

Individuals and interactions, Over processes and tools

Suppose the team finds any issue in software then they search for another process or tool to resolve the issue. But, in Agile, it is preferable to interact with client, manager or team regarding issue and make sure that the issue gets resolved.

Working software, Over comprehensive documentation

Documentation is needed, but working software is much needed. Agile is not saying that documentation is not needed, but working software is much needed. For example, you have 20-page documents, but you do not have a single prototype of the software. In such a case, the client will not be happy because, in the end, the client needs a document.

Customer collaboration, Over contract negotiation

Contract negotiation is important as they make the budget of software, but customer collaboration is more important than over contract negotiation. For example, if you stuck with the requirements or process, then do not go for a contract which we have negotiated. You need to interact with the customer, gather their requirements.

Responding to change, over following a plan

In the waterfall model, everything is planned, i.e., at what time, each phase will be completed. Sometimes you need to implement the new requirements in the middle of the software, so you need to be versatile to make changes in the software.

Agile Principles

- **Customer satisfaction through early and continuous software delivery** – Customers are happier when they receive working software at regular intervals, rather than waiting extended periods of time between releases.
- **Accommodate changing requirements throughout the development process** – The ability to avoid delays when a requirement or feature request changes.
- **Frequent delivery of working software** – Scrum accommodates this principle since the team operates in software sprints or iterations that ensure regular delivery of working software.
- **Collaboration between the business stakeholders and developers throughout the project** – Better decisions are made when the business and technical team are aligned.
- **Support, trust, and motivate the people involved** – Motivated teams are more likely to deliver their best work than unhappy teams.
- **Enable face-to-face interactions** – Communication is more successful when development teams are co-located.

- **Working software is the primary measure of progress** – Delivering functional software to the customer is the ultimate factor that measures progress.
- **Agile processes to support a consistent development pace** – Teams establish a repeatable and maintainable speed at which they can deliver working software, and they repeat it with each release.
- **Attention to technical detail and design enhances agility** – The right skills and good design ensures the team can maintain the pace, constantly improve the product, and sustain change.
- **Simplicity** – Develop just enough to get the job done for right now.
- **Self-organizing teams encourage great architectures, requirements, and designs** – Skilled and motivated team members who have decision-making power, take ownership, communicate regularly with other team members, and share ideas that deliver quality products.
- **Regular reflections on how to become more effective** – Self-improvement, process improvement, advancing skills, and techniques help team members work more efficiently.

Scrum

Scrum: SCRUM is an agile development method which concentrates particularly on how to manage tasks within a team based development environment.

Basically, Scrum is derived from activity that occurs during rugby match. Scrum believes in empowering the development team and advocates working in small teams (say- 7 to 9 members).

It consists of **three roles and their responsibilities** are explained as follows:

- **Scrum Master:** Master is responsible for setting up the team, sprint meeting and removes obstacles to progress
- **Product owner:** The Product Owner creates product backlog, prioritizes the backlog and is responsible for the delivery of the functionality at each iteration
- **Scrum Team:** Team manages its own work and organizes the work to complete the sprint or cycle

Sprint: Sprint is a time-boxed period in which the scrum team needs to finish the set amount of work. Each sprint has a specified timeline, i.e., 2 weeks to 1 month. The scrum team agrees with this timeline during the sprint planning meeting.

Roles

- Product Owner
- Scrum Master
- Team

1

Artifacts

- Product backlog
- Sprint backlog
- Burn-down charts

2

Ceremonies

- Sprint Planning
- Sprint Review
- Sprint Retrospective
- Daily scrum meeting

3

Scrum Roles

There are three scrum roles:

Product Owner

There is a client who wants to develop his software, so he approaches to the company who can develop his software. What does the company do? The Company assigns a role, i.e., Product Owner. Product Owner is the person who communicates with the clients understands their requirements. Product Owner is the responsible person from the company for software development.

Scrum Master

During the sprint, Agile says that the team should meet together once daily. When the team is following scrum means that they are conducting meetings daily for 10 to 15 minutes. This meeting is known as a scrum meeting. Scrum Master is the person who handles the scrum meeting.

Scrum Team

The team comprises of persons who work on the project. It can be developers, testers or designers. When we talk about Agile or Scrum then we talk about the team, we do not talk about developers, or testers as an individual. Agile says that developers can work as a tester or testers can work as a developer when the need arises.

Artifacts of Scrum

The documentation and stuff which are prepared in scrum are known as Artifacts.

Product Backlog

Product Backlog is a collection of activities that need to be done within the project. When we want to develop software, then we need to perform the 'n' number of activities. For example, we need to develop the e-commerce website then we have to do the 'n' number of activities such we need to create the login page, payment system, cart system, etc. and these 'n' number of activities which are needed to develop the software is known as the product backlog.

Sprint Backlog

We know that in a scrum, we break the scrum into 'n' number of sprints and the objective of a sprint is to bring the small functionality of the software and ship it to the client for demo. In Product backlog, we have to do all the activities which are required to develop the software while in the sprint backlog, a small set of product backlog activities are performed within that sprint. The 'n' number of sprint backlogs is equal to the 1 product backlog.

Burndown Chart

Burndown chart is the outcome of the sprint, which shows the progress in a sprint. After each sprint, we need to examine the progress of each sprint. The burndown chart tells how you are working on the sprint. In the burndown chart, the graph starts from some time, i.e., where the activity gets started, and at the end of the sprint, the graph reaches to zero where the activity ends. It is generally an inclined line from top to bottom.

Scrum Ceremonies

Let's look at the following Scrum Ceremonies:

Sprint Planning

Scrum consists of a number of sprints which have a different set of modules used to deliver the software. Before starting the sprint planning, we have a meeting known as sprint planning, and in sprint planning, we discuss what we are going to do in a sprint. In sprint planning, product owner discusses about each feature of a product and estimates the effort involved by the development team.

Daily Scrum

In Scrum, meetings are conducted daily for 15 minutes by Scrum Master, where Scrum Master is the person who manages the meeting. Meeting consists of scrum master, developers, testers, designers, product owner, the client where product owner and client are optional.

Sprint Review

After the completion of each sprint, the meeting is conducted with a client in which a product is shown to the client for demo and team discuss the features they added in the project.

Sprint Backlog

Sprint Backlog is a set of activities that need to be performed within this sprint. Out of the Product Backlog, a set of activities are captured in a sprint backlog, and each activity of a sprint backlog is assigned to a specific person. The minimum time to complete the sprint is 4 days, but it can be stretched to 2-3 weeks.

Sprint

After Sprint Backlog, the team starts working on a sprint, and it can take around 1 to 3 weeks to complete the sprint. The completion of sprint varies from project to project. When sprint gets started, a daily meeting is conducted known as Daily Scrum, and the Scrum Master conducts this meeting. In Daily Scrum, a meeting is conducted daily, and the meeting can be stretched to 10-15 meetings. Meeting has a predefined format, i.e., a team member has to tell what he was doing yesterday, what he will do today, and what are the things which are hampering him to complete his work. It's the responsibility of the scrum master to resolve the issues faced by the team members.

Sprint delivery

When a sprint is completed, then the sprint is delivered to the client. The product is delivered to the client means that minimum set of product backlog known as sprint backlog is completed. The sprint delivery is done so that the client can view the product, it's not that we have developed something and client cannot view.

Sprint Review and Retrospective

Once the sprint delivery is over, there are two types of meetings held, i.e., sprint review and retrospective. The **sprint review** is a meeting in which team members sit together, and they give the demo to the client about the product that they have developed in this sprint.

- A **retrospective** is another meeting which is held between team members. In this meeting, they discuss what is right in this sprint and what went wrong in this sprint, such as the issues hampering their work.
- After sprint review, we go back to the Product Backlog and then sprint planning is done to select the sprint backlog, i.e., sprint2, in this way, this cycle goes on until and unless the whole product is developed and shipped to the client.

Scrum Board

- **Product Backlog:** Product Backlog is a set of activities that need to be done to develop the software.
- **Sprint Backlog:** Sprint Backlog is a backlog that has taken some of the activities from the product backlog which needs to be completed within this sprint.
- **Scrum Board:** Scrum Board is a board that shows the status of all the activities that need to be done within this sprint.

Scrum Board consists of four status:

Open

The 'Open' status means that the tasks which are available in 'Open' are not yet started.

In progress

The 'In progress' status means the developers completed their tasks.

Testing

The 'testing' means that the task is in a testing phase.

Closed

The 'closed' means the task has been completed.

The Agile: Scrum Framework at a glance

Inputs from Executives,
Team, Stakeholders,
Customers, Users



Product Owner



The Team

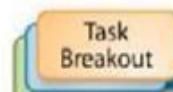


Product Backlog



Sprint Planning Meeting

Estimating Stories
(Story points)



Sprint Backlog



Sprint end date and team deliverable do not change



Duration: 15 Mins
➤ Status 24 hours last/next
➤ Blockers If any



Sprint Review



Finished Work

Final Demo to
PO/QA/DEV



Concept of Stories

- A user story is the smallest unit of work in an agile framework. It's an end goal, not a feature, expressed from the software user's perspective.
- A user story is an informal, general explanation of a software feature written from the perspective of the end user or customer.
- The purpose of a user story is to articulate how a piece of work will deliver a particular value back to the customer.
- User stories are a few sentences in simple language that outline the desired outcome. They don't go into detail.
- Requirements are added later, once agreed upon by the team.

Why Create Stories?

For development teams new to agile, user stories sometimes seem like an added step. Why not just break the big project (the epic) into a series of steps and get on with it? But stories give the team important context and associate tasks with the value those tasks bring

User stories serve a number of key benefits:

- **Stories keep the focus on the user.**
- **Stories enable collaboration.**
- **Stories drive creative solutions.**
- **Stories create momentum.**

Format of User Stories?

- Format of User Stories are.....

- "As a [persona], I [want to], [so that]."

- Breaking this down:

- "As a [persona]":** Who are we building this for? We're not just after a job title, we're after the persona of the person. Max. Our team should have a shared understanding of who Max is. We've hopefully interviewed plenty of Max's. We understand how that person works, how they think and what they feel. We have empathy for Max.

- "Wants to":** Here we're describing their intent — not the features they use. What is it they're actually trying to achieve? This statement should be implementation free — if you're describing any part of the UI and not what the user goal is you're missing the point.

- "So that":** how does their immediate desire to do something this fit into their bigger picture? What's the overall benefit they're trying to achieve? What is the big problem that needs solving?

Writing User Stories : Example

Agile User Story Writing Template

Use story no	As a <type of user/persona>	I want to <goal/ objective>	So that <benefit/ result/some reason>
1	Project team member	know all my tasks in advance	I can prepare and plan my time properly
2	Content manager	get a weekly report of content analytics	I can monitor the effectiveness of content writer
3	CEO	get a weekly report from all department heads on their team goals	I know whether my strategy is working
4	Middle-aged woman on a sabbatical	earn a certificate for digital marketing from coursera	I can restart my career
5	Software developer	reskill myself by attending the training program organized by my employer	I can stay employed

What is Kanban?

- Kanban is a very popular framework for development in the agile software development methodology.
- It provides a transparent way of visualizing the tasks and work capacity of a team.
- It mainly uses physical and digital boards to allow the team members to visualize the current state of the project they are working on.
- Kanban originated in Toyota in the 1940s.
- Kanban's meaning in Japanese is "billboards."
- The Kanban board has columns and story cards.
- The columns are nothing, but workflow states and cards are nothing but a demonstration of the actual task a team member is performing.

When to use Kanban?

- Kanban can be used in any domain, and it can be used very effectively in software development.
- Kanban project management helps in improving the efficiency of the team.
- It is a pull-based system. Tasks are being pulled as soon as an individual is free.
- Kanban should be used when you want to release your work at any time. It requires git branching, but it is doable.
- Kanban should be used when you want to change the priorities on the fly. For that, all you need to do is to put this story on the top of the to-do queue.
- It should be used when you want to visualize your work, and you want to see the progress of your tasks visually.

Kanban Cards



JIRA

(Defect tracking and Management Tool)



JIRA

Introduction

- JIRA is a tool developed by Australian Company Atlassian.
- It is used for bug tracking, issue tracking, and project management.
- The name "JIRA" is actually inherited from the Japanese word "Gojira" which means "Godzilla".
- The basic use of this tool is to track issue and bugs related to your software and Mobile apps.
- The JIRA dashboard consists of many useful functions and features which make handling of issues easy

Importance of JIRA

- Requirements and Test case management
- In Agile Methodology
- Project Management
- Software Development
- Product Management
- Task Management
- Bug Tracking

How to Use JIRA?

- **Step 1)** Open Jira software and navigate to the Jira Home icon
- **Step 2)** Select Create project option
- **Step 3)** Choose a template from the library
- **Step 4)** Set up the columns as per your need from Board settings
- **Step 5)** Create an issue
- **Step 6)** Invite your Team members and start working



Log in to your account

Enter email

Continue

OR



Continue with Google



Continue with Microsoft



Continue with Apple



Continue with Slack

Can't log in? • Sign up for an account

 ? CC Settings

Switch to [Show all products](#)

[!\[\]\(cb5ab46b82866be15d30c1210788e67e_img.jpg\)](#) [!\[\]\(2f5a35607f5f797f73160e0116caf311_img.jpg\)](#) [!\[\]\(c2848fba33b799e2fac1e541cb8c9f47_img.jpg\)](#) [!\[\]\(91c6da7af38ad63d57aab54d9b03b654_img.jpg\)](#) [!\[\]\(f3cc484338eb3e5ba48fdcbea7030236_img.jpg\)](#) [!\[\]\(62dac67878f4dc3050656f8b43e8ade5_img.jpg\)](#) [!\[\]\(67f03baab82a66c85824c4d8625a9dcd_img.jpg\)](#) [!\[\]\(851e3a7c9210d4d5f5bf6346948b9d81_img.jpg\)](#) [!\[\]\(bfbc622e737ed96162e1a3c7316334e3_img.jpg\)](#) [!\[\]\(d4646d6a9bbcfc55cb779e3846f7cee2_img.jpg\)](#)

Jira Software
rahulsanghavi

Jira Software
topstech

Confluence
[TRY](#)

Jira Service Management

Opsgenie

Account settings

Atlassian Support

Atlassian Community

Self-managed licensing

Administration

Frequent

[!\[\]\(b6040b399bc33a9b73f0cd658ebd9351_img.jpg\)](#)
DealMart1
Jira

[!\[\]\(0e53016280bcbf4ce9eaf3fd70317993_img.jpg\)](#)
E-mobile shoap
Jira

[!\[\]\(9add99edff973fadd9e37118c8118fa9_img.jpg\)](#)
Banking System
Jira

X

Project templates

Project templates

Software development

Service management

Work management

Marketing

Human resources

Finance

Design

Personal

Operations

Legal

Sales

PRODUCTS

Jira Software

Jira Service Management

Jira Work Management

Software development

Plan, track and release great software. Get up and running quickly with templates that suit the way your team works. Plus, integrations for DevOps teams that want to connect work across their entire toolchain.



Kanban

Visualize and advance your project forward using issues on a powerful board.



Scrum LAST CREATED

Sprint toward your project goals with a board, backlog, and roadmap.



Bug tracking

Manage a list of development tasks and bugs.

[← Back to project types](#)

Add project details

You can change these details anytime in your project settings.

Name *

E-Commerce

Access Anyone with access to rahulsanghavi can access and administer this project. Upgrade your plan to customize project permissions.

Key 1*

COM

Connect repositories, documents, and more

Sync your team's work from other tools with this project for better visibility, access, and automation.

Template

Change template



Scrum

Sprint toward your project goals with a board, backlog, and roadmap.

Type

Change type



Team-managed

Control your own working processes and practices in a self-contained space.

Cancel

Create project

Jira Software Your work Projects Filters Dashboards People Apps Create

Search

E-Commerce Software project

Roadmap

Backlog

Board

Code

Project pages

Add shortcut

Project settings

You're in a team-managed project

Learn more

Projects / E-Commerce

COM board

Search CC User

GROUP BY None

TO DO IN PROGRESS DONE



You haven't started a sprint

You can't do anything on your board because you haven't started a sprint yet. Go to the backlog to plan and start a sprint.

Go to Backlog

Quickstart

Jira Software Your work Projects Filters Dashboards People Apps Create Search

E-Commerce Software project

Roadmap Backlog Board Code Project pages Add shortcut Project settings

Backlog (0 issues) Your backlog is empty.

+ Create issue

Issues without epic

for a new e-commerce website to launch, the highest business value will be when a new user is able to buy an item from website.

What will the Epic be called?

0 0 0 Create sprint

Quickstart

Jira Software Your work Projects Filters Dashboards People Apps Create Q Search ⚙️ 🌐 🔍

E-Commerce Software project Roadmap Backlog Board Code Project pages Add shortcut Project settings

Create issue Import issues Configure fields Insights Create sprint

Project*: E-Commerce (COM)

Issue Type*: Story

Some issue types are unavailable due to incompatible field configuration and/or workflow associations.

Summary*: First time visitor can make the payment as a guest user without having to register

Description:
Style: B I U A \approx A \approx \approx \approx \approx \approx \approx \approx
"As a first time visitor to the e-commerce website
-I want to be able to buy a listed product
-so that I can use the product I buy"

Assignee: Automatic

Assign to me

Labels

Sprint

Jira Software sprint field

Story point estimate

Measurement of complexity and/or size of a requirement.

Reporter*: Chintan Chovatiya

Start typing to get a list of possible matches.

Attachment: Drop files to attach, or browse.

Linked Issues: blocks

Issue

Flagged

Impediment

Allows to flag issues with impediments.

Create another **Create** Cancel Quickstart

Jira Software Your work Projects Filters Dashboards People Apps Create Search

E-Commerce Software project

Roadmap

Backlog

Board

Code

Project pages

Add shortcut

Project settings

You're in a team-managed project

Learn more

Projects / E-Commerce

Backlog

Epic

Issues without epic

for a new e-commerce website to launch, the highest business value will be when a new user is able to buy an item from website.

+ Create Epic

COM Sprint 1 Add dates (1 issue)

COM-2 First time visitor can make the payment as a guest user without having to register

+ Create issue

Backlog (0 issues)

Your backlog is empty.

+ Create issue

Start sprint

Insights

Jira Software Your work Projects Filters Dashboards People Apps Create Search

E-Commerce Software project

Roadmap Backlog

Board

Code Project pages Add shortcut Project settings

You're in a team-managed project Learn more

Projects / E-Commerce

COM Sprint 1

9 days remaining Complete sprint ...

GROUP BY None Insights

TO DO 1 ISSUE

IN PROGRESS

DONE +

First time visitor can make the payment as a guest user without having to register

COM-2

Quickstart

Module - 4

Defect Tracking

Tools

Or

Bug Tracking Tools

Bug Tracking Tools

1. Bugzilla:



2. JIRA:



5. Redmine:



3. Mantis:



4. Trac:





Bug Tracking Tools

6. HP ALM/Quality Center:



7. FogBugz:



10. Zoho bug tracker:



8. IBM Rational ClearQuest:



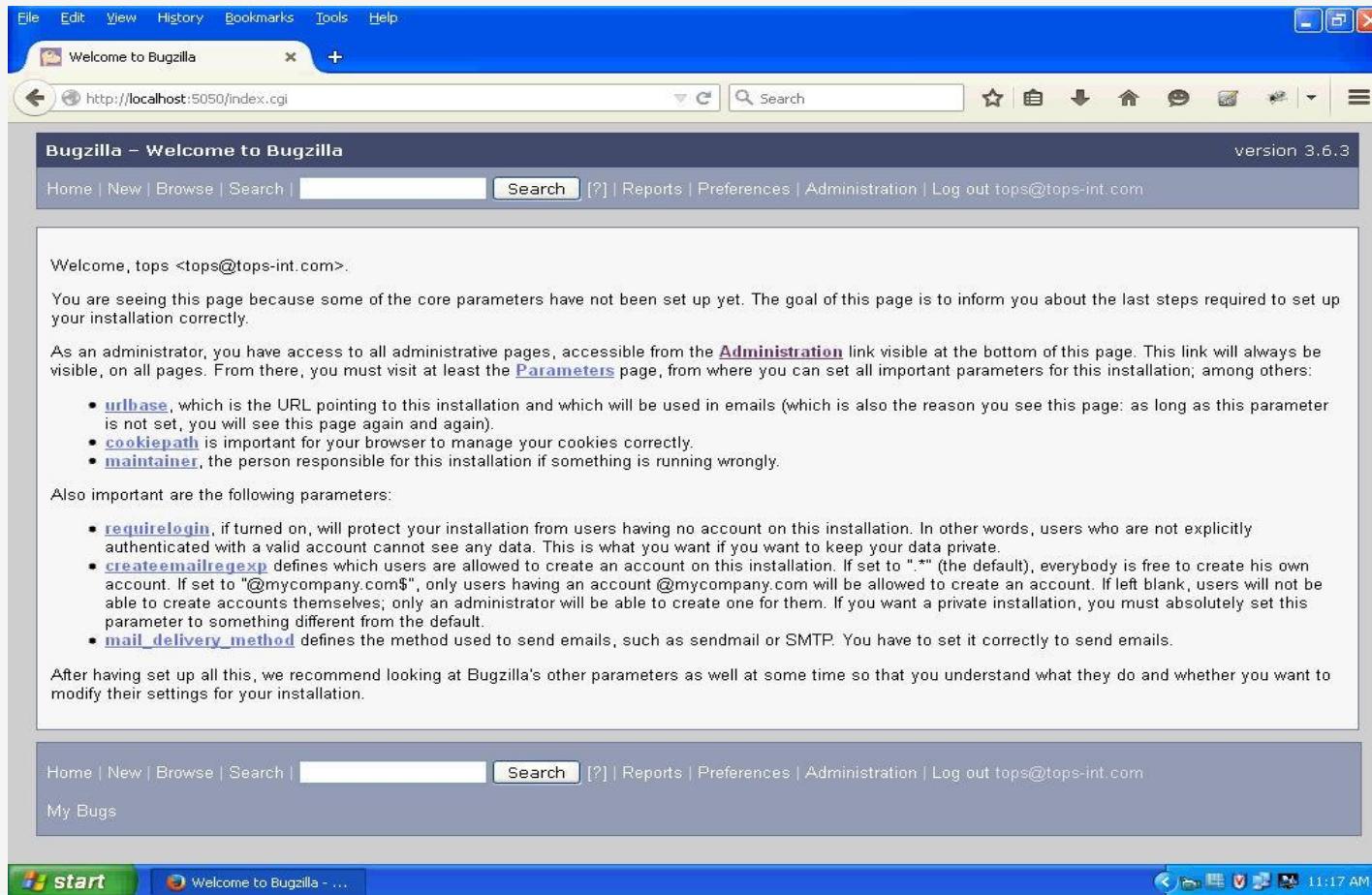
9. Lighthouse:



Bugzilla

- Bugzilla is an open-source issue/bug tracking system that allows developers effectively to keep track of outstanding problems with their product. It is written in Perl and uses MYSQL database.
- Bugzilla is a defect tracking tool, however it can be used as a test management tool as such it can be easily linked with other test case management tools like Quality Center, Testlink etc.
- This open bug-tracker enables users to stay connected with their clients or employees, to communicate about problems effectively throughout the data-management chain.
- Key features of Bugzilla includes
 - Advanced search capabilities
 - E-mail Notifications
 - Modify/file Bugs by e-mail
 - Time tracking
 - Strong security
 - Customization
 - Localization

Bugzilla : Welcome Screen



The screenshot shows a web browser window titled "Welcome to Bugzilla". The address bar displays "http://localhost:5050/index.cgi". The page content is the "Welcome to Bugzilla" page for version 3.6.3. It includes a message for the administrator about core parameters, instructions for setting up the installation, and a list of important parameters like `urlbase`, `cookiepath`, and `maintainer`. It also lists other parameters such as `requirelogin`, `createemailregexp`, and `mail_delivery_method`. A note at the bottom encourages users to explore other parameters. The browser's status bar shows the URL "Welcome to Bugzilla - ..." and the system time "11:17 AM".

Welcome, tops <tops@tops-int.com>.

You are seeing this page because some of the core parameters have not been set up yet. The goal of this page is to inform you about the last steps required to set up your installation correctly.

As an administrator, you have access to all administrative pages, accessible from the [Administration](#) link visible at the bottom of this page. This link will always be visible, on all pages. From there, you must visit at least the [Parameters](#) page, from where you can set all important parameters for this installation, among others:

- [urlbase](#), which is the URL pointing to this installation and which will be used in emails (which is also the reason you see this page: as long as this parameter is not set, you will see this page again and again).
- [cookiepath](#) is important for your browser to manage your cookies correctly.
- [maintainer](#), the person responsible for this installation if something is running wrongly.

Also important are the following parameters:

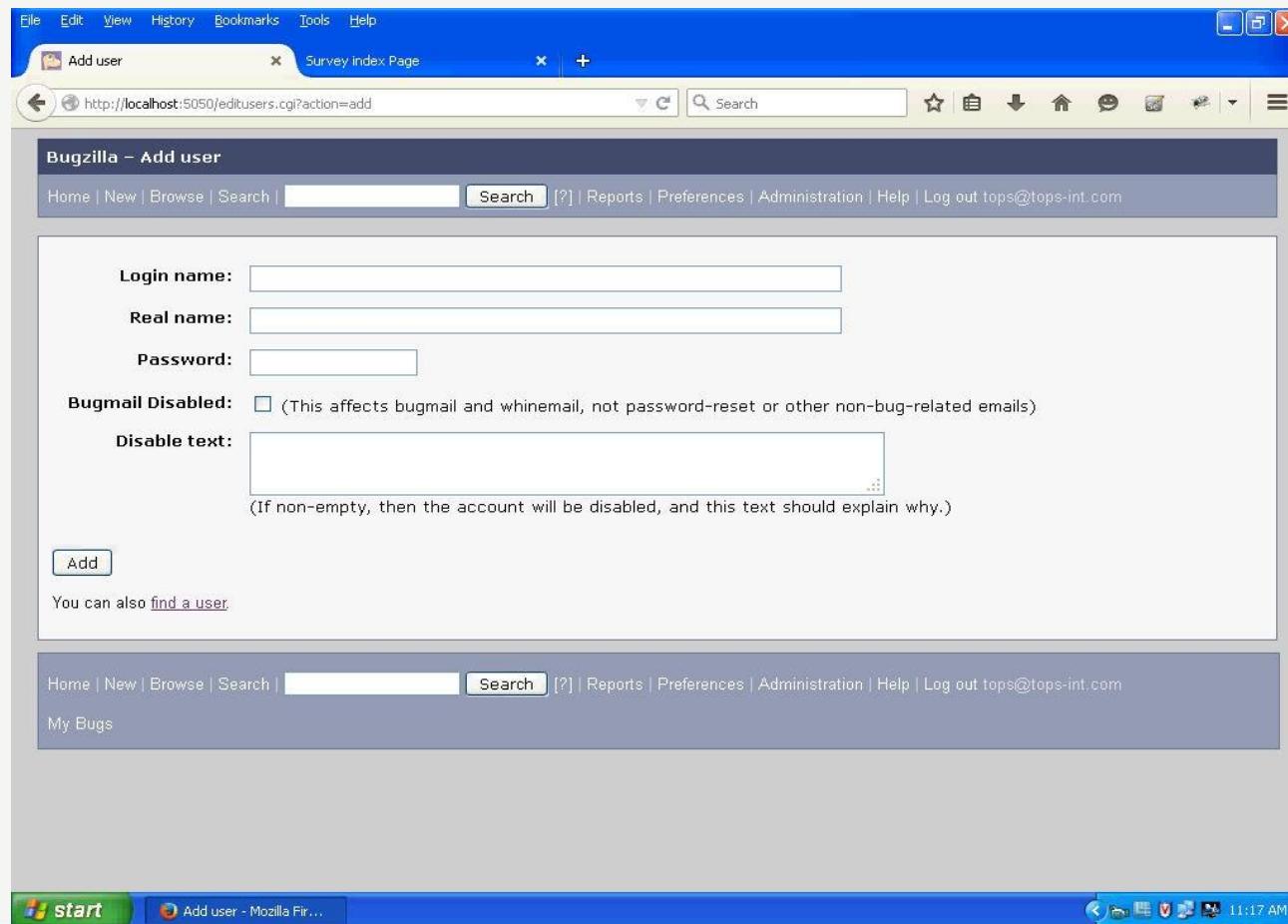
- [requirelogin](#), if turned on, will protect your installation from users having no account on this installation. In other words, users who are not explicitly authenticated with a valid account cannot see any data. This is what you want if you want to keep your data private.
- [createemailregexp](#) defines which users are allowed to create an account on this installation. If set to "*" (the default), everybody is free to create his own account. If set to "@mycompany.com\$", only users having an account @mycompany.com will be allowed to create an account. If left blank, users will not be able to create accounts themselves; only an administrator will be able to create one for them. If you want a private installation, you must absolutely set this parameter to something different from the default.
- [mail_delivery_method](#) defines the method used to send emails, such as sendmail or SMTP. You have to set it correctly to send emails.

After having set up all this, we recommend looking at Bugzilla's other parameters as well at some time so that you understand what they do and whether you want to modify their settings for your installation.

Home | New | Browse | Search | [?] | Reports | Preferences | Administration | Log out tops@tops-int.com

My Bugs

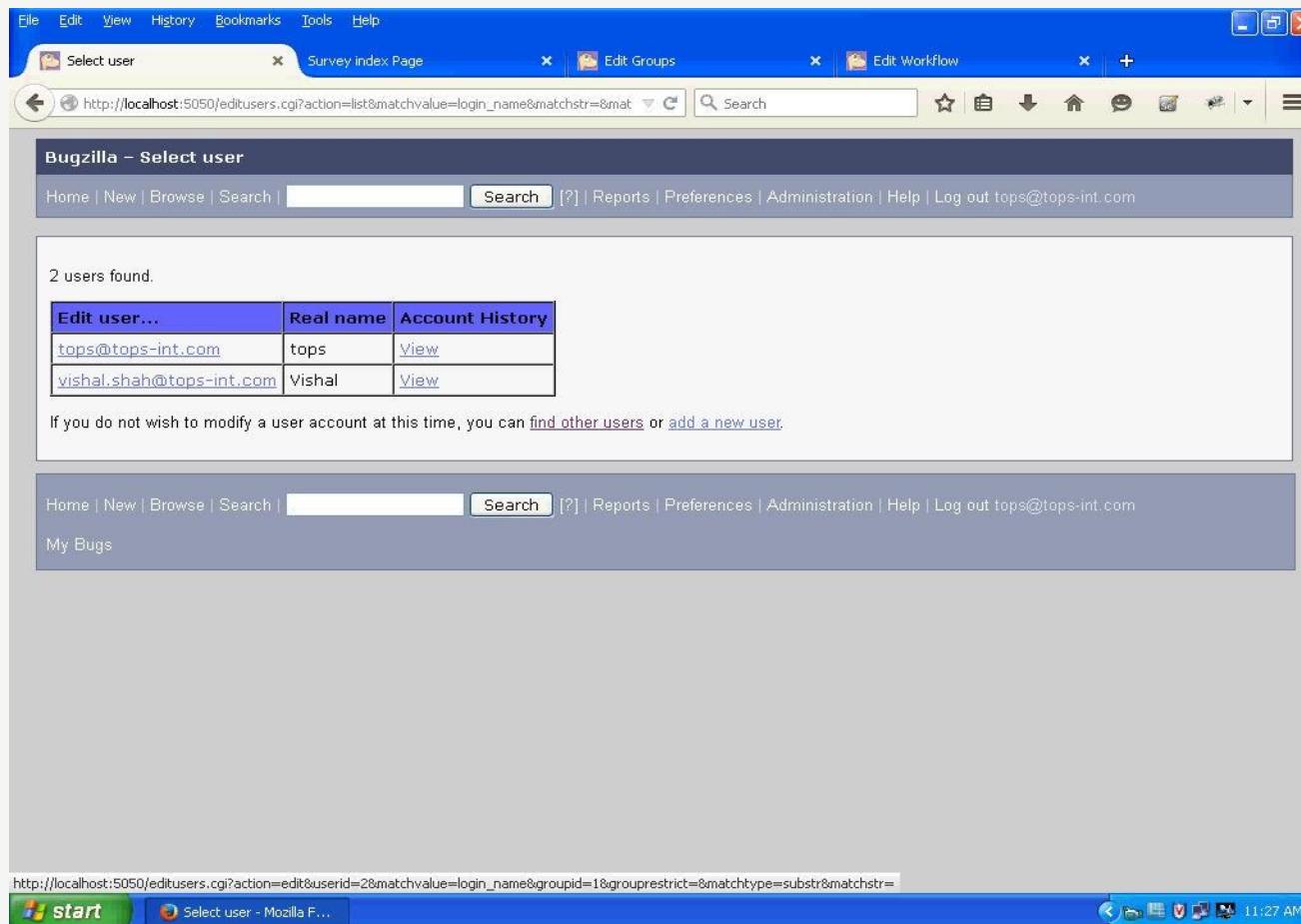
Bugzilla : Add User



The screenshot shows a Mozilla Firefox browser window with the following details:

- Title Bar:** File Edit View History Bookmarks Tools Help
Add user Survey index Page
- Address Bar:** http://localhost:5050/editusers.cgi?action=add
- Content Area:**
 - Bugzilla – Add user**
 - Home | New | Browse | Search | Search | [?] | Reports | Preferences | Administration | Help | Log out tops@tops-int.com
 - Login name:**
 - Real name:**
 - Password:**
 - Bugmail Disabled:** (This affects bugmail and whinemail, not password-reset or other non-bug-related emails)
 - Disable text:**
(If non-empty, then the account will be disabled, and this text should explain why.)
 - Add** button
 - You can also [find a user](#).
- Bottom Navigation Bar:** Home | New | Browse | Search | Search | [?] | Reports | Preferences | Administration | Help | Log out tops@tops-int.com
- Bottom Status Bar:** My Bugs
- Taskbar:** start | Add user - Mozilla Fir... | 11:17 AM

Bugzilla : Search User



The screenshot shows a Mozilla Firefox browser window with four tabs open:

- Select user
- Survey index Page
- Edit Groups
- Edit Workflow

The active tab is "Select user". The address bar shows the URL: `http://localhost:5050/editusers.cgi?action=list&matchvalue=login_name&matchstr=&mat`. The page title is "Bugzilla - Select user".

The main content area displays a table of users found:

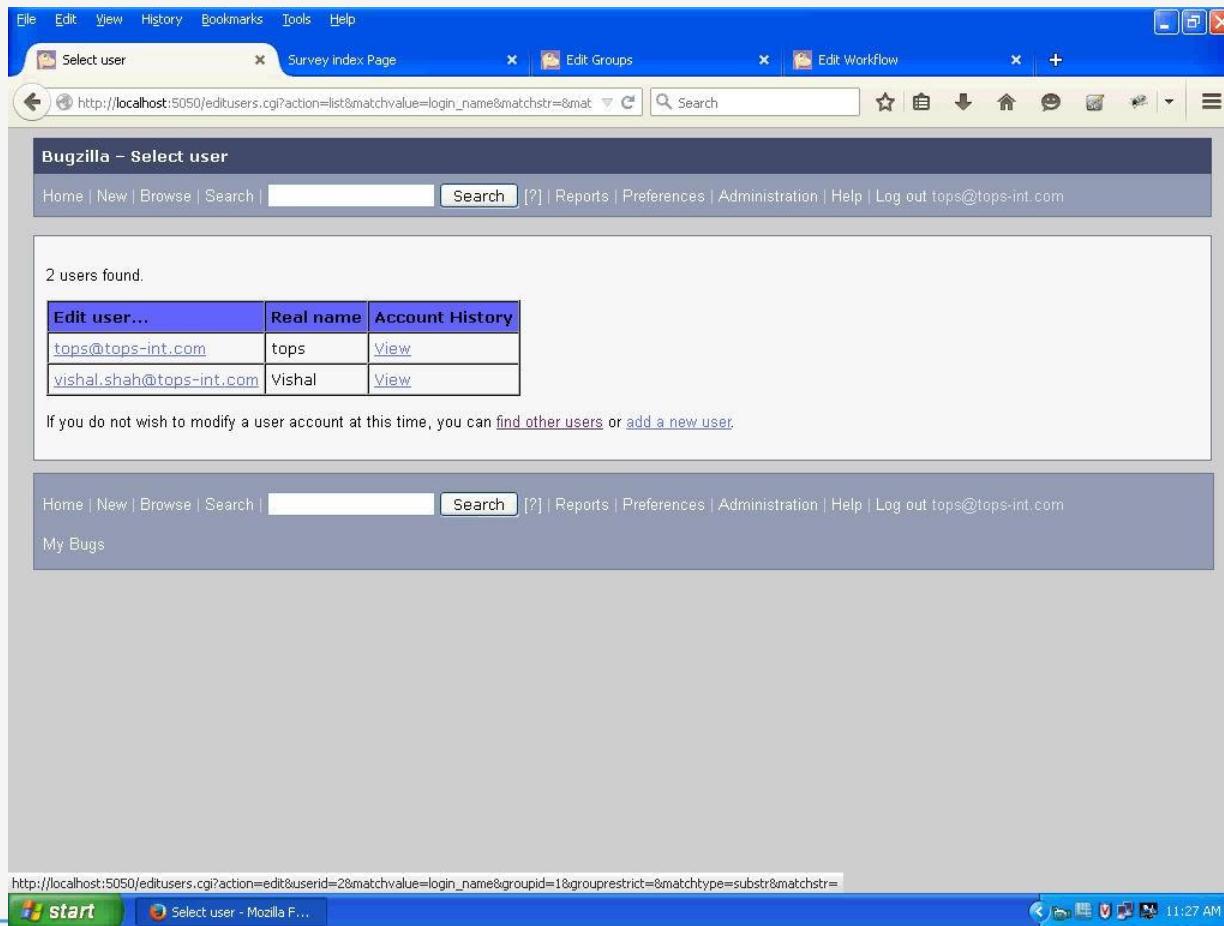
Edit user...	Real name	Account History
tops@tops-int.com	tops	View
vishal.shah@tops-int.com	Vishal	View

A message below the table states: "If you do not wish to modify a user account at this time, you can [find other users](#) or [add a new user](#)".

The bottom of the page includes a navigation bar with links: Home | New | Browse | Search | [?] | Reports | Preferences | Administration | Help | Log out tops@tops-int.com

The status bar at the bottom of the browser window shows the full URL: `http://localhost:5050/editusers.cgi?action=edit&userid=2&matchvalue=login_name&groupid=1&grouprestrict=0&matchtype=substr&matchstr=`.

Select User



The screenshot shows a Mozilla Firefox browser window with three tabs open:

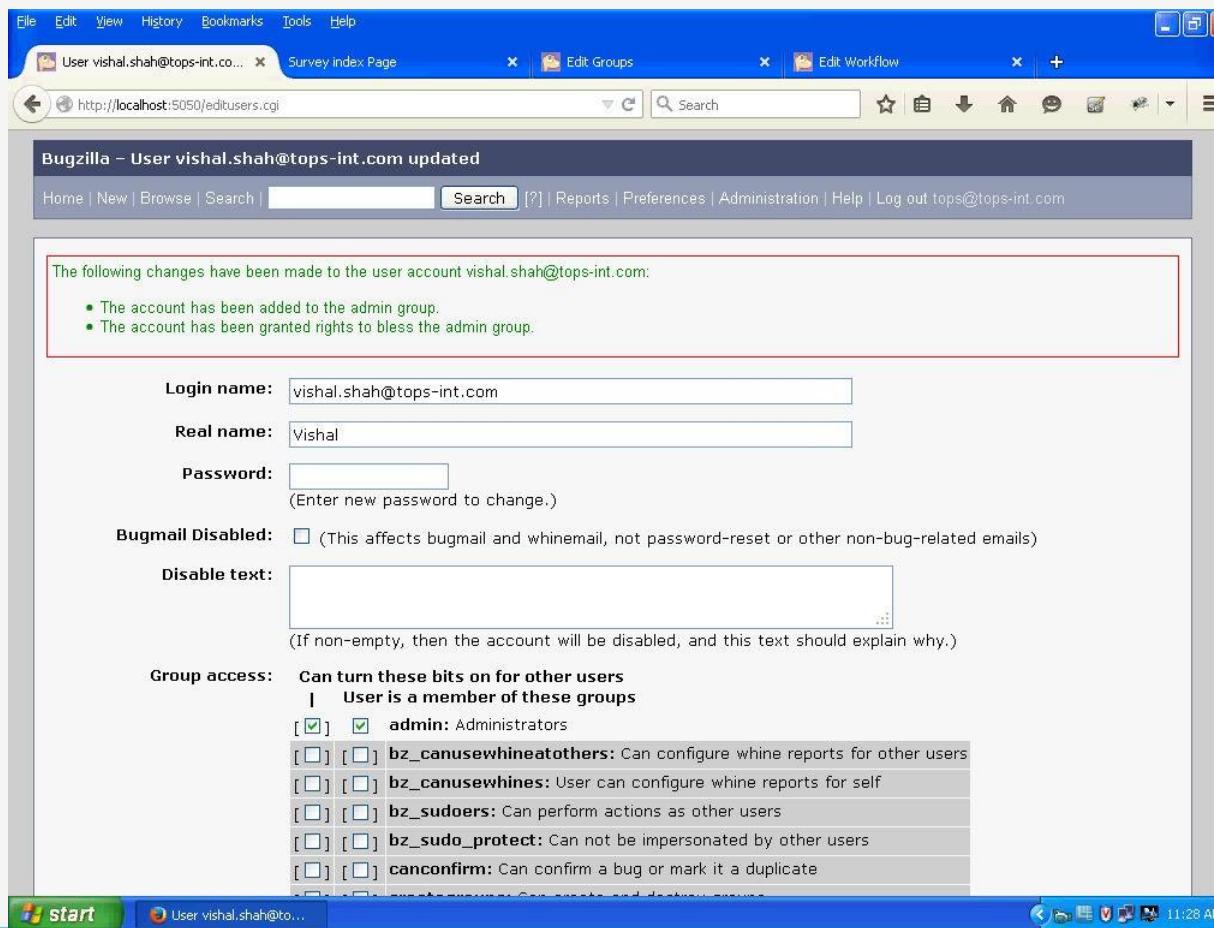
- Select user**: The active tab, displaying the Bugzilla "Select user" page. The URL is http://localhost:5050/editusers.cgi?action=list&matchvalue=login_name&matchstr=&mat. The page shows a table with two users found:

Edit user...	Real name	Account History
tops@tops-int.com	tops	View
vishal.shah@tops-int.com	Vishal	View

If you do not wish to modify a user account at this time, you can [find other users](#) or [add a new user](#).
- Survey index Page**
- Edit Groups**

The browser's address bar also contains the URL http://localhost:5050/editusers.cgi?action=edit&userid=2&matchvalue=login_name&groupid=1&grouprestrict=&matchtype=substr&matchstr=.

Edit User



The screenshot shows a web browser window with multiple tabs open. The active tab is titled "User vishal.shah@tops-int.co... Survey index Page". The URL in the address bar is "http://localhost:5050/editusers.cgi". The page content is as follows:

Bugzilla – User vishal.shah@tops-int.com updated

The following changes have been made to the user account vishal.shah@tops-int.com:

- The account has been added to the admin group.
- The account has been granted rights to bless the admin group.

Login name: vishal.shah@tops-int.com

Real name: Vishal

Password: (Enter new password to change.)

Bugmail Disabled: (This affects bugmail and whinemail, not password-reset or other non-bug-related emails)

Disable text:
(If non-empty, then the account will be disabled, and this text should explain why.)

Group access: Can turn these bits on for other users
| User is a member of these groups

[] [] **admin:** Administrators

[] [] **bz_canusewhineatothers:** Can configure whine reports for other users

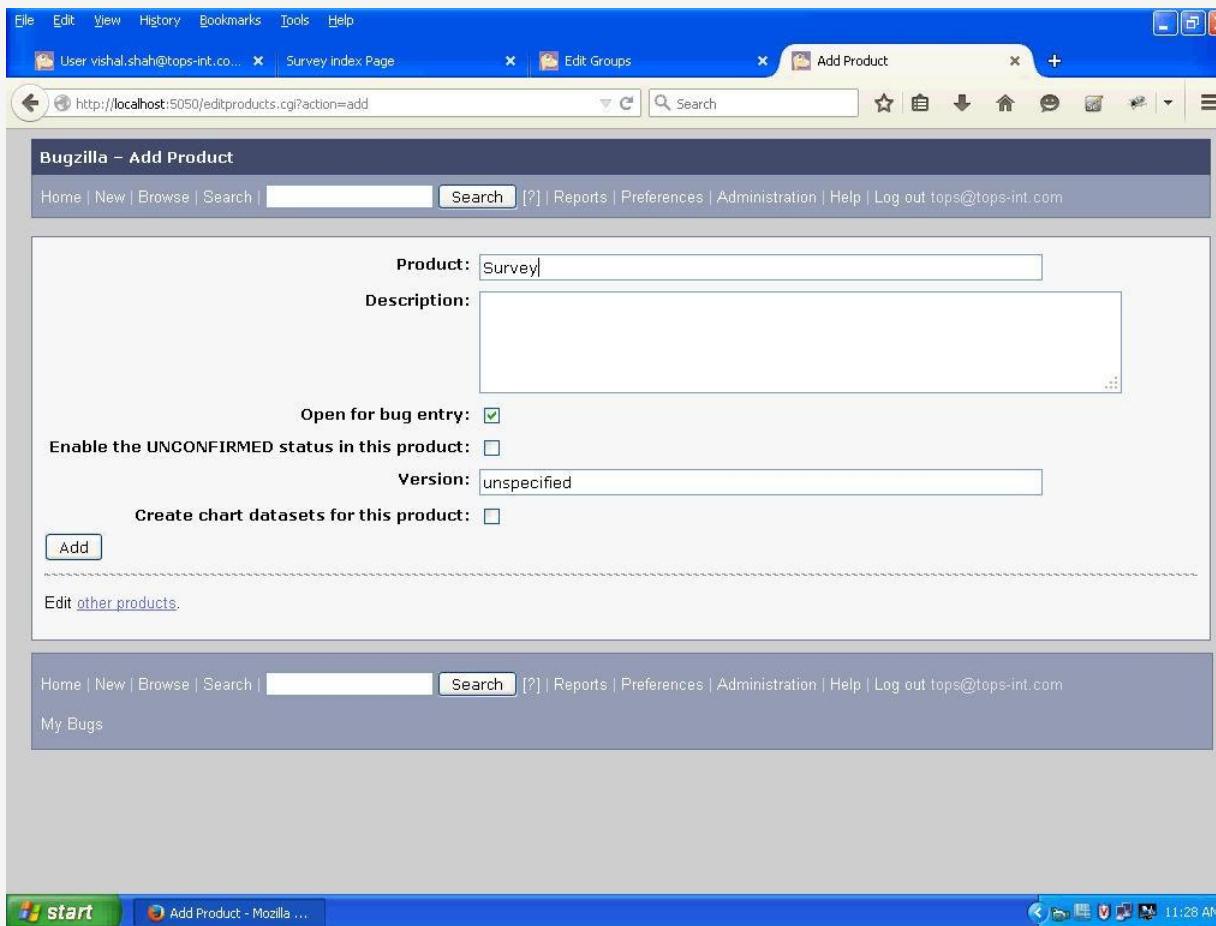
[] [] **bz_canusewhines:** User can configure whine reports for self

[] [] **bz_sudoers:** Can perform actions as other users

[] [] **bz_sudo_protect:** Can not be impersonated by other users

[] [] **canconfirm:** Can confirm a bug or mark it a duplicate

Add product



File Edit View History Bookmarks Tools Help

User vishal.shah@tops-int.co... Survey index Page Edit Groups Add Product

http://localhost:5050/editproducts.cgi?action=add

Bugzilla – Add Product

Home | New | Browse | Search | Search | [?] | Reports | Preferences | Administration | Help | Log out tops@tops-int.com

Product: Survey

Description:

Open for bug entry:

Enable the UNCONFIRMED status in this product:

Version: unspecified

Create chart datasets for this product:

Add

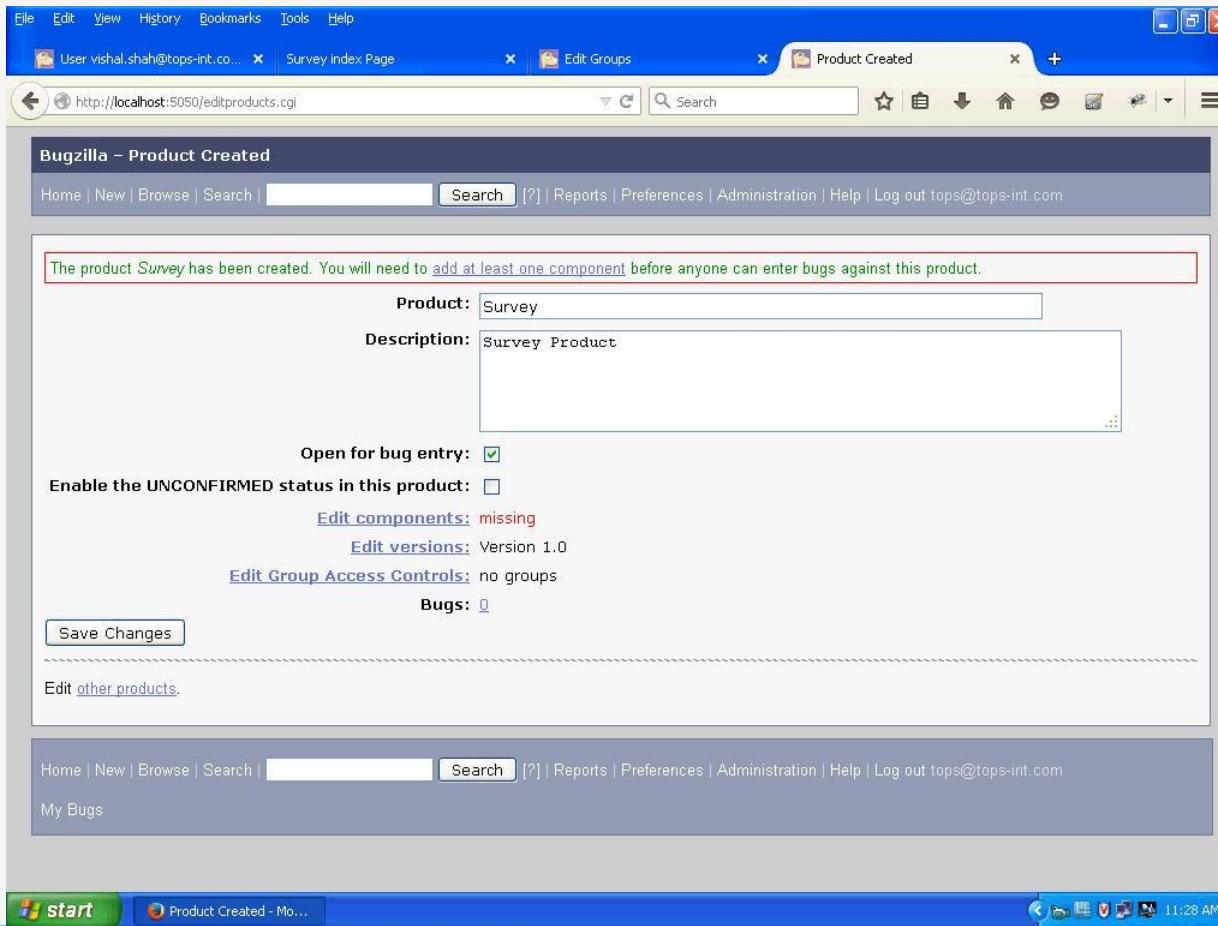
Edit other products.

Home | New | Browse | Search | Search | [?] | Reports | Preferences | Administration | Help | Log out tops@tops-int.com

My Bugs

start Add Product - Mozilla ... 11:28 AM

Product Added

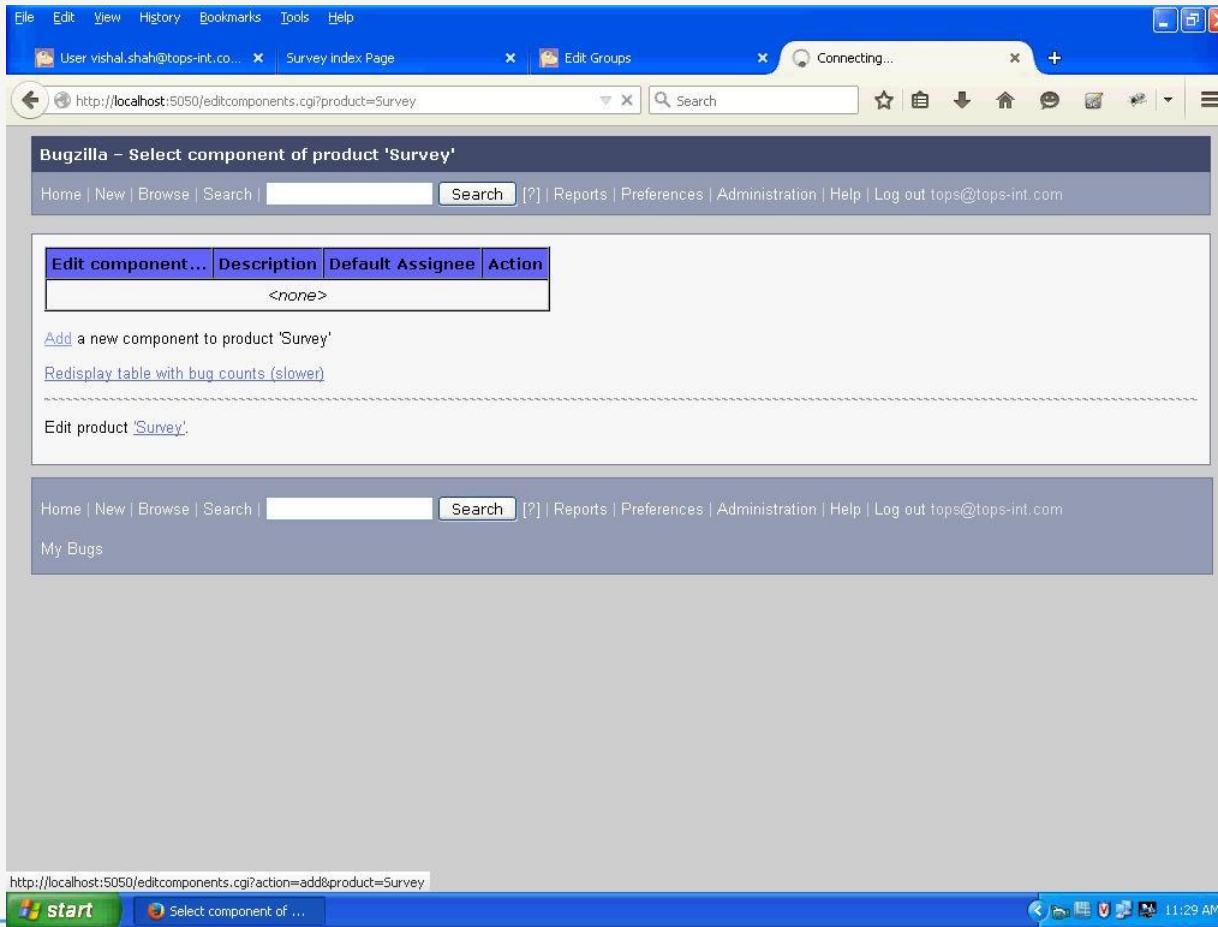


The screenshot shows a web browser window with three tabs: "User vishal.shah@tops-int.co...", "Survey index Page", and "Edit Groups". The active tab is "Product Created" at the URL <http://localhost:5050/editproducts.cgi>. The page title is "Bugzilla – Product Created". The main content area displays the following information:

Product: Survey
Description: Survey Product
Open for bug entry:
Enable the UNCONFIRMED status in this product:
[Edit components](#): missing
[Edit versions](#): Version 1.0
[Edit Group Access Controls](#): no groups
Bugs: 0
[Save Changes](#)

Below this, there is a link to "Edit other products". At the bottom of the page, there is a footer bar with links to "Home", "New", "Browse", "Search", "Search", "?", "Reports", "Preferences", "Administration", "Help", and "Log out tops@tops-int.com". The footer also shows "My Bugs". The bottom of the screen shows the Windows taskbar with the "start" button, the current window title "Product Created - Mo...", and the system clock showing 11:28 AM.

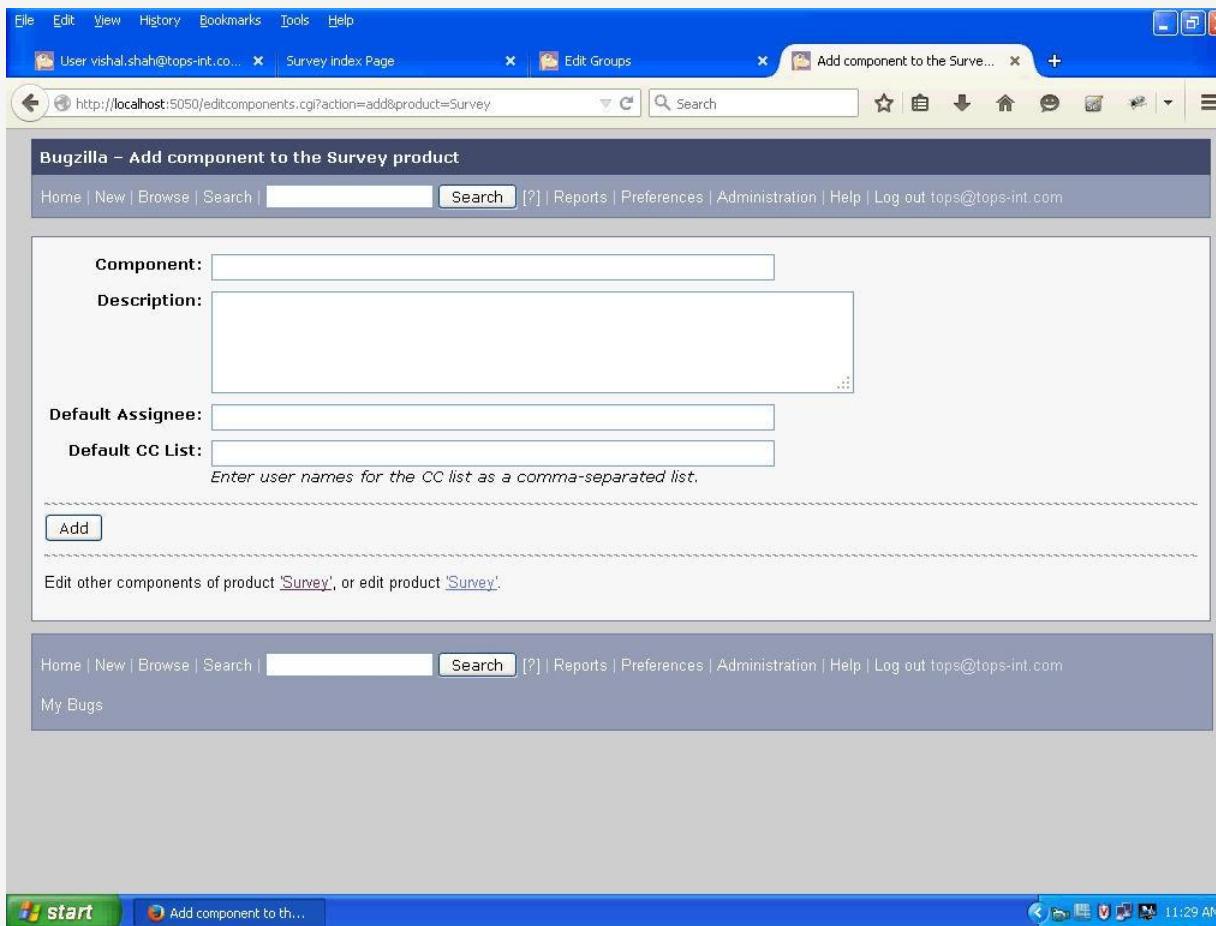
Select component on product



The screenshot shows a web browser window with the following details:

- Address Bar:** http://localhost:5050/editcomponents.cgi?product=Survey
- Title Bar:** Bugzilla – Select component of product 'Survey'
- Header:** Home | New | Browse | Search | [?] | Reports | Preferences | Administration | Help | Log out tops@tops-int.com
- Table:** A table with four columns: "Edit component...", "Description", "Default Assignee", and "Action". The "Edit component..." column is highlighted with a blue background. The "Action" column contains the text "<none>".
- Text Links:** Add a new component to product 'Survey', Redisplay table with bug counts (slower), Edit product 'Survey'.
- Footer:** Home | New | Browse | Search | [?] | Reports | Preferences | Administration | Help | Log out tops@tops-int.com
- Bottom Status Bar:** My Bugs
- Taskbar:** Shows the Windows Start button and the title "Select component of ...". The system tray indicates the date and time as 11:29 AM.

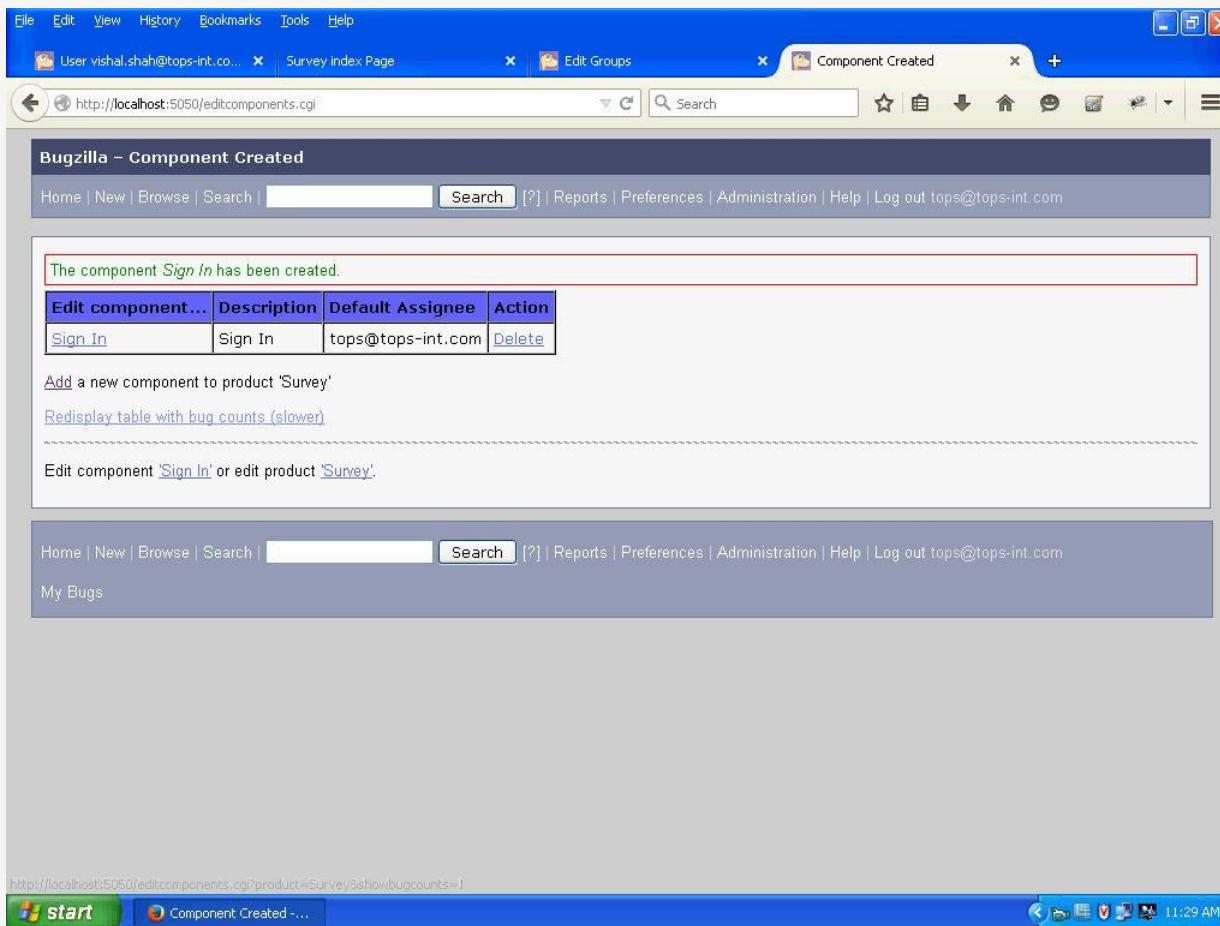
Add component to product



The screenshot shows a web browser window with the following details:

- Address Bar:** http://localhost:5050/editcomponents.cgi?action=add&product=Survey
- Title Bar:** Add component to the Survey product
- Content Area:**
 - Component:** [Text input field]
 - Description:** [Text area]
 - Default Assignee:** [Text input field]
 - Default CC List:** [Text input field]
Enter user names for the CC list as a comma-separated list.
 - Add:** [Submit button]
 - Link:** Edit other components of product 'Survey', or edit product 'Survey'.
- Bottom Navigation:** Home | New | Browse | Search | Search | Reports | Preferences | Administration | Help | Log out tops@tops-int.com
- Bottom Status Bar:** My Bugs

Component Added

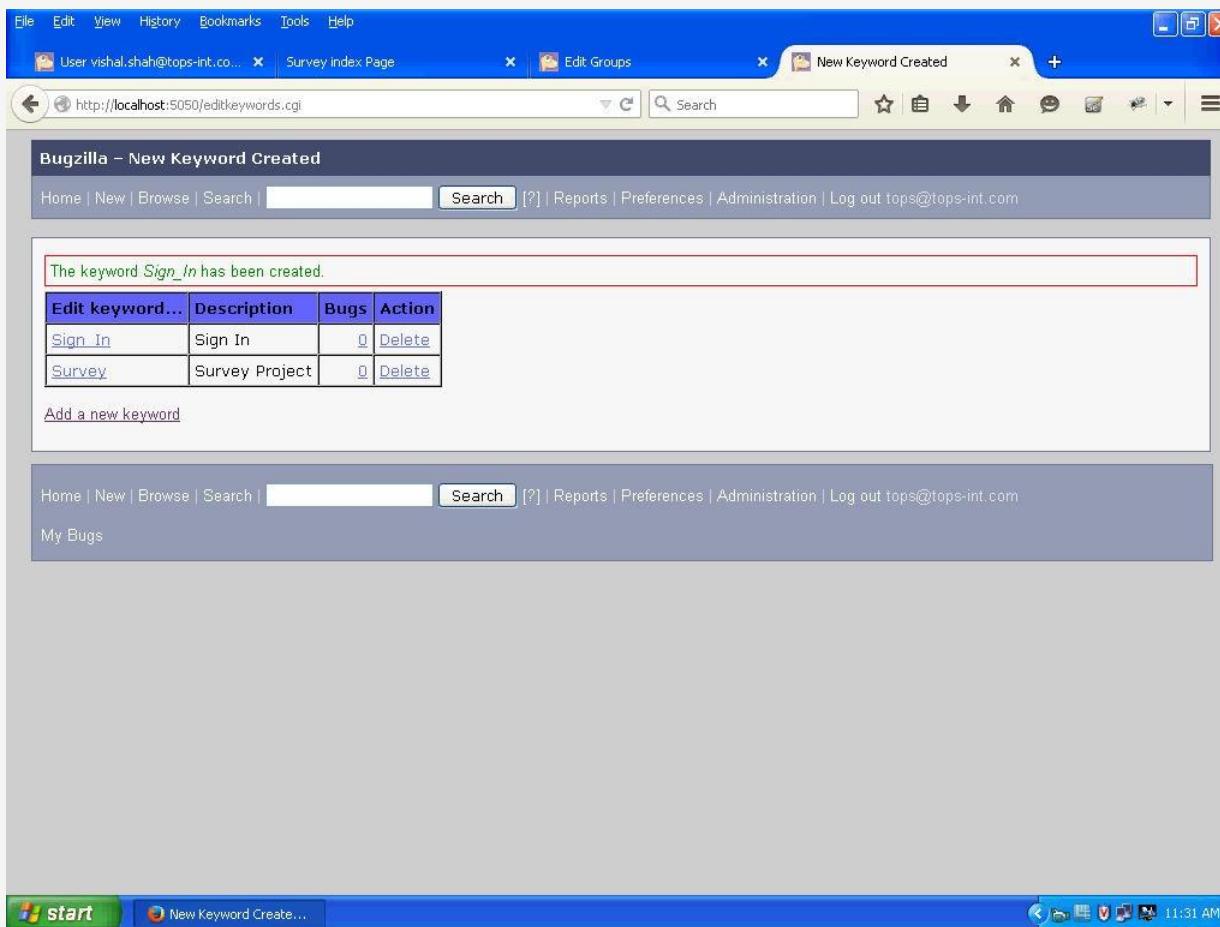


The screenshot shows a web browser window with the following details:

- Address Bar:** http://localhost:5050/editcomponents.cgi
- Title Bar:** Survey index Page, Edit Groups, Component Created
- Content Area:**
 - Bugzilla – Component Created**
 - Message:** The component *Sign In* has been created.
 - Table:** A table showing the newly created component.

Edit component...	Description	Default Assignee	Action
Sign In	Sign In	tops@tops-int.com	Delete
 - Links:** Add a new component to product 'Survey', Redisplay table with bug counts (slower), Edit component [Sign In](#) or edit product [Survey](#).
 - Footer:** Home | New | Browse | Search | Search | Reports | Preferences | Administration | Help | Log out tops@tops-int.com
- Taskbar:** Shows the URL http://localhost:5050/editcomponents.cgi?product=Survey&show=bugcounts=1, the Windows Start button, and the title Component Created - ...

New Keyword Added

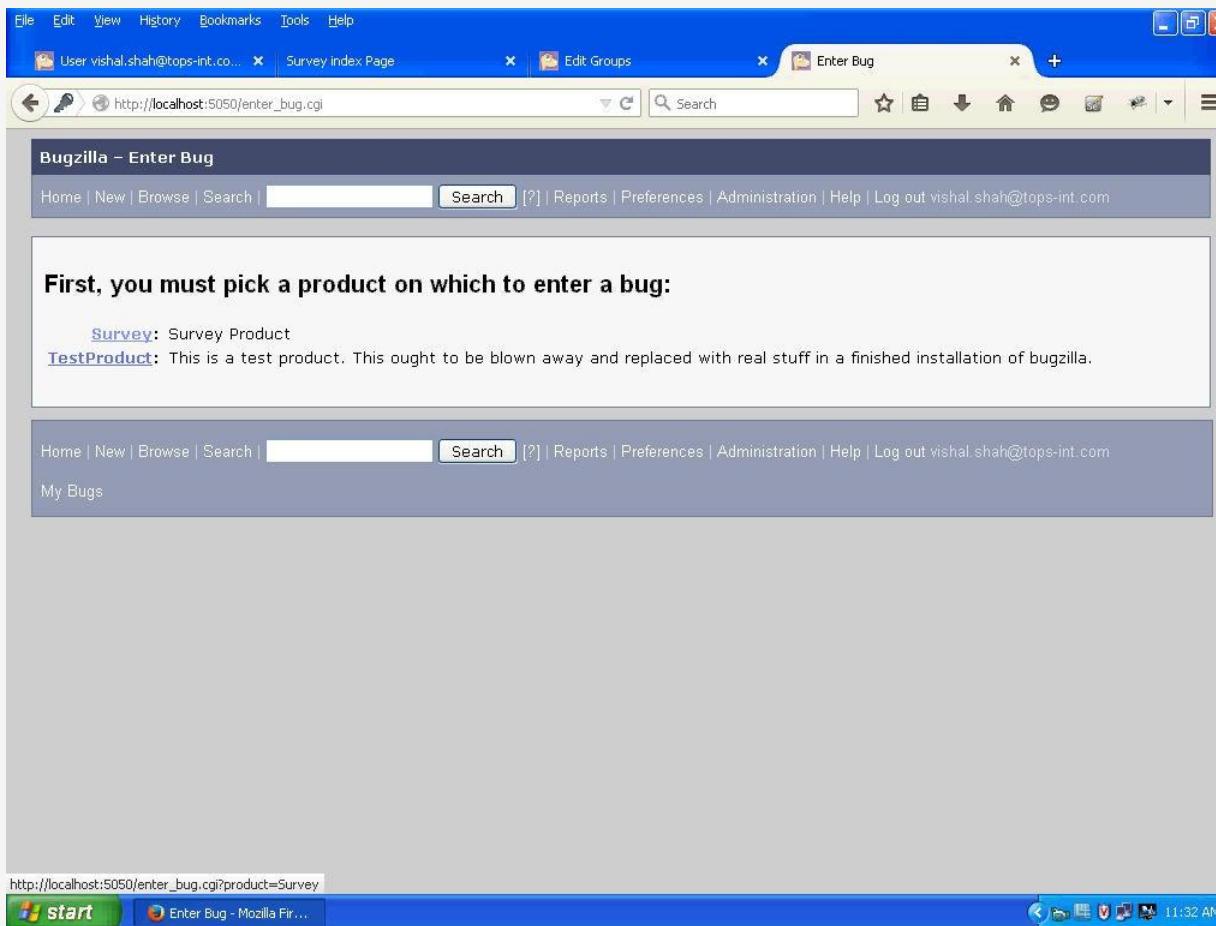


The screenshot shows a web browser window with three tabs: "User vishal.shah@tops-int.co...", "Survey index Page", and "Edit Groups". The active tab is "New Keyword Created" at the URL <http://localhost:5050/editkeywords.cgi>. The page title is "Bugzilla – New Keyword Created". It displays a message: "The keyword *Sign_In* has been created." Below this is a table:

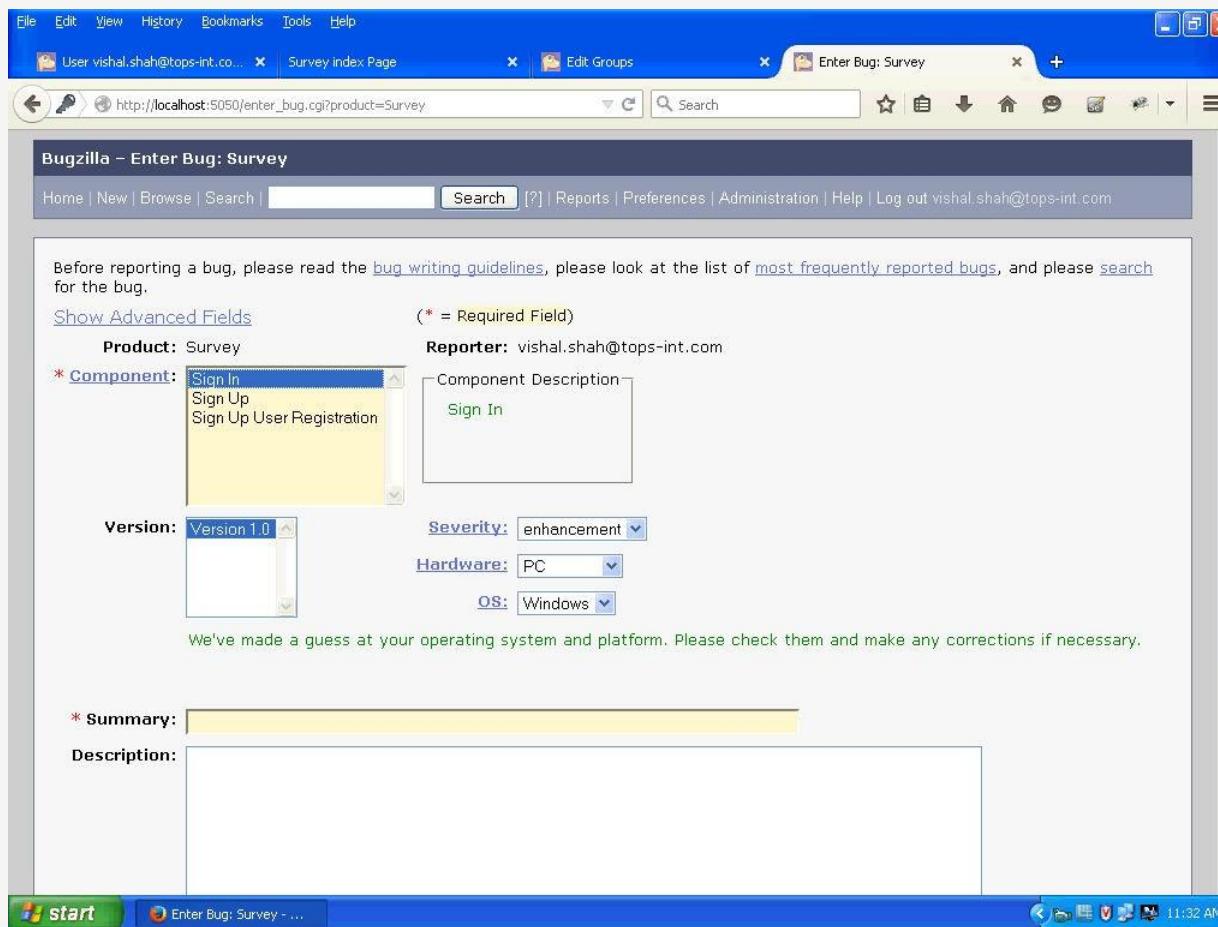
Edit keyword...	Description	Bugs	Action
Sign_In	Sign In	0	Delete
Survey	Survey Project	0	Delete

Below the table is a link "[Add a new keyword](#)". The browser's status bar shows "New Keyword Create...".

Select Product before enter new Bug



New Bug entry Main Page



File Edit View History Bookmarks Tools Help

User vishal.shah@tops-int.co... Survey index Page Edit Groups Enter Bug: Survey

http://localhost:5050/enter_bug.cgi?product=Survey

Bugzilla – Enter Bug: Survey

Home | New | Browse | Search | Search | [?] | Reports | Preferences | Administration | Help | Log out vishal.shah@tops-int.com

Before reporting a bug, please read the [bug writing guidelines](#), please look at the list of [most frequently reported bugs](#), and please [search](#) for the bug.

(* = Required Field)

Show Advanced Fields

Product: Survey **Reporter:** vishal.shah@tops-int.com

*** Component:** Sign In
Sign Up
Sign Up User Registration

Component Description:
Sign In

Version: Version 1.0 **Severity:** enhancement

Hardware: PC **OS:** Windows

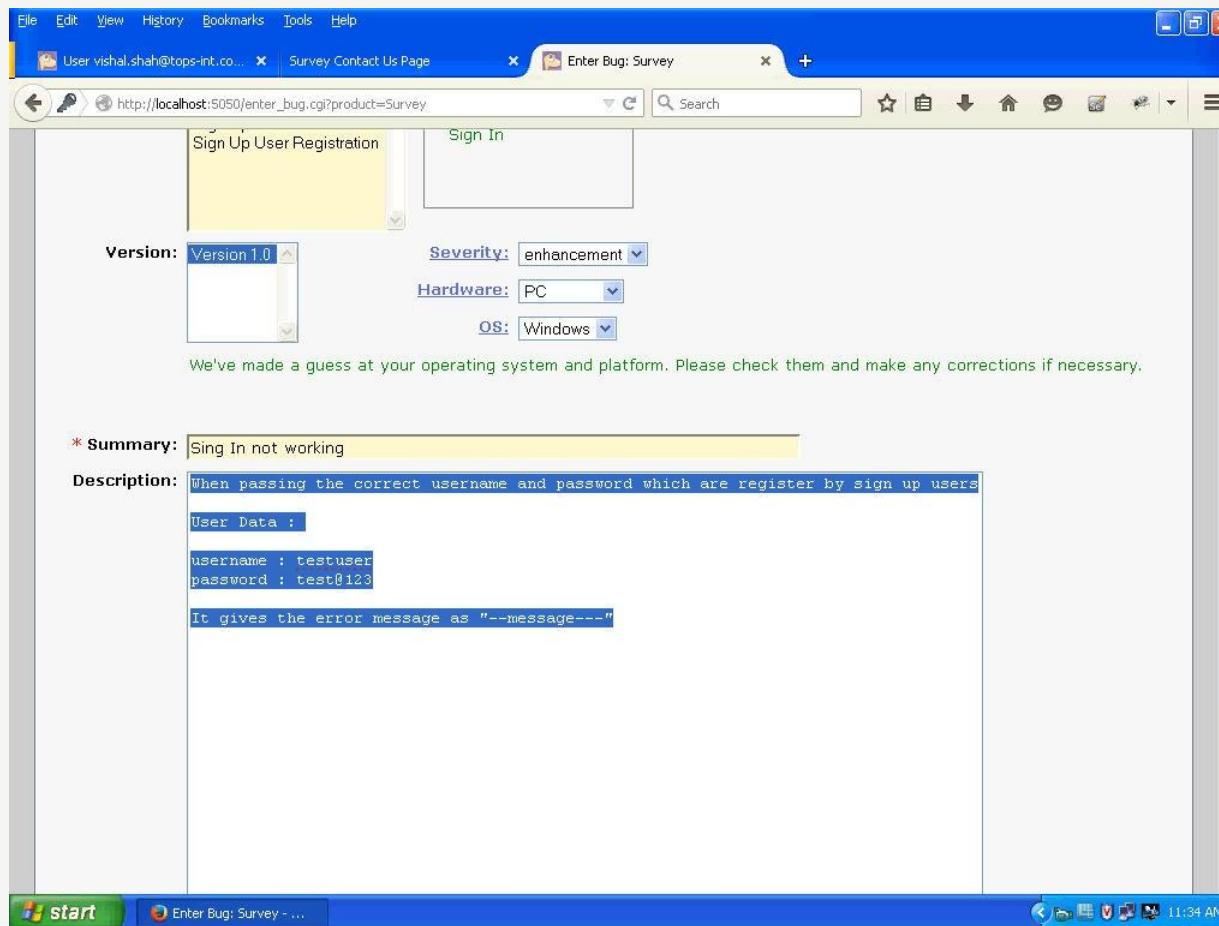
We've made a guess at your operating system and platform. Please check them and make any corrections if necessary.

*** Summary:**

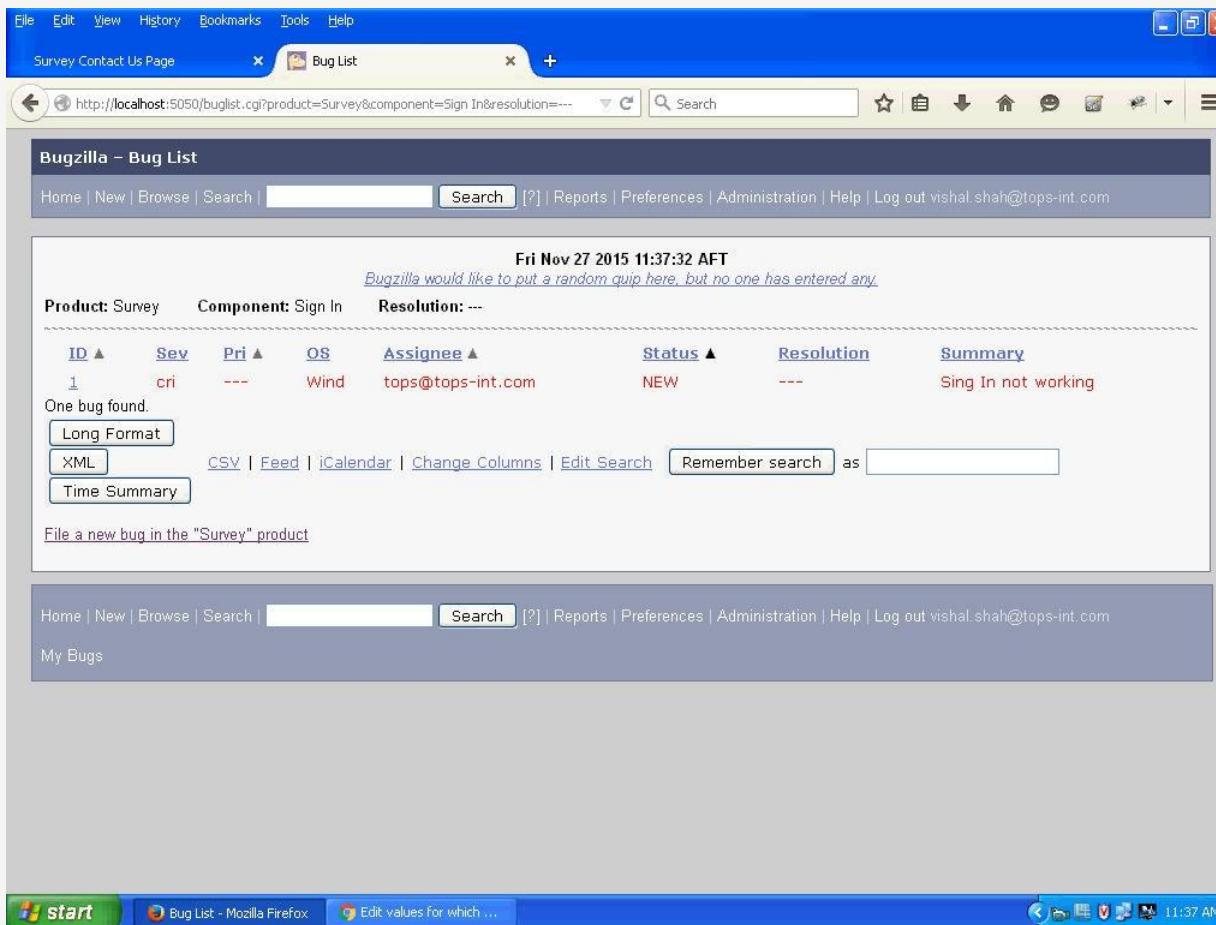
Description:

start Enter Bug: Survey - ... 11:32 AM

Add details into the new bug entry



Bugs which were entered



File Edit View History Bookmarks Tools Help

Survey Contact Us Page Bug List

http://localhost:5050/buglist.cgi?product=Survey&component=Sign+In&resolution=---

Search

Bugzilla – Bug List

Home | New | Browse | Search | Search | [?] | Reports | Preferences | Administration | Help | Log out vishal.shah@tops-int.com

Fri Nov 27 2015 11:37:32 AFT

Bugzilla would like to put a random quip here, but no one has entered any.

Product: Survey Component: Sign In Resolution: ---

ID ▲	Sev ▲	Pri ▲	OS ▲	Assignee ▲	Status ▲	Resolution	Summary
1	cri	---	Wind	tops@tops-int.com	NEW	---	Sing In not working

One bug found.

Long Format
XML
Time Summary

CSV | Feed | iCalendar | Change Columns | Edit Search | Remember search as

File a new bug in the "Survey" product

Home | New | Browse | Search | Search | [?] | Reports | Preferences | Administration | Help | Log out vishal.shah@tops-int.com

My Bugs

start Bug List - Mozilla Firefox Edit values for which ... 11:37 AM

Assign the Bug

File Edit View History Bookmarks Tools Help

Survey Contact Us Page x Bug 1 – Sing In not working +

http://localhost:5050/show_bug.cgi?id=1

Search

Bug 1 - Sing In not working ([edit](#))

Status: NEW ([edit](#)) **Reported:** 2015-11-27 11:36 AFT by [vishal](#)

Product: Survey **Modified:** 2015-11-27 11:36 AFT ([History](#))

Component: Sign In **CC List:** Add me to CC list
0 users ([edit](#))

Version: Version 1.0 **See Also:** [Add Bug URLs:](#) []

Platform: All Windows

Importance: High critical

Assigned To: tops ([edit](#))

URL: <http://survey-javatops.rhcloud.com/contact.jsp>

Keywords: []

Depends on: []

Blocks: []

Show dependency tree / graph

Orig. Est.	Current Est.	Hours Worked	Hours Left	%Complete	Gain	Deadline
0.0	0.0	0.0 + 0	0.0	0	0.0	(YYYY-MM-DD)

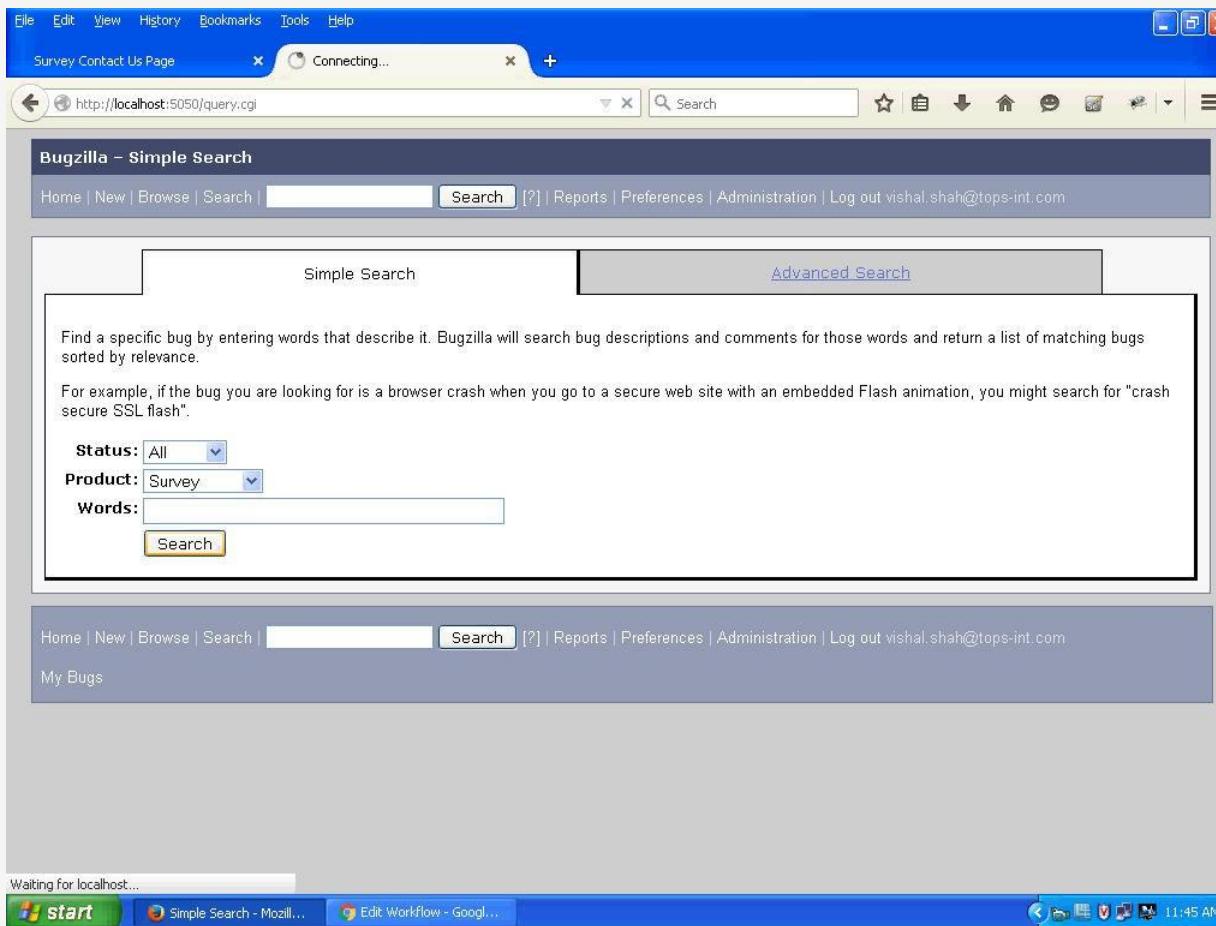
[Summarize time \(including time for bugs blocking this bug\)](#)

Attachments
[Add an attachment \(proposed patch, testcase, etc.\)](#)

Additional Comments:
[]

start Bug 1 – Sing In not w... Edit values for which ... 11:38 AM

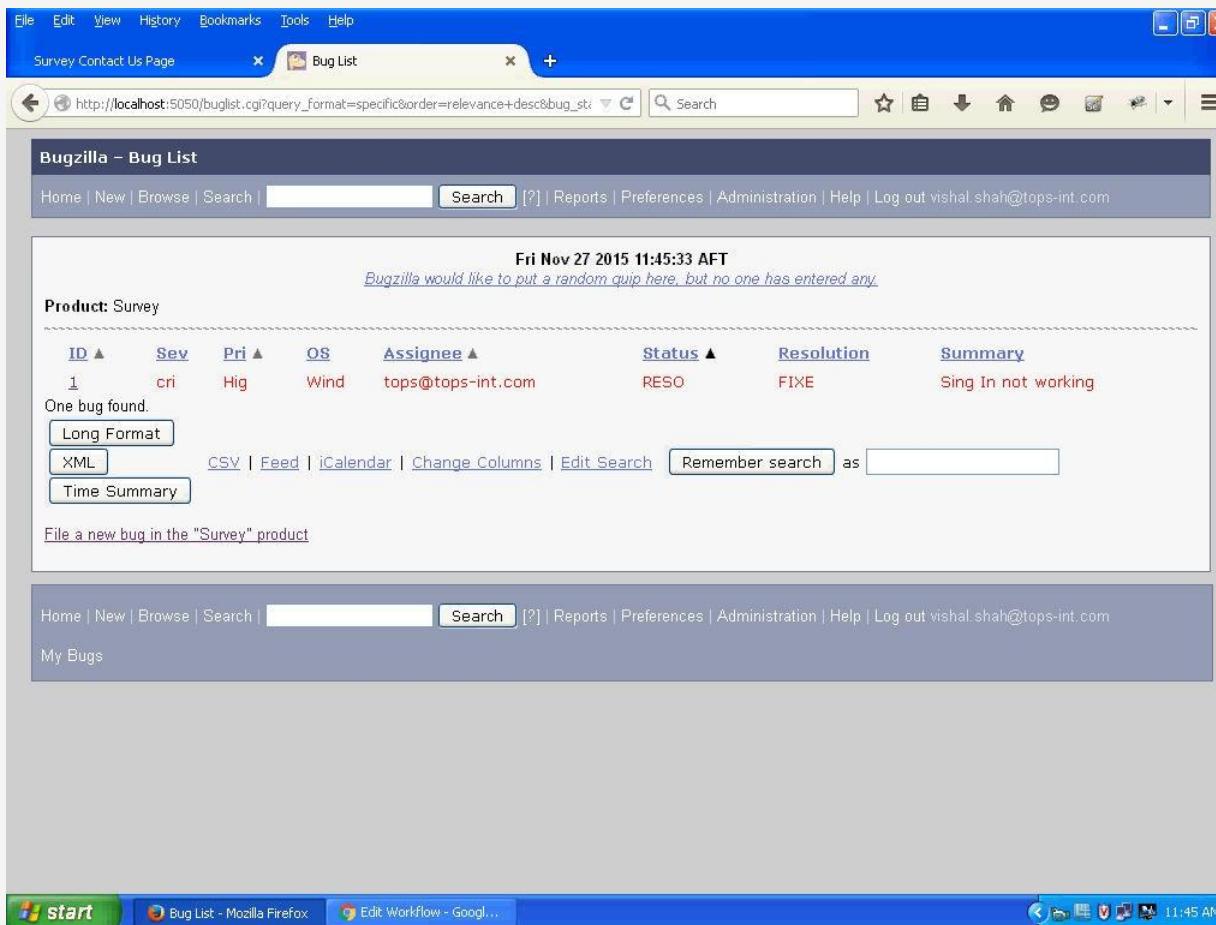
Search the Entered Bug



The screenshot shows a web browser window with the following details:

- Title Bar:** File Edit View History Bookmarks Tools Help
Survey Contact Us Page x Connecting... +
- Address Bar:** http://localhost:5050/query.cgi
- Page Title:** Bugzilla – Simple Search
- Header:** Home | New | Browse | Search | Search | [?] | Reports | Preferences | Administration | Log out vishal.shah@tops-int.com
- Content Area:**
 - Simple Search:** A section containing instructions and search filters.
 - Instructions: "Find a specific bug by entering words that describe it. Bugzilla will search bug descriptions and comments for those words and return a list of matching bugs sorted by relevance."
 - For example, if the bug you are looking for is a browser crash when you go to a secure web site with an embedded Flash animation, you might search for "crash secure SSL flash".
 - Filters:
 - Status: All
 - Product: Survey
 - Words:
 - Search button:
 - Advanced Search:** A link to the advanced search interface.
- Footer:** Home | New | Browse | Search | Search | [?] | Reports | Preferences | Administration | Log out vishal.shah@tops-int.com
- Bottom Status Bar:** My Bugs
- Bottom Taskbar:** Waiting for localhost...
start Simple Search - Mozilla... Edit Workflow - Google...
- System Tray:** 11:45 AM

Resolved Bug Status



File Edit View History Bookmarks Tools Help

Survey Contact Us Page Bug List

http://localhost:5050/buglist.cgi?query_format=specific&order=relevance+desc&bug_st... C Search

Bugzilla – Bug List

Home | New | Browse | Search | Search | [?] | Reports | Preferences | Administration | Help | Log out vishal.shah@tops-int.com

Fri Nov 27 2015 11:45:33 AFT
Bugzilla would like to put a random quip here, but no one has entered any.

Product: Survey

ID ▲	Sev ▲	Pri ▲	OS	Assignee ▲	Status ▲	Resolution	Summary
1	cri	Hig	Wind	tops@tops-int.com	RESO	FIXE	Sing In not working

One bug found.

Long Format CSV | Feed | iCalendar | Change Columns | Edit Search Remember search as

Time Summary

File a new bug in the "Survey" product

Home | New | Browse | Search | Search | [?] | Reports | Preferences | Administration | Help | Log out vishal.shah@tops-int.com

My Bugs

start Bug List - Mozilla Firefox Edit Workflow - Google...

11:45 AM

Module – 3 [Testing On Live Application]

Agenda

- Responsive Testing

- Basic Intro Of API Testing

- Cross Browser Testing

- Advanced Mobile Testing(apk, ipa)

- Database / SQL

Responsive Testing

What is Responsive Testing?

- A responsive web design involves creating a flexible web page that is accessible from any device, starting from a mobile phone to a tablet.
- Furthermore, a responsive web design improves users' browsing experience.
- Considering this from a quality assurance perspective, a responsive web design requires thorough evaluation using a variety of devices before it is ready to go live.
- Software testers may find it challenging to perform responsive design testing as a variety of factors are to be looked into during the testing phase.
- Some points to be understand for Responsive Testing.
 - The challenges involved in testing a responsive website
 - How website testing differs from a mobile device to a computer
 - Rules and guidelines to be followed during responsive design testing and
 - Lastly, various tools available to perform responsive testing

Responsive Testing Tools

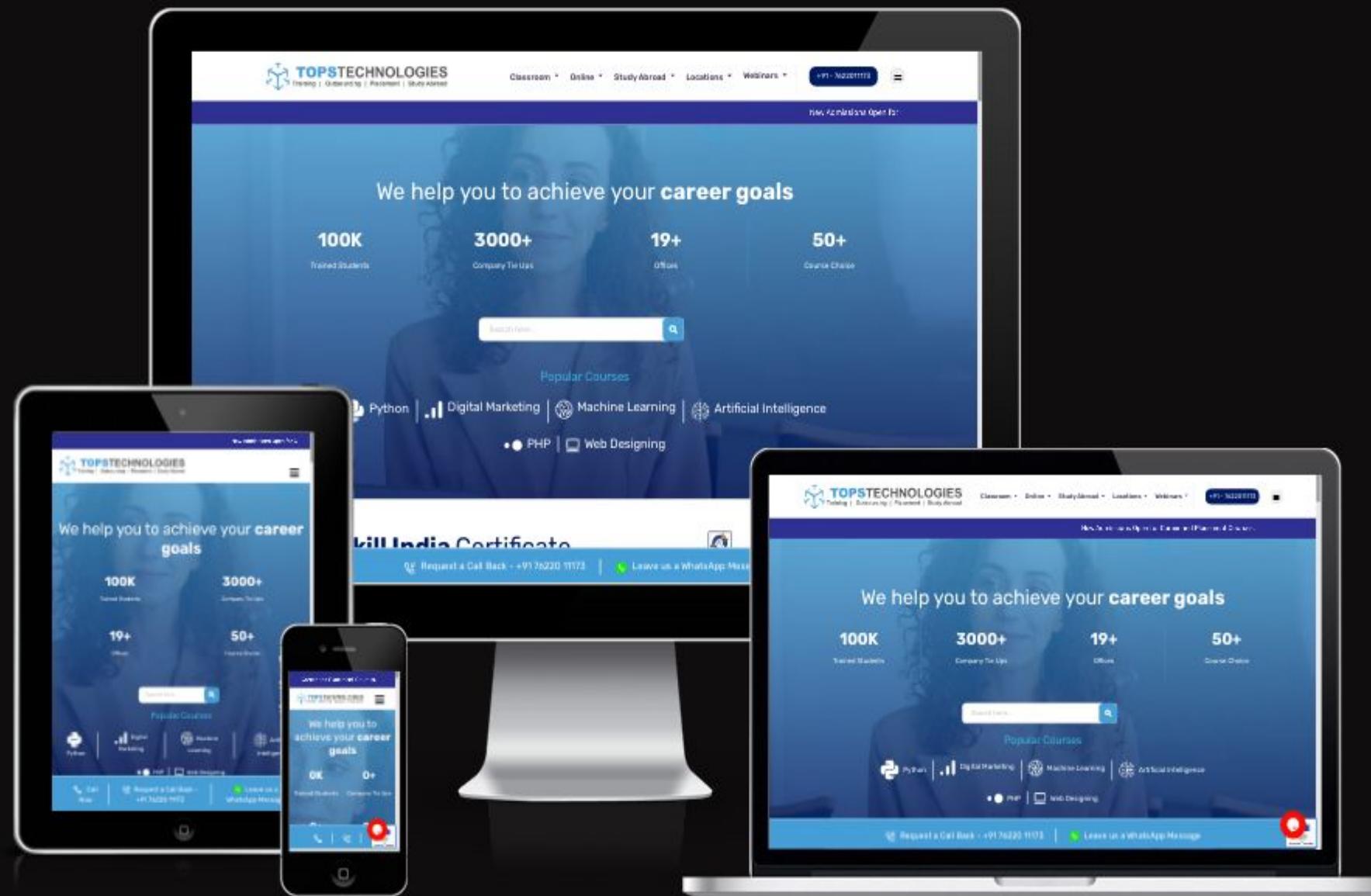
- LT Browser
- Lambda Testing
- Google Resizer
- I am responsive
- Pixel tuner

For Ex : <https://ui.dev/amiresponsive>

1) Search on this box to enter your site name

<https://www.tops-int.com/>

GO!



API Testing

What is API Testing?

- Application Programming Interface (API) is a **software interface that allows two applications to interact with each other without any user intervention**
- another definition , **API (Application Programming Interface)** is a computing interface which enables communication and data exchange between two separate software systems.
- The purpose of API Testing is to check the functionality, reliability, performance, and security of the programming interfaces.
- In API Testing, instead of using standard user inputs(keyboard) and outputs, you use software to send calls to the API, get output, and note down the system's response.
- API tests are very different from GUI Tests and won't concentrate on the look and feel of an application.

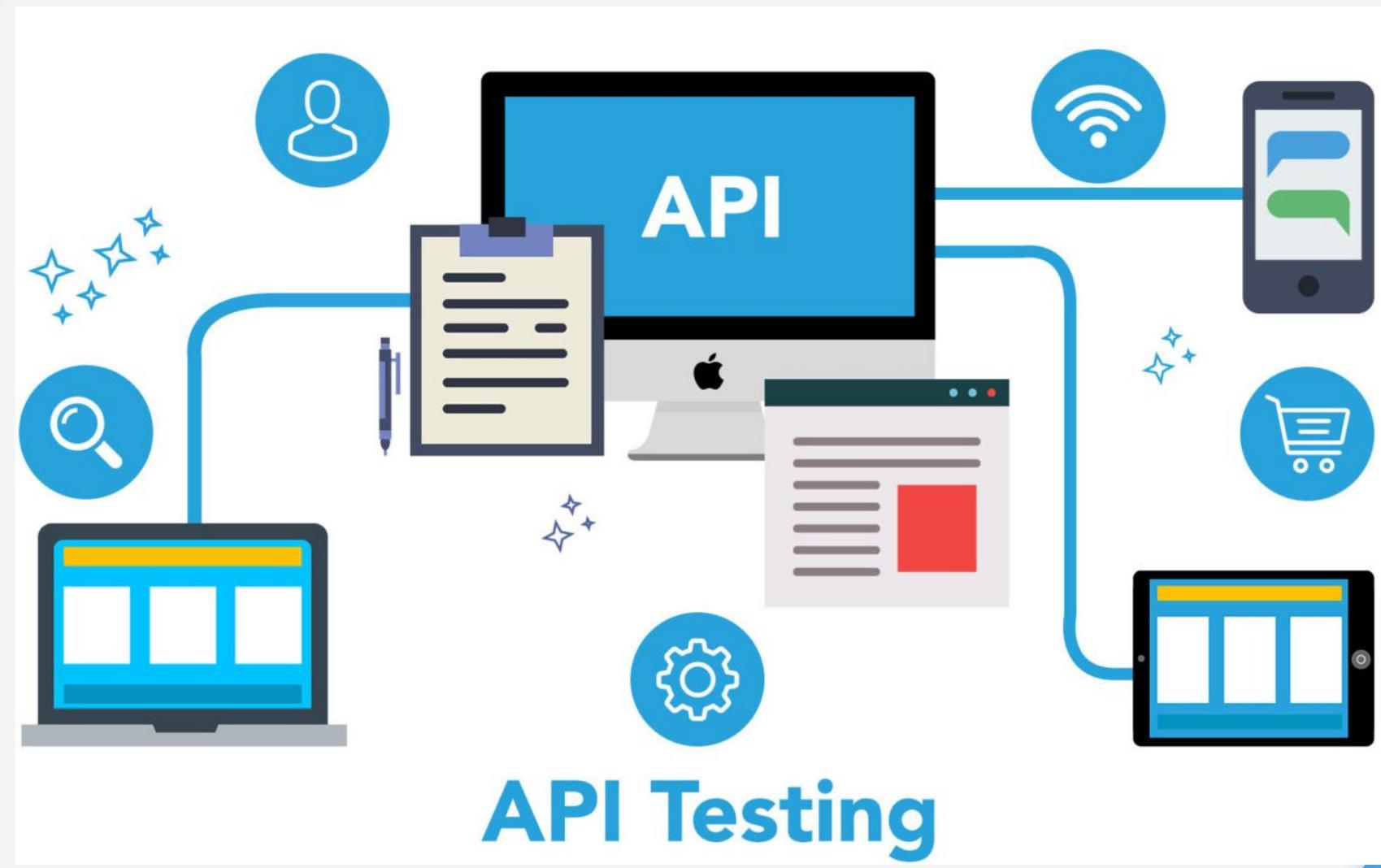
Types of API Testing

There are mainly 3 types of API Testing

- **Open APIs:** These types of APIs are publicly available to use like OAuth APIs from Google. It has also not given any restriction to use them. So, they are also known as Public APIs.
- **Partner APIs:** Specific rights or licenses to access this type of API because they are not available to the public.
- **Internal APIs:** Internal or private. These APIs are developed by companies to use in their internal systems. It helps you to enhance the productivity of your teams.

Tools for API Testing

- PostMan
- SoapUI
- Jmeter
- VRest



Cross Browser Testing

What is Cross Browser Testing?

- Cross Browser Testing checks the compatibility of a web application or website on different browsers, operating systems, and devices.
- Through Cross Browser Testing we ensure that the web application or website works in a uniform and smooth fashion on the browsers and devices that your customers may use –
 - **Browsers** - Chrome, Firefox, Safari, Internet Explorer, Edge, UC Browser, Opera, etc.
 - **Operating systems** - Windows, Android, iOS, macOS, etc.
 - **Devices** - Mobile, laptop, desktop, tablet, smart TV, etc.
- A user might be using any of the devices with any combination of browser or OS. Hence, it is required to check the compatibility, performance, and UI of the application on all these combinations.

Why is Cross Browser Testing needs?

- Any website or application consists of HTML, JavaScript, and CSS components.
- We do not know over which browser our user might see our application.
- We know that each browser has its technology, capabilities, and limitations.
- Because each browser interprets an application in its own way, this results in different outputs for different users.
- We need to be sure that when rendered on different browsers the final output should be the same for all users.

Tools Are:

- Lambda Test**
- BrowserStack**
- SauceLab**

Advanced Mobile Testing

Android Version List

Android Lollipop	Lemon Meringue Pie	5.0 – 5.0.2	21	November 4, 2014 ^[17]	November 2017
		5.1 – 5.1.1	22	March 2, 2015 ^[18]	March 2018
Android Marshmallow	Macadamia Nut Cookie	6.0 – 6.0.1	23	October 2, 2015 ^[19]	August 2018
Android Nougat	New York Cheesecake	7.0	24	August 22, 2016	August 2019
		7.1 – 7.1.2	25	October 4, 2016	October 2019
Android Oreo	Oatmeal Cookie	8.0	26	August 21, 2017	January 2021
		8.1	27	December 5, 2017	October 2021
Android Pie	Pistachio Ice Cream ^[20]	9	28	August 6, 2018	January 2022
Android 10	Quince Tart ^[21]	10	29	September 3, 2019	August 2022
Android 11	Red Velvet Cake ^[21]	11	30	September 8, 2020	
Android 12	Snow Cone	12	31	October 4, 2021	
Android 12L	Snow Cone v2	12.1 ^[a]	32	March 7, 2022	
Android 13	Tiramisu ^[23]	13 ^[b]	33	Q3 2022	

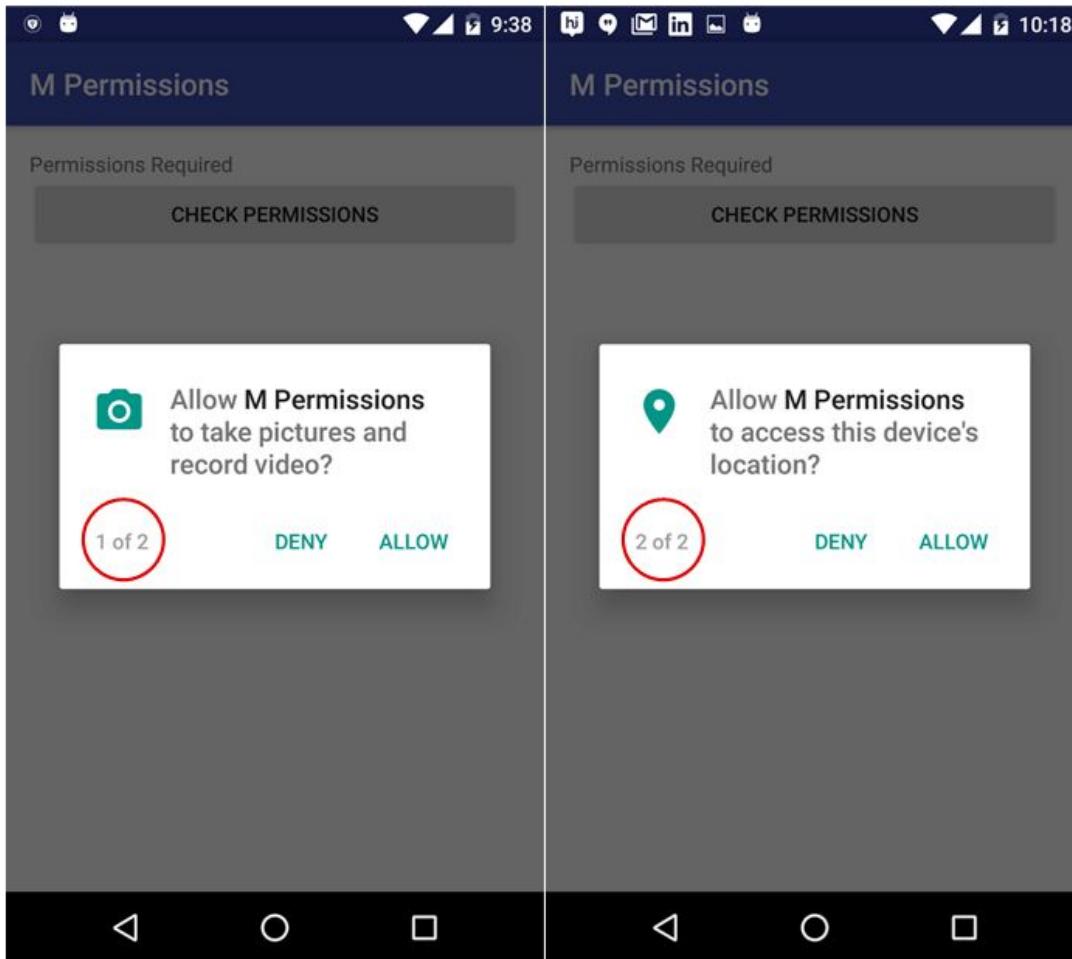
Legend: Old version Older version, still maintained Latest version Latest preview version

Iphone Version List

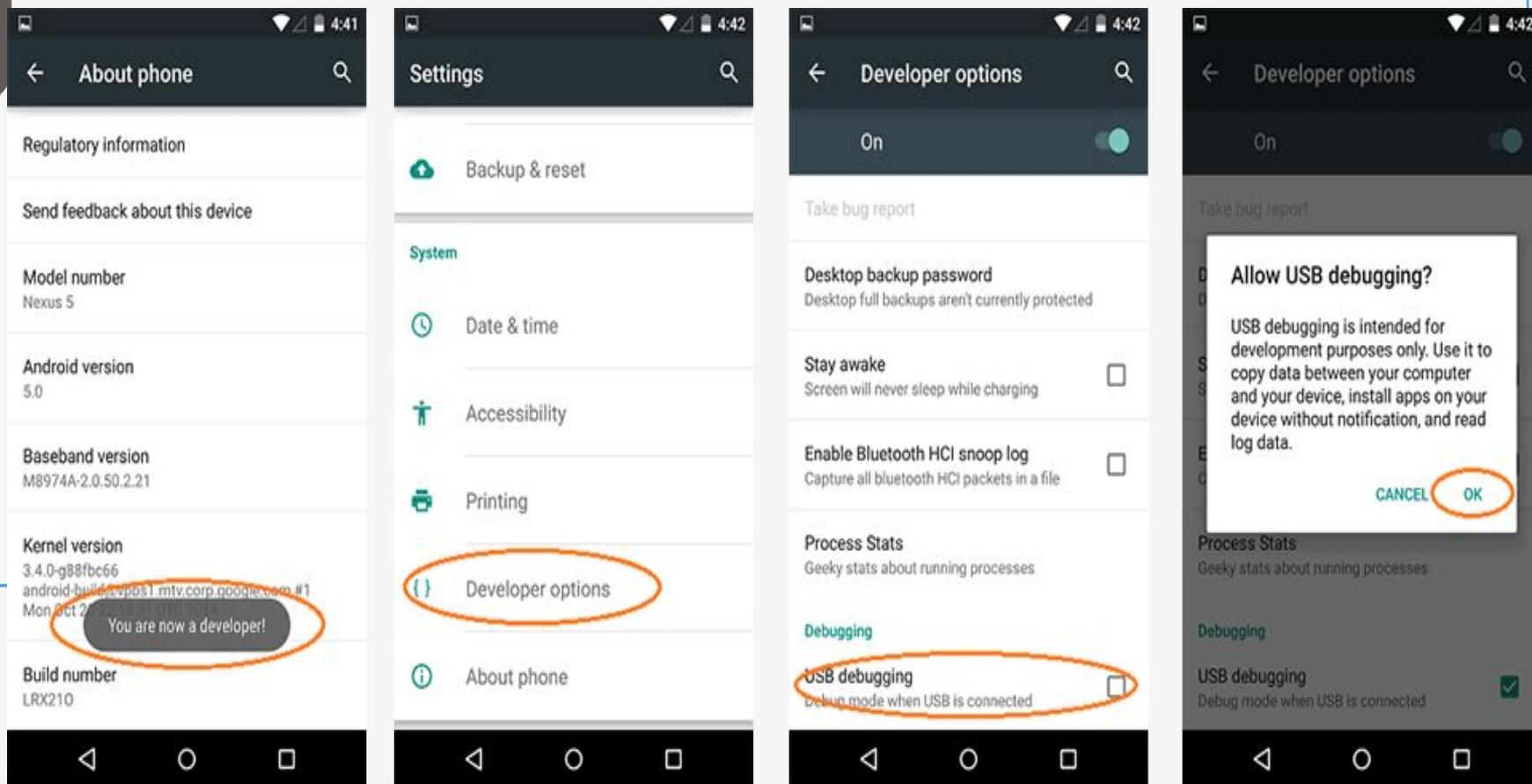
iPhone 11	iOS 13.0	September 20, 2019		<i>current</i>	<i>latest iOS</i>
iPhone 11 Pro / 11 Pro Max	iOS 13.0	September 20, 2019	October 13, 2020	<i>current</i>	<i>latest iOS</i>
iPhone SE (2nd)	iOS 13.4	April 24, 2020	March 8, 2022	<i>current</i>	<i>latest iOS</i>
iPhone 12 / 12 Mini	iOS 14.1 (12) iOS 14.2 (12 Mini)	October 23, 2020 (12) November 13, 2020 (12 Mini)		<i>current</i>	<i>latest iOS</i>
iPhone 12 Pro / 12 Pro Max	iOS 14.1 (12 Pro) iOS 14.2 (12 Pro Max)	October 23, 2020 (12 Pro) November 13, 2020 (12 Pro Max)	September 14, 2021	<i>current</i>	<i>latest iOS</i>
iPhone 13 / 13 Mini	iOS 15.0	September 24, 2021		<i>current</i>	<i>latest iOS</i>
iPhone 13 Pro / 13 Pro Max	iOS 15.0	September 24, 2021		<i>current</i>	<i>latest iOS</i>
iPhone SE (3rd)	iOS 15.4	March 18, 2022		<i>current</i>	<i>latest iOS</i>

Android App Permission

Android M Permissions -Requesting Multiple permissions



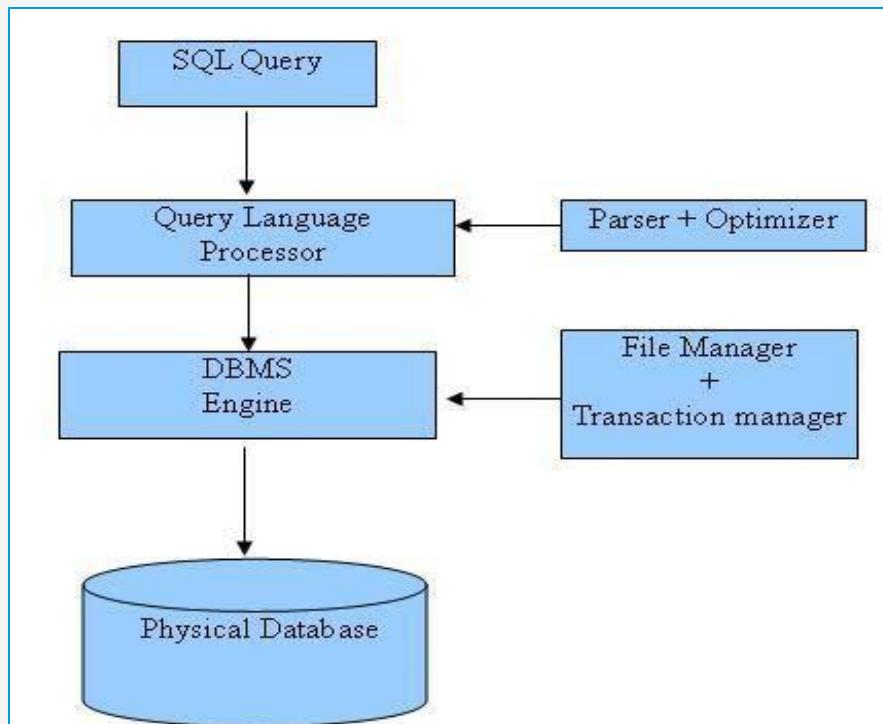
How to ON Developer Option Mode?



Database / SQL

DataBase and SQL Objectives

“The large majority of today's business applications revolve around relational databases and the SQL programming language (Structured Query Language). Few businesses could function without these technologies...”



Relational Databases

RDBMS stands for **Relational Database Management System**. RDBMS is the basis for SQL, and for all modern database systems like MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.

A Relational database management system (RDBMS) is a database management system (DBMS) that is based on the relational model as introduced by **E. F. Codd**.

Most of today's databases are relational:

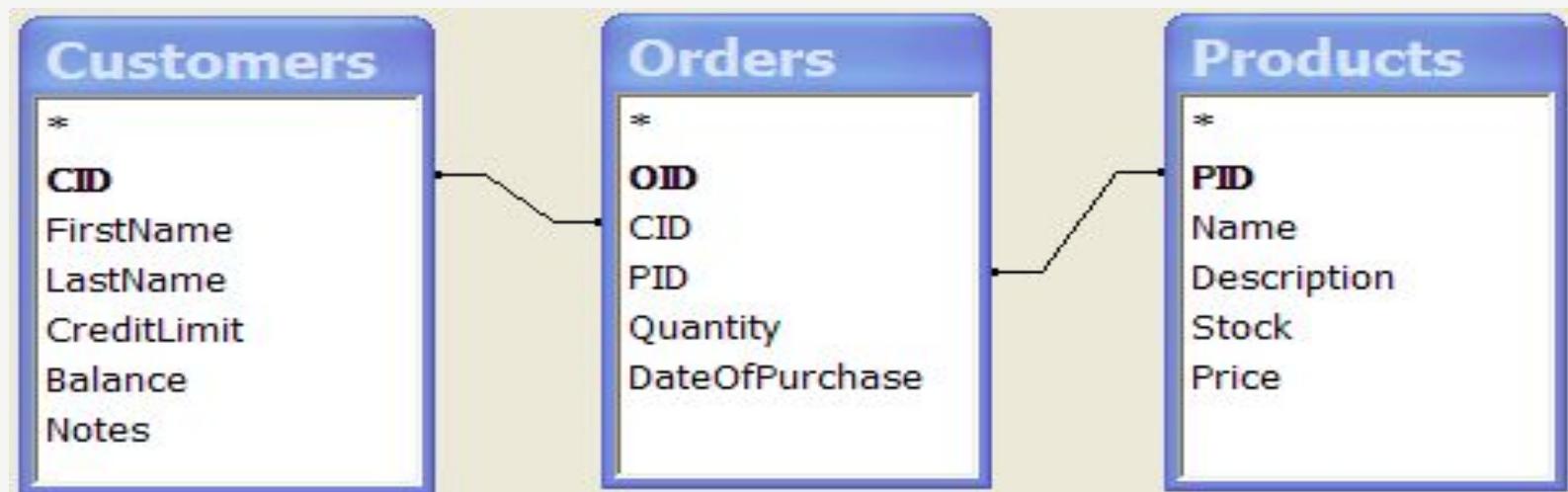
- database contains 1 or more *tables*
- table contains 1 or more *records*
- record contains 1 or more *fields*
- fields contain the data

So why is it called "relational"?

- tables are related (*joined*) based on common fields

Example

- Here's a simple **database schema** for tracking sales
 - 3 tables, related by primary keys (CID, OID, PID)
 - primary keys (in **boldface**) are unique record identifiers
 - customer may place order for one product at a time...



Customers...

- Here's some data for the Customers table
 - ignore last row, it's a MS Access mechanism for adding rows...

	CID	FirstName	LastName	CreditLimit	Balance	Notes
▶	1	Jim	Bag	\$1,000.00	\$0.00	works at the gym
	3	Kathie	O'Dahl	\$9,999.99	\$0.00	a friend with a special name!
	5	Bryan	Lore	\$1,000.00	\$900.00	a brother-in-law
	6	Amy	Lore	\$1,000.00	\$100.00	a sister-in-law
	14	Bill	Gates	\$2,000,000,000.00	\$89,992.00	
	116	Jane	Doe	\$1,000.00	\$420.00	
	666	Bad	Guy	\$1,000,000.00	\$235,000.00	a very bad guy...
*	0			\$0.00	\$0.00	

Products...

- Here's some data for the Products table

	PID	Name	Description	Stock	Price
▶	1	Flying Squirrels	yes, they really do fly!	3	\$899.99
	2	Cats		100	\$19.99
	3	Dogs	we only carry dalmations	20	\$79.03
	4	Ants		50000	\$0.09
	5	Birds		1000	\$4.95
	6	Elephants		10	\$389.95
	7	Red Fire Ants		10000	\$0.49
	8	Racoons		25	\$2.25
*	0			0	\$0.00

Record:  1  of 8

Orders...

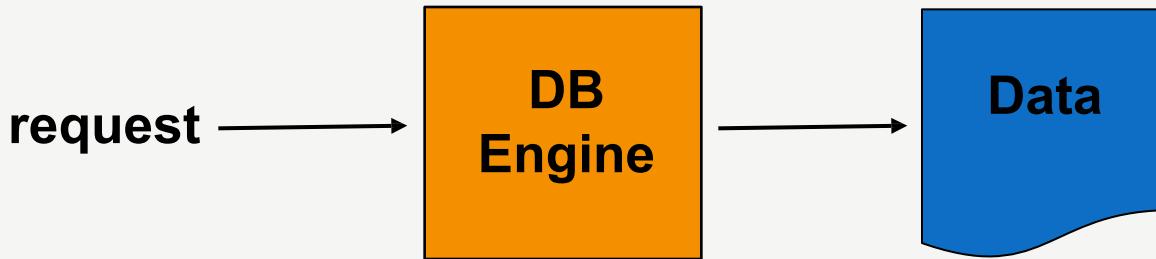
- Here's some data for the Orders table
 - how do you read this?
 - e.g. order #9906 states Kathie O'Dahl purchased 600 Ants
 - must join tables together to figure that out...

	OID	CID	PID	Quantity	DateOfPurchase
▶	9906	3	4	600	Wednesday, June 28, 2000
	12351	116	4	100	Tuesday, January 01, 2002
	22209	1	2	2	Thursday, January 03, 2002
	22210	1	2	1	Friday, June 28, 2002
	33410	1	3	1	Tuesday, October 01, 2002
*	0	0	0	0	

Database Management Systems

- A DBMS consists of 2 main pieces:

- the data
- the DB engine
- the data is typically stored in one or more files

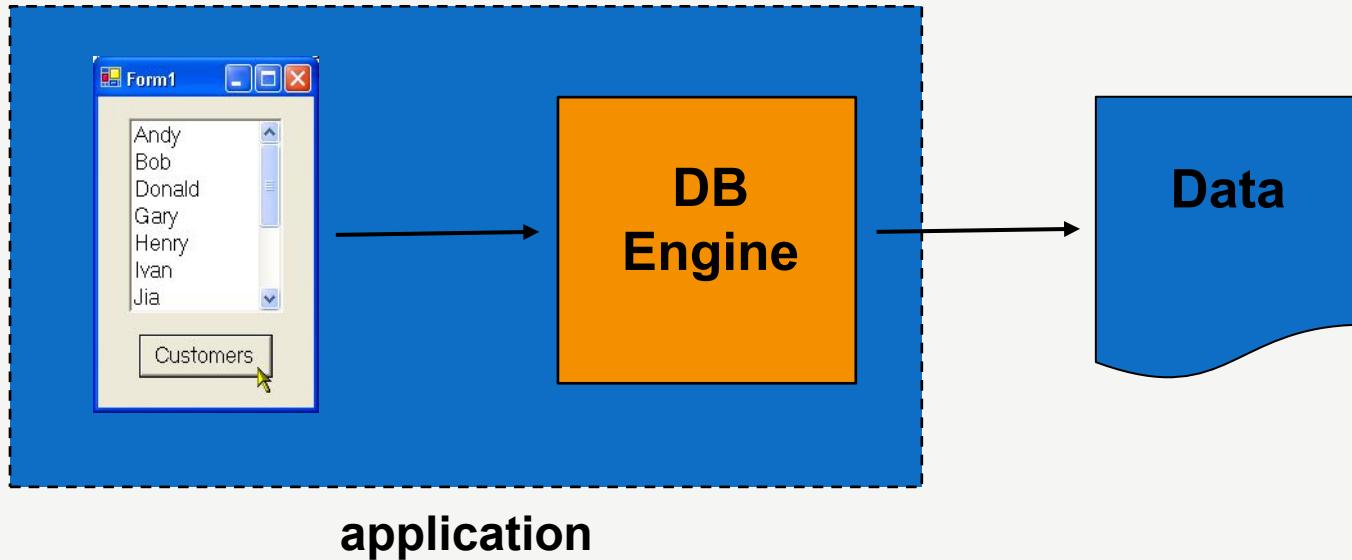


- Two most common types of DBMS are:

- Local
- Server

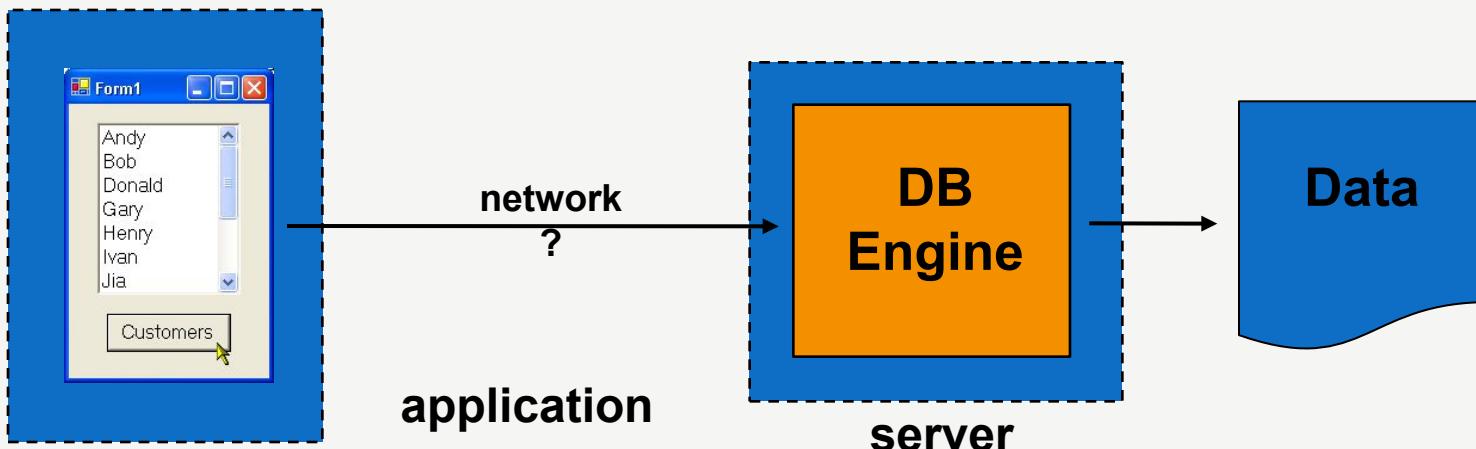
Local DBMS

- A local DBMS is where DB engine runs as part of application
- Example?
- MS Access
- underlying DB engine is JET ("Joint Engine Technology")



Server DBMS

- A server DBMS is where DB engine runs as a separate process
 - typically on a different machine (i.e. server)
- Examples?
 - MS SQL Server, Oracle, DB2, MySQL



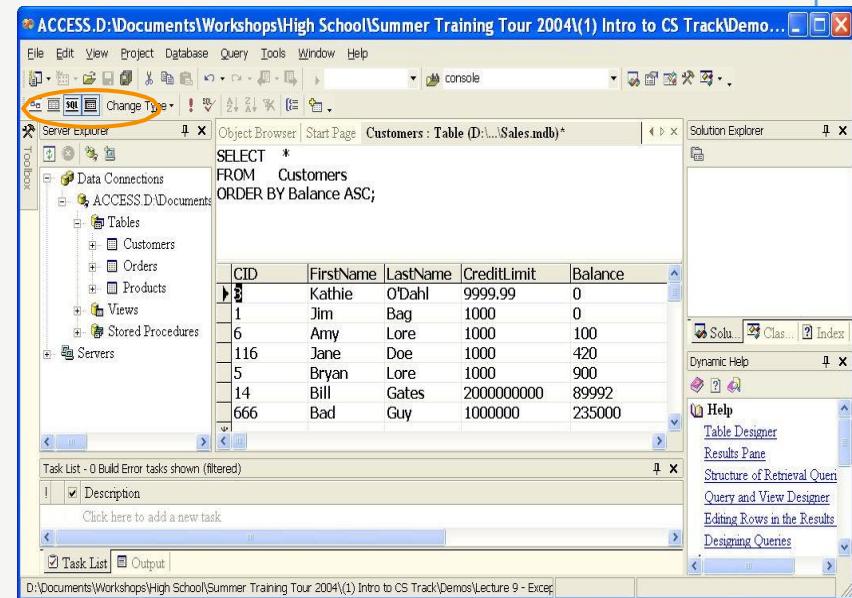
Databases & Visual Studio .NET

Most databases ship with tools for opening / manipulating DB

- MS Access database: use the MS Access Office product
- SQL Server database: use Query Analyzer

But you can also use Visual Studio .NET!

- View menu, Server Explorer
- right-click on Data Connections
- Add Connection...
- MS Access database:
 - Provider: JET 4.0 OLE DB
 - Connection:browse...
 - Connection:test...
- Drill-down to Tables
- Double-click on a table
- "Show SQL Pane" via toolbar



Structure Query Language

SQL tutorial gives unique learning on **Structured Query Language** and it helps to make practice on SQL commands which provides immediate results.

SQL is a language of database, it includes database creation, deletion, fetching rows and modifying rows etc.

SQL is an **ANSI (American National Standards Institute)** standard but there are many different versions of the SQL language.

SQL is the standard programming language of relational DBs

SQL is a standard computer language for accessing and manipulating databases.

SQL is a great example of a **declarative** programming language

- your declare what you want, DB engine figures out how...

What is SQL?

SQL is Structured Query Language, which is a computer language for storing, manipulating and retrieving data stored in relational database.

SQL is the standard language for Relation Database System. All relational database management systems like MySQL, MS Access, Oracle, Sybase, Informix, postgres and SQL Server use SQL as standard database language.

Also, they are using different dialects, such as:

- MS SQL Server using T-SQL, ANSI SQL
- Oracle using PL/SQL,
- MS Access version of SQL is called JET SQL (native format) etc.

Why SQL?

- Allows users to access data in relational database management systems.
- Allows users to describe the data.
- Allows users to define the data in database and manipulate that data.
- Allows to embed within other languages using SQL modules, libraries & pre-compilers.
- Allows users to create and drop databases and tables.
- Allows users to create view, stored procedure, functions in a database.
- Allows users to set permissions on tables, procedures, and views

What is SQL? (Cont...)

- SQL stands for Structured Query Language
- SQL allows you to access a database
- SQL is an ANSI standard computer language
- SQL can execute queries against a database
- SQL can retrieve data from a database
- SQL can insert new records in a database
- SQL can delete records from a database
- SQL can update records in a database
- SQL is easy to learn
- SQL is written in the form of *queries*
- *action queries* insert, update & delete data
- *select queries* retrieve data from DB

SQL Process

When you are executing an SQL command for any RDBMS, the system determines the best way to carry out your request and SQL engine figures out how to interpret the task.

There are various components included in the process.

- These components are Query Dispatcher, Optimization Engines, Classic Query Engine and SQL Query Engine, etc.
- Classic query engine handles all non-SQL queries but SQL query engine won't handle logical files.

SQL Commands

- **DDL** – Data Definition Language
- **DML** – Data Manipulation Language
- **DCL** – Data Control Language
- **DQL** – Data Query Language

SQL Join Types

- **INNER JOIN**: returns rows when there is a match in both tables.
- **LEFT JOIN**: returns all rows from the left table, even if there are no matches in the right table.
- **RIGHT JOIN**: returns all rows from the right table, even if there are no matches in the left table.
- **FULL JOIN**: returns rows when there is a match in one of the tables.

DDL - Data Definition Language

Command	Description
CREATE	Creates a new table, a view of a table, or other object in database
ALTER	Modifies an existing database object, such as a table.
DROP	Deletes an entire table, a view of a table or other object in the database.

DQL – Data Query Language

Command	Description
SELECT	Retrieves certain records from one or more tables

DML – Data Manipulation Language

Command	Description
INSERT	Creates a record
UPDATE	Modifies records
DELETE	Deletes records

DCL – Data Control Language

Command	Description
GRANT	Gives a privilege to user
REVOKE	Takes back privileges granted from user

Create, Drop, Use Database Syntax

- **SQL CREATE DATABASE STATEMENT**

```
CREATE DATABASE database_name;
```

- **SQL DROP DATABASE Statement:**

```
DROP DATABASE database_name;
```

- **SQL USE STATEMENT**

```
USE DATABASE database_name;
```

Create, Drop, Alter Table Syntax

SQL CREATE TABLE STATEMENT

```
CREATE TABLE table_name( column1 datatype, column2 datatype, column3  
datatype, ..... , columnN datatype, PRIMARY KEY( one or more columns ) );
```

SQL DROP TABLE STATEMENT

```
DROP TABLE table_name;
```

SQL TRUNCATE TABLE STATEMENT

```
TRUNCATE TABLE table_name;
```

SQL ALTER TABLE STATEMENT

```
ALTER TABLE table_name{ADD|DROP|MODIFY}column_name{data_type};
```

SQL ALTER TABLE STATEMENT (RENAME)

```
ALTER TABLE table_name RENAME TO new_table_name;
```

Insert, Update, Delete Syntax

SQL INSERT INTO STATEMENT

```
INSERT INTO table_name( column1, column2....columnN) VALUES ( value1,  
value2....valueN);
```

SQL UPDATE STATEMENT

```
UPDATE table_name SET column1 = value1, column2 =  
value2....columnN=valueN  
[ WHERE CONDITION ];
```

SQL DELETE STATEMENT

```
DELETE FROM table_name WHERE {CONDITION};
```

Select Statement Syntax

SQL SELECT STATEMENT

```
SELECT column1, column2....columnN FROM table_name;
```

SQL DISTINCT CLAUSE

```
SELECT DISTINCT column1, column2....columnN FROM  
table_name;
```

SQL WHERE CLAUSE

```
SELECT column1, column2....columnN FROM table_name WHERE  
CONDITION;
```

SQL AND/OR CLAUSE

```
SELECT column1, column2....columnN FROM table_name WHERE  
CONDITION-1 {AND|OR} CONDITION-2;
```

Select Statement Syntax

SQL IN CLAUSE

```
SELECT column1, column2....columnN FROM table_name WHERE  
column_name IN (val-1, val-2,...val-N);
```

SQL BETWEEN CLAUSE

```
SELECT column1, column2....columnN FROM table_name WHERE  
column_name BETWEEN val-1 AND val-2;
```

SQL LIKE CLAUSE

```
SELECT column1, column2....columnN FROM table_name WHERE  
column_name LIKE { PATTERN };
```

SQL ORDER BY CLAUSE

```
SELECT column1, column2....columnN FROM table_name WHERE  
CONDITION ORDER BY column_name {ASC|DESC};
```

Select Statement Syntax

SQL GROUP BY CLAUSE

```
SELECT SUM(column_name) FROM table_name WHERE CONDITION  
GROUP BY column_name;
```

SQL COUNT CLAUSE

```
SELECT COUNT(column_name) FROM table_name WHERE CONDITION;
```

SQL HAVING CLAUSE

```
SELECT SUM(column_name) FROM table_name WHERE CONDITION  
GROUP BY column_name HAVING (arithmeticfunction condition);
```

Create and Drop Index Syntax

- **SQL CREATE INDEX Statement :**

```
CREATE UNIQUE INDEX index_name ON table_name( column1,  
column2,...columnN);
```

- **SQL DROP INDEX STATEMENT**

```
ALTER TABLE table_name DROP INDEX index_name;
```

- **SQL DESC Statement :**

```
DESC table_name;
```

Commit and Rollback Syntax

SQL COMMIT STATEMENT

COMMIT;

SQL ROLLBACK STATEMENT

ROLLBACK;

Inner Join Syntax

- The most frequently used and important of the joins is the **INNER JOIN**. They are also referred to as an **EQUIJOIN**.
- The **INNER JOIN** creates a new result table by combining column values of two tables (table1 and table2) based upon the join-predicate. The query compares each row of table1 with each row of table2 to find all pairs of rows which satisfy the join-predicate. When the join-predicate is satisfied, column values for each matched pair of rows of A and B are combined into a result row.

SYNTAX:

- The basic syntax of **INNER JOIN** is as follows:

```
SELECT table1.column1, table2.column2...FROM table1INNER JOIN table2ON  
table1.common_filed = table2.common_field;
```

Left Join Syntax

- The SQL **LEFT JOIN** returns all rows from the left table, even if there are no matches in the right table. This means that if the ON clause matches 0 (zero) records in right table, the join will still return a row in the result, but with NULL in each column from right table.

- This means that a left join returns all the values from the left table, plus matched values from the right table or NULL in case of no matching join predicate.

SYNTAX:

- The basic syntax of **LEFT JOIN** is as follows:

```
SELECT table1.column1, table2.column2...FROM table1LEFT JOIN table2ON  
table1.common_field = table2.common_field;
```

Right Join Syntax

- The SQL **RIGHT JOIN** returns all rows from the right table, even if there are no matches in the left table. This means that if the ON clause matches 0 (zero) records in left table, the join will still return a row in the result, but with NULL in each column from left table.

- This means that a right join returns all the values from the right table, plus matched values from the left table or NULL in case of no matching join predicate.

SYNTAX:

- The basic syntax of **RIGHT JOIN** is as follows:

```
SELECT table1.column1, table2.column2...FROM table1RIGHT JOIN table2ON  
table1.common_field = table2.common_field;
```

Full Join Syntax

- The SQL **FULL JOIN** combines the results of both left and right outer joins.
 - The joined table will contain all records from both tables, and fill in NULLs for missing matches on either side.
- SYNTAX:**
- The basic syntax of **FULL JOIN** is as follows:

```
SELECT table1.column1, table2.column2...FROM table1FULL JOIN table2ON  
table1.common_filed = table2.common_field;
```

Module - 4 [Automation Core Testing]

Introduction

- Test automation is the use of software to control the execution of tests, the comparison of actual outcomes to predicted outcomes, the setting up of test preconditions, and other test control and test reporting functions.
- Commonly, test automation involves automating a manual process already in place that uses a formalized testing process.
- Although manual tests may find many defects in a software application, it is a laborious and time consuming process.
- In addition it may not be effective in finding certain classes of defects.
- Test automation is a process of writing a computer program to do testing that would otherwise need to be done manually.
- Once tests have been automated, they can be run quickly.
- This is often the most cost effective method for software products that have a long maintenance life, because even minor patches over the lifetime of the application can cause features to break which were working at an earlier point in time.

Agenda

- Fundamental of Automation

- Testing

- Automation Framework

- Load Runner Up

- Selenium IDE

Fundamental of Automation Testing

Introduction

- Test automation is the use of software to control the execution of tests, the comparison of actual outcomes to predicted outcomes, the setting up of test preconditions, and other test control and test reporting functions.
- Commonly, test automation involves automating a manual process already in place that uses a formalized testing process.
- Although manual tests may find many defects in a software application, it is a laborious and time consuming process.
- In addition it may not be effective in finding certain classes of defects.
- Test automation is a process of writing a computer program to do testing that would otherwise need to be done manually.
- Once tests have been automated, they can be run quickly.
- This is often the most cost effective method for software products that have a long maintenance life, because even minor patches over the lifetime of the application can cause features to break which were working at an earlier point in time.

Approach to Test Automation

- **Code-driven testing:** The public (usually) interfaces to classes, modules, or libraries are tested with a variety of input arguments to validate that the results that are returned are correct.

- **Graphical user interface testing:** A testing framework generates user interface events such as keystrokes and mouse clicks, and observes the changes that result in the user interface, to validate that the observable behavior of the program is correct.

Factors of Automated Test Tools

- Examine your current testing process and determine where it needs to be adjusted for using automated test tools.
- Be prepared to make changes in the current ways you perform testing.
- Involve people who will be using the tool to help design the automated testing process.
- Create a set of evaluation criteria for functions that you will want to consider when using the automated test tool. These criteria may include the following:
 - Test repeatability
 - Criticality/risk of applications
 - Operational simplicity
 - Ease of automation
 - Level of documentation of the function (requirements, etc.)
- Examine your existing set of test cases and test scripts to see which ones are most applicable for test automation.
- Train people in basic test-planning skills.

Challenges of Test Automation

- Buying the Wrong Tool
- Inadequate Test Team Organization
- Lack of Management Support
- Incomplete Coverage of Test Types by the selected tool
- Inadequate Tool Training
- Difficulty using the tool
- Lack of a Basic Test Process or Understanding of What to Test
- Lack of Configuration Management Processes
- Lack of Tool Compatibility and Interoperability
- Lack of Tool Availability

Why Automation?

- **Speed** – Automation Scripts run very fast when compared to human users
- **Reliable** – Tests perform precisely the same operations each time they are run, thereby eliminating human error.
- **Repeatable** – We can test how the application reacts after repeated execution of the same operation
- **Comprehensive** – We can build a suite of tests that covers every feature in our application
- **Reusable** – We can reuse tests on different versions of an application, even if the user interface changes.
- **Automate your testing procedure when you have lot of regression work.**
- **Automate your load testing work for creating virtual users to check load capacity of your application.**
- **Automate your testing work when your GUI is almost frozen but you have lot of frequently functional changes.**

When to Automate?

Which Software Testing should be Automated?

- Test that need to be Execute of every build of the Application

- Test that use Multiple Data Value (Retesting & Regression Testing)

- Test that Required data From Application intimates(G.U.I. Attributes)

- Load and stress Testing

Which Software should not be Automate?

- Usability testing one time testing

- Quick look Tester as soon as possible testing Ad-hoc testing /Random testing

- Customer Requirement are frequently changing

Why Use Automated Testing Tools?

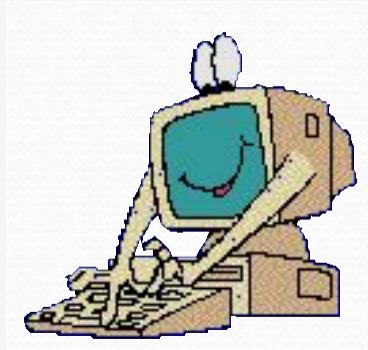


No Testing
Testing



Manual Testing

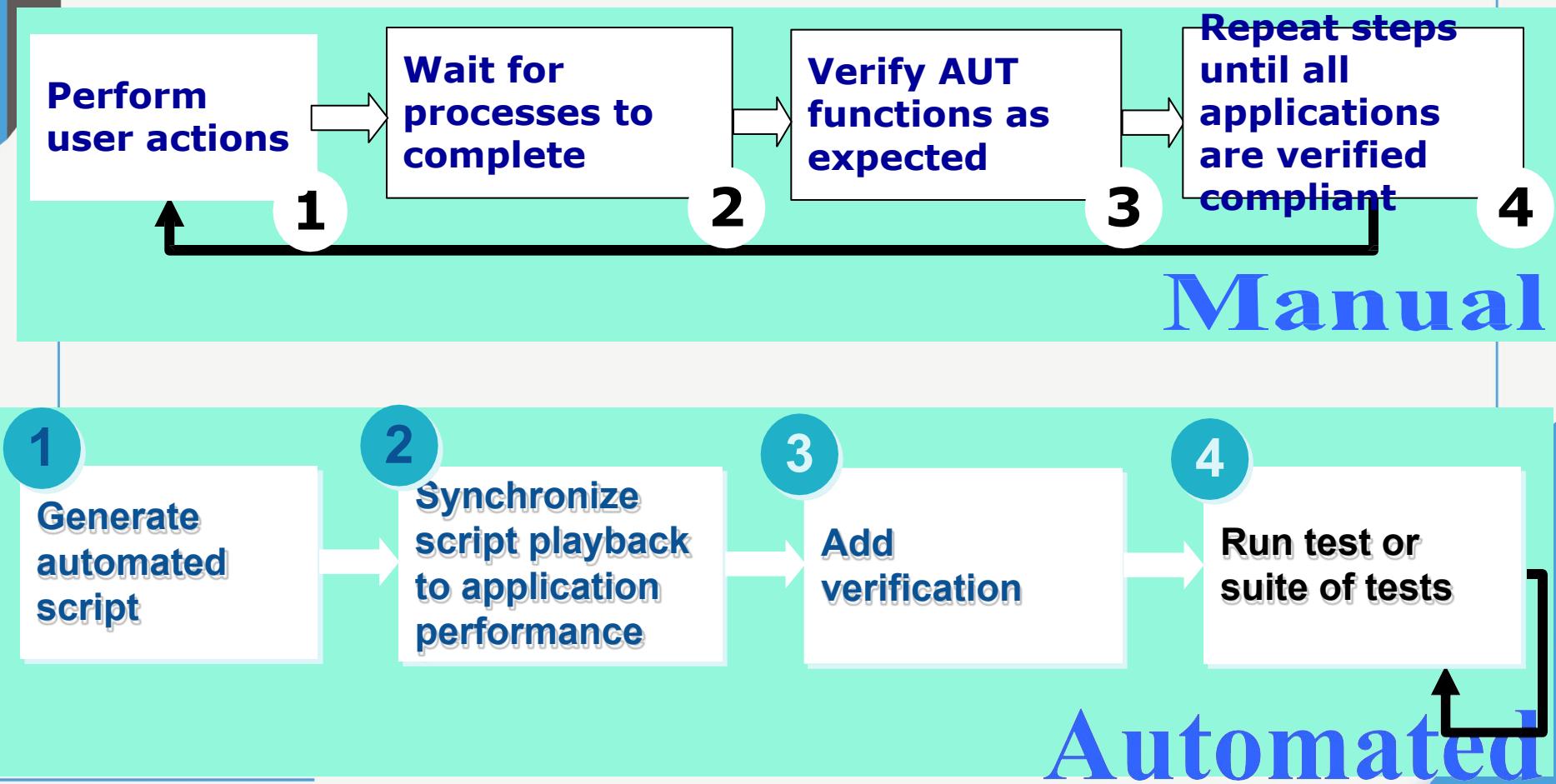
- **Time consuming**
- **Low reliability**
- **Human resources**
- **Inconsistent**



Automated Testing Testing Speed

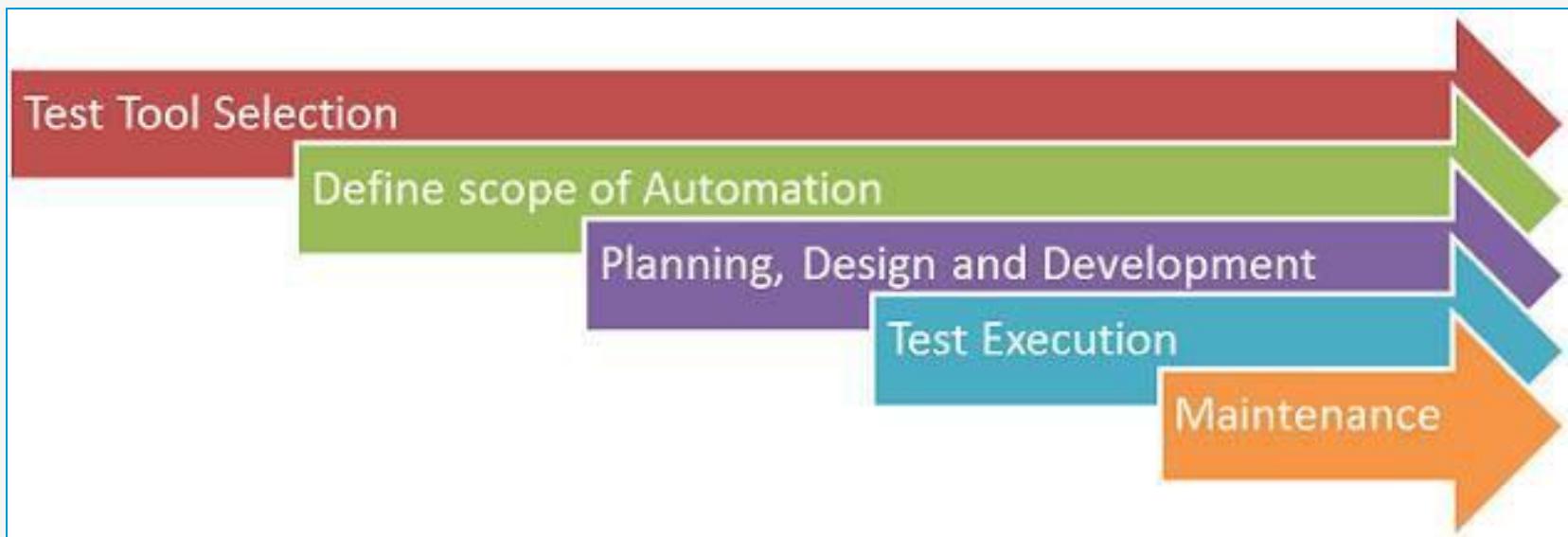
- **Repeatability**
- **Programming capabilities**
- **Coverage**
- **Reliability**
- **Reusability**

Manual To Automated Testing



Automation Process

- Test Tool Selection
- Define the Scope of Automation
- Planning, Designing and Development
- Test Estimation
- Maintenance



Framework in Automation

A **framework is set of automation guidelines** which help in

- Maintaining consistency of Testing
- Improves test structuring
- Minimum usage of code
- Less Maintenance of code
- Improve re-usability
- Non-Technical testers can be involved in code
- Training period of using the tool can be reduced
- Involves Data wherever appropriate

There are four types of framework used in software automation testing:

- Data Driven Automation Framework
- Keyword Driven Automation Framework
- Modular Automation Framework
- Hybrid Automation Framework

Benefits of Automation Testing

- 70% faster than the manual testing
- Wider test coverage of application features
- Reliable in results
- Ensure Consistency
- Saves Time and Cost
- Improves accuracy
- Human Intervention is not required while execution
- Increases Efficiency
- Better speed in executing tests
- Re-usable test scripts
- Test Frequently and thoroughly
- More cycle of execution can be achieved through automation
- Early time to market

Automation Framework

What is Framework?

- A **test automation framework** is a set of assumptions, concepts and tools that provide support for automated software testing.
- The main advantage of such a framework is the low cost for maintenance.
- If there is change to any test case then only the test case file needs to be updated and the Driver Script and Startup script will remain the same.
- Ideally, there is no need to update the scripts in case of changes to the application.
- Choosing the right framework/scripting technique helps in maintaining lower costs.
- The costs associated with test scripting are due to development and maintenance efforts.
- The approach of scripting used during test automation has effect on costs.

What is Framework?

Instead of providing a bookish definition of a framework, let's consider an example:

- I am sure you have attended a seminar / lecture / conference where the participants was asked to observe the following guidelines -
- Participants should occupy their seat 5 minutes before start of lecture
- Bring along a notebook and pen for note taking.
- Read the abstract so you have an idea of what the presentation will be about.
- Mobile Phones should be set on silent
- Use the exit gates at opposite end to the speaker should you require to leave in middle of the lecture.

Load Runner up by HP(LR)

Agenda

- Why Performance ?
- Definitions: Performance Testing
- Benchmark Design
- Performance Testing Tools
- Load Runner Components
- What is load testing process?
- Getting Familiar with Mercury Tours
- Application Requirements

Some Testing Definitions

- **Stress Testing:** Stress Testing is done in order to check when the application fails by reducing the system resources such as RAM, HDD etc. and keeping the number of users as constant.

- **Load Testing:** Load Testing is done in order to check when the application fails by increasing the number of users and keeping the system resources as constant.

- **Performance Testing:** The term performance can mean measuring response time, throughput resource utilization, or some other system characteristic(or group them) by varying the number of users.

Benchmark Design

- The Benchmark is the representative workload used during the performance test run. It should be representative of the likely real-world operating conditions.

- Benchmark is provided by the client.

- In Industry scenario the benchmark is as follows:

- No. of transactions passed per second ≥ 8
- Response time ≤ 5 sec.

Performance Testing Tools

- Segue Silk Performer
- Rational Team Test
- Mercury Load Runner
- Empirix e-Load/RSW
- Soft Light Tools Loader

Selenium IDE

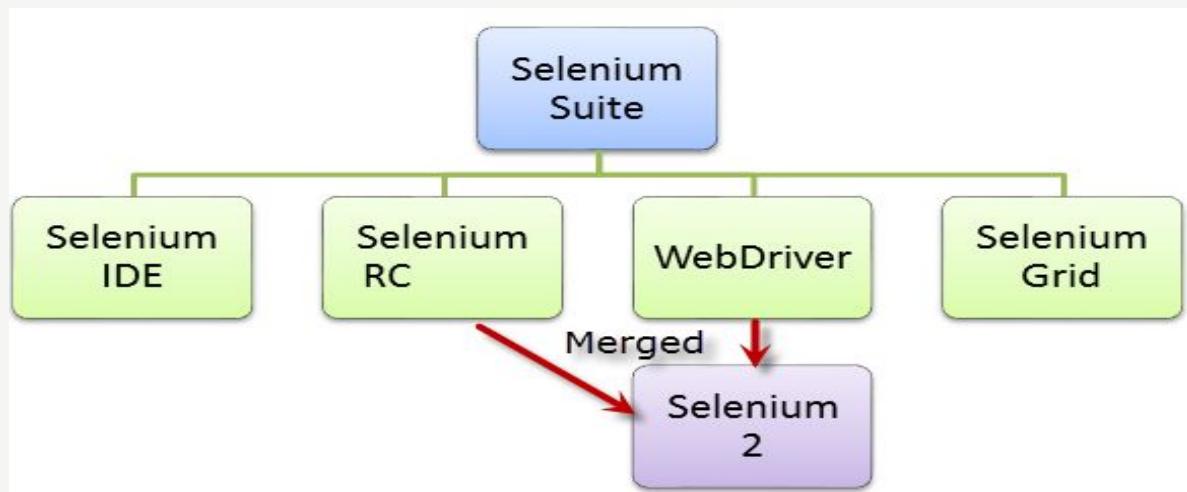
What is Selenium?

- Selenium is a free (open source) automated testing suite for web applications across different browsers and platforms.

- It is quite similar to HP Quick Test Pro (QTP) only that Selenium focuses on automating web-based applications.

- Selenium is not just a single tool but a suite of software's, each catering to different testing needs of an organization. It has four components.

- Selenium Integrated Development Environment (IDE)
- Selenium Remote Control (RC)
- Web Driver
- Selenium Grid



Who developed Selenium?

Since Selenium is a collection of different tools, it had different developers as well. Below are the key persons who made notable contributions to the Selenium Project

- Primarily, Selenium was created by Jason Huggins in 2004.

- An engineer at **ThoughtWorks**, he was working on web application that required frequent testing.

- Having realized that the repetitious manual testing of their application was becoming more and more inefficient, he created a JavaScript program that would automatically control the browser's actions.

- He named this program as "**JavaScriptTestRunner**".



- Seeing potential in this idea to help automate other web applications, he made **JavaScriptRunner** open-source which was later re-named as **Selenium Core**.

Selenium RC

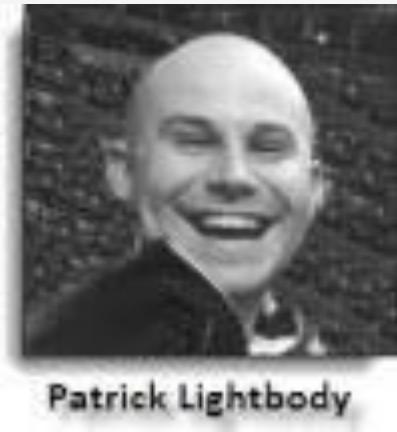
Unfortunately; testers using Selenium Core had to install the whole application under test and the web server on their own local computers because of the restrictions imposed by the same origin policy. So another Thought Work's engineer, Paul Hammant , decided to create a server that will act as an HTTP proxy to “trick” the browser into believing that Selenium Core and the web application being tested come from the same domain. This system became known as the Selenium Remote Control or Selenium 1.



Paul Hammant

Selenium Grid

Selenium Grid was developed by Patrick Lightbody to address the need of minimizing test execution times as much as possible. He initially called the system “Hosted QA.” It was capable of capturing browser screenshots during significant stages, and also of sending out Selenium commands to different machines simultaneously.



Selenium IDE

Shinya Kasatani of Japan created Selenium IDE, a Firefox extension that can automate the browser through a record-and-playback feature. He came up with this idea to further increase the speed in creating test cases. He donated Selenium IDE to the Selenium Project in 2006.



Selenium Web Driver

Simon Stewart created WebDriver circa 2006 when browsers and web applications were becoming more powerful and more restrictive with JavaScript programs like Selenium Core. It was the first cross-platform testing framework that could control the browser from the OS level.



Birth of Selenium 2

In 2008, the whole Selenium Team decided to merge WebDriver and Selenium RC to form a more powerful tool called Selenium 2, with WebDriver being the core. Currently, Selenium RC is still being developed but only in maintenance mode. Most of the Selenium Project's efforts are now focused on Selenium 2.

So, Why the Name Selenium?

It came from a joke which Jason cracked one time to his team. Another automated testing framework was popular during Selenium's development, and it was by the company called Mercury Interactive (yes, the company who originally made QTP before it was acquired by HP). Since Selenium is a well-known antidote for Mercury poisoning, Jason suggested that name. His teammates took it, and so that is how we got to call this framework up to the present.

Introduction Selenium IDE

- Selenium Integrated Development Environment (IDE) is the simplest framework in the Selenium suite and is the easiest one to learn.
- It is a Firefox plugin that you can install as easily as you can with other plugins. However, because of its simplicity, Selenium IDE should only be used as a prototyping tool.
- If you want to create more advanced test cases, you will need to use either Selenium RC or WebDriver.

Pros & Cons Selenium IDE

Pros

- Very easy to use and install.
- No programming experience is required, though knowledge of HTML and DOM are needed
- Can export tests to formats usable in Selenium RC and WebDriver
- Has built-in help and test results reporting module.
- Provides support for extensions.

Cons

- Available only in Fire fox
- Designed only to create prototypes of tests.
- No support for iteration and conditional operations
- Test execution is slow compared to that of Selenium RC and WebDriver.

Introduction Selenium RC

- Selenium RC was the flagship testing framework of the whole Selenium project for a long time.

- This is the first automated web testing tool that allowed users to use a programming language they prefer.

- As of version 2.25.0, RC can support the following programming languages:

- Java
- C#
- PHP
- Python
- Perl
- Ruby

Pros & Cons Selenium RC

Pros

- Cross browser and cross platform
- Can perform looping and conditional operations
- Can support data-driven testing
- Has matured and complete API
- Can readily support new browsers
- Faster execution than IDE

Cons

- Installation is more complicated than IDE
- Must have programming knowledge
- Needs Selenium RC Server to be running
- API contains redundant and confusing commands
- Browser interaction is less realistic
- Inconsistent result & Users JavaScript
- Slower execution time than WebDriver.

Introduction Selenium Driver

- The WebDriver proves itself to be better than both Selenium IDE and Selenium RC in many aspects.
- It implements a more modern and stable approach in automating the browser's actions. WebDriver, unlike Selenium RC, does not rely on JavaScript for automation. It controls the browser by directly communicating to it.
- The supported languages are the same as those in Selenium RC.
 - Java
 - C#
 - PHP
 - Python
 - Perl
 - Ruby

Pros & Cons Selenium Driver

Pros

- Simpler installation than Selenium RC
- Communicates directly to the browser
- Browser integration is more realistic.
- No need for a separate component such as the RC Server
- Faster execution time than IDE and RC

Cons

- Installation is more complicated than Selenium IDE
- Requires programming knowledge
- Cannot readily support new browser
- Has no built-in mechanism for logging runtime message and generating test results

Introduction Selenium Driver

- Selenium Grid is a tool used together with Selenium RC to run parallel tests across different machines and different browsers all at the same time. Parallel execution means running multiple tests at once.

- Features:

- Enables simultaneous running of tests in multiple browsers and environments.
- Saves time enormously.
- Utilizes the hub-and-nodes concept. The hub acts as a central source of Selenium commands to each node connected to it.

Features of Selenium Driver

- Enables simultaneous running of tests in multiple browsers and environments.
- Saves time enormously.
- Utilizes the hub-and-nodes concept. The hub acts as a central source of Selenium commands to each node connected to it.

Advantages of Selenium over QTP

Selenium	QTP
Open source, free to use, and free of charge.	Commercial.
Highly extensible	Limited add-ons
Can run tests across different browsers	Can only run tests in Firefox , Internet Explorer and Chrome
Supports various operating systems	Can only be used in Windows
Supports mobile devices	Supports mobile devise using 3 rd party software
Can execute tests while the browser is minimized to be visible on the desktop	Needs to have the application under test
Can execute tests in parallel.	Can only execute in parallel but using Quality Center which is again a paid product.

Automation Testing

SELENIUM

Agenda

Module-1 [Introduction to Automation Testing]

Module -2 [Core Java]

Module-3 [Selenium Web Driver]

Module-4 [Automation Testing Framework]

Module -5 [Maven Project Management Tool]

Module-1 [Introduction to Automation Testing]

- Introduction to Automation Testing
- Selenium
- QTP V/S Selenium

Automation Testing

Introduction

- Test automation is the use of software to control the execution of tests, the comparison of actual outcomes to predicted outcomes, the setting up of test preconditions, and other test control and test reporting functions.
- Commonly, test automation involves automating a manual process already in place that uses a formalized testing process.
- Although manual tests may find many defects in a software application, it is a laborious and time consuming process.
- In addition it may not be effective in finding certain classes of defects.
- Test automation is a process of writing a computer program to do testing that would otherwise need to be done manually.
- Once tests have been automated, they can be run quickly.
- This is often the most cost effective method for software products that have a long maintenance life, because even minor patches over the lifetime of the application can cause features to break which were working at an earlier point in time.

Automation Testing

Why Automation?

- **Speed**—Automation Scripts run very fast when compared to human users
- **Reliable**—Tests perform precisely the same operations each time they are run, thereby eliminating human error.
- **Repeatable**—We can test how the application reacts after repeated execution of the same operation
- **Comprehensive**—We can build a suite of tests that covers every feature in our application
- **Reusable**—We can reuse tests on different versions of an application, even if the user interface changes.
- Automate your testing procedure when you have lot of regression work.
- Automate your load testing work for creating virtual users to check load capacity of your application.

Difference between Manual and Automation Testing

Manual Testing

- Manual testing is not accurate at all times due to human error, hence it is less reliable.
- Manual testing is time-consuming, taking up human resources.
- Investment is required for human resources.
- Manual testing is only practical when the test cases are run once or twice, and frequent repetition is not required.
- Manual testing allows for human observation, which may be more useful if the goal is user-friendliness or improved customer experience.

Automated Testing

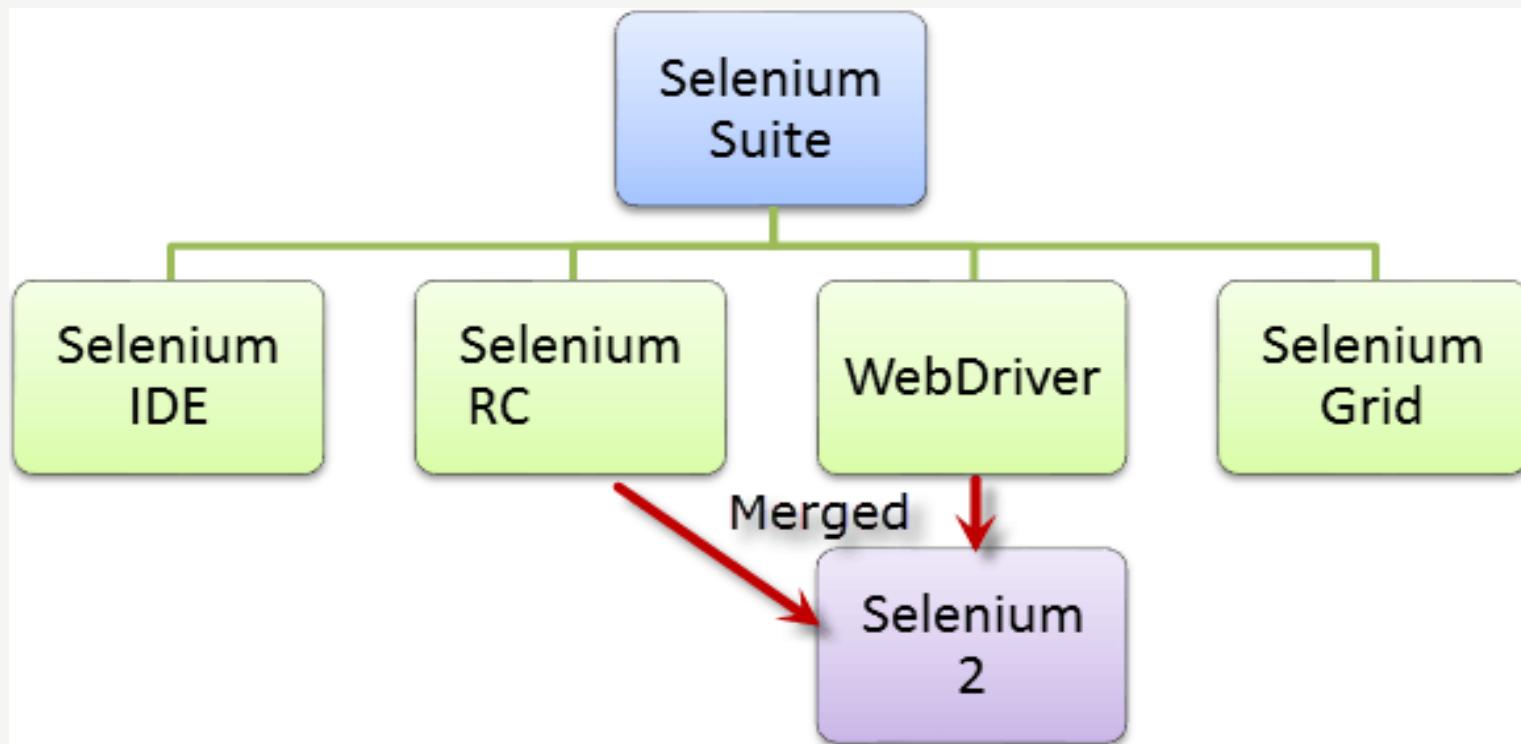
- Automated testing is more reliable, as it is performed by tools and/or scripts.
- Automated testing is executed by software tools, so it is significantly faster than a manual approach.
- Investment is required for testing tools.
- Automated testing is a practical option when the test cases are run repeatedly over a long time period.
- Automated testing does not entail human observation and cannot guarantee user-friendliness or positive customer experience.

Introduction to Selenium

- Selenium is a free (open source) automated testing suite for web applications across different browsers and platforms.
- It is quite similar to HP Quick Test Pro (QTP) only that Selenium focuses on automating web-based applications.

Introduction to Selenium

Selenium is not just a single tool but a suite of software's, each catering to different testing needs of an organisation. It has four components.



Selenium Components

Introduction Selenium IDE

- Selenium Integrated Development Environment (IDE) is the simplest framework in the Selenium suite and is the easiest one to learn.
- It is a Firefox plugin that you can install as easily as you can with other plugins.
- However, because of its simplicity, Selenium IDE should only be used as a prototyping tool.
- If you want to create more advanced test cases, you will need to use either Selenium RC or WebDriver.

Selenium (Pros and Cons)



PROS

- Very easy to use and install.
- No programming experience is required, though knowledge of HTML and DOM are needed.
- Can export tests to formats usable in Selenium RC and WebDriver.
- Has built-in help and test results reporting module.
- Provides support for extensions.

CONS

- Available only in Firefox.
- Designed only to create prototypes of tests.
- No support for iteration and conditional operations.
- Test execution is slow compared to that of Selenium RC and WebDriver.

Introduction Selenium Remote Control (Selenium RC)

Selenium RC was the **flagship testing framework** of the whole Selenium project for a long time. This is the first automated web testing tool that **allowed users to use a programming language they prefer**. As of version 2.25.0, RC can support the following programming languages:

Java

C#

PHP

Python

Perl

Ruby

Selenium RC(Pros and Cons)

Pros

- Cross-browser and cross-platform
- Can perform looping and conditional operations
- Can support data-driven testing
- Has matured and complete API
- Can readily support new browsers
- Faster execution than IDE

Cons

- Installation is more complicated than IDE
- Must have programming knowledge
- Needs Selenium RC Server to be running
- API contains redundant and confusing commands
- Browser interaction is less realistic
- Inconsistent results & Uses Javascript
- Slower execution time than WebDriver

Introduction WebDriver

The WebDriver proves itself to be **better than both Selenium IDE and Selenium RC** in many aspects. It implements a more modern and stable approach in automating the browser's actions. WebDriver, unlike Selenium RC, does not rely on JavaScript for automation. **It controls the browser by directly communicating to it.**

The supported languages are the same as those in Selenium RC.

Java

C#

PHP

Python

Perl

Ruby

WebDriver(Pros and Cons)

Pros

Simpler installation than Selenium RC

Communicates directly to the browser

Browser interaction is more realistic

No need for a separate component such as the RC Server

Faster execution time than IDE and RC

Cons

Installation is more complicated than Selenium IDE

Requires programming knowledge

Cannot readily support new browsers

Has no built-in mechanism for logging runtime messages and generating test results

Selenium Grid

- Selenium Grid is a tool **used together with Selenium RC** to run **parallel tests** across different machines and different browsers all at the same time. Parallel execution means running multiple tests at once.

Features:

- Enables **simultaneous running of tests in multiple browsers and environments**.
- **Saves time** enormously.
- Utilizes the **hub-and-nodes** concept. The hub acts as a central source of Selenium commands to each node connected to it.

Browser and Environment Support

	Selenium IDE	Selenium RC	WebDriver
Browser Support	Mozilla Firefox	Mozilla Firefox Internet Explorer Google Chrome Safari Opera Konqueror Others	Internet Explorer versions 6 to 9, both 32 and 64-bit Firefox 3.0, 3.5, 3.6, 4.0, 5.0, 6, 7 and above (current version is 16.0.1)

Browser and Environment Support

Selenium IDE	Selenium RC	WebDriver
		<p>Google Chrome 12.0.712.0 and above (current version is 22.0.1229.94 m)</p> <p>Opera 11.5 and above (current version is 12.02)</p> <p>Android - 2.3 and above for phones and tablets (devices & emulators)</p> <p>iOS 3+ for phones (devices & emulators) and 3.2+ for tablets (devices & emulators)</p> <p>HtmlUnit 2.9 and above (current version is 2.10)</p>

Browser and Environment Support

	Selenium IDE	Selenium RC	WebDriver
Operating System	Windows Mac OS X Linux	Windows Mac OS X Linux Solaris	All operating systems where the browsers above can run.
Browser	Firefox, Internet Explorer, Chrome, Safari, Opera	Firefox, Internet Explorer, Chrome, Safari, Opera	Firefox, Internet Explorer, Chrome, Safari, Opera

Selenium Features

Open source	- Works on multiple browsers and multiple operating systems as compared to other tools in market.
Supports multiple platform and browser	- You can develop selenium code and make it run parallelly on multiple machines using different browsers.
Mobile testing	- Support for Android and Iphone Testing.
Simple	- Selenium IDE is a simple tool which comes as an addon in firefox and is easy to use. It has the record and run feature which is very strong.

Selenium Features

	<ul style="list-style-type: none">- You can also extend the functionality/scope of IDE with the help of many plugins available
	<ul style="list-style-type: none">- You can also create your own Selenium IDE plugins
	<p>Webdriver is the latest version of selenium and is very strong. Its removed lots of drawbacks in RC and introduced many more features in selenium.</p> <ul style="list-style-type: none">- Selenium when used with Hudson can be used for Continuous integration
	<ul style="list-style-type: none">- Object oriented datadriven or hybrid testing framework can be made very easily.
	<ul style="list-style-type: none">- You can use open source frameworks such as junit, testng, nuint etc and can write selenium test cases in them

QTP VS SELENIUM

Quick Test Professional(QTP) is a proprietary automated testing tool previously owned by the company **Mercury Interactive** before it was **acquired by Hewlett-Packard in 2006**. The Selenium Tool Suite has many advantages over QTP (as of version 11) as detailed below -

QTP VS SELENIUM

QTP	SELENIUM
Open source, free to use, and free of charge.	Commercial.
Highly extensible	Limited add-ons
Can run tests across different browsers	Can only run tests in Firefox, Internet Explorer and Chrome
Supports various operating systems	Can only be used in Windows
Supports mobile devices	Supports mobile device using 3rd party software
Can execute tests while the browser is minimized	Needs to have the application under test to be visible on the desktop
Can execute tests in parallel.	Can only execute in parallel but using Quality Center which is again a paid product.

Advantages of QTP over Selenium

QTP	SELENIUM
Can test both web and desktop applications	Can only test web applications
Comes with a built-in object repository	Has no built-in object repository
Automates faster than Selenium because it is a fully featured IDE.	Automates at a slower rate because it does not have a native IDE and only third party IDE can be used for development
Data-driven testing is easier to perform because it has built-in global and local data tables.	Data-driven testing is more cumbersome since you have to rely on the programming language's capabilities for setting values for your test data
Can access controls within the browser(such as the Favorites bar, Address bar, Back and Forward buttons, etc.)	Cannot access elements outside of the web application under test

Advantages of QTP over Selenium

QTP	SELENIUM
Provides professional customer support	No official user support is being offered.
Has native capability to export test data into external formats	Has no native capability to export runtime data onto external formats
Parameterization Support is in built	Parameterization can be done via programming but is difficult to implement.
Test Reports are generated automatically	No native support to generate test /bug reports.

Advantages of QTP over Selenium

Though clearly, QTP has more advanced capabilities, Selenium outweighs QTP in three main areas:

- **Cost** (because Selenium is completely free)
- **Flexibility** (because of a number of programming languages, browsers, and platforms it can support)
- **Parallel testing** (something that QTP is capable of but only with use of Quality Center)

Who uses Selenium

ThoughtWorks®

Google™



ATLASSIAN



eBay®



amazon.com®



emt



aqrис
thinks agile

Scope and Growth of Selenium Testing

- Nowadays, software quality assurance (QA) has become an integral part of each project.
- Many organizations even adopt test-based development approach to make the coding
 - compatible with testing modules.
- The QA professionals further use several test automation tools and frameworks to
 - facilitate test management.
- As a compact and robust software testing framework, Selenium is used widely to web
 - application.
- The open source framework was originally developed by a team of testers at ThoughtWorks in 2014.

Scope and Growth of Selenium Testing

- But within a decade, Selenium has become synonymous with internet application testing.
- A number of recent reports have highlighted that Selenium skills has become one of the most commonly and widely required test automation technology.
- As each enterprise has to optimize the look, feel and performance of a web application to deliver user experience, Selenium testing is expected to grow steadily over next few years.

Scope and Growth of Selenium Testing

Why Selenium will Grow Steadily Over Next Few Years?

- Open Source and Free
- Selenium was released by ThoughtWork under the Apache 2.0 license. So the testing framework is open source, and can be downloaded by users without paying in charges.

Scope and Growth of Selenium Testing

Support for Multiple Web Browsers and Operating Systems

Each internet application must run across several web browsers and operating systems to keep the users engaged. So QA professionals look for a set of tools that allows them to quickly evaluate the performance of the application on some of the widely used web browsers.

Scope and Growth of Selenium Testing

A Complete IDE for Internet Application Testing	<p>The Selenium IDE comes with a set of advanced tools that allows users to record, edit and debug tests without putting any extra effort. As the IDE was earlier known as Selenium Recorder.</p>
Client API for Several Programming Languages	<p>The Selenium IDE records scripts in a special test scripting language called Selenese. But the testers have option to write the test scripts in a variety of programming languages in addition to Selenese.</p> <p>The Selenium Client API supports a number of widely used programming languages like C#, Java, Perl, PHP, Ruby and Python.</p>

Scope and Growth of Selenium Testing

Comprehensive Documentation and Support	<p>While migrating to a new web application testing framework, the QA professionals require additional training. The tools provided by Selenium are well documented. So the QA professionals can refer to a complete documentation and use guide to understand how the tools function.</p>

Advantages of Selenium

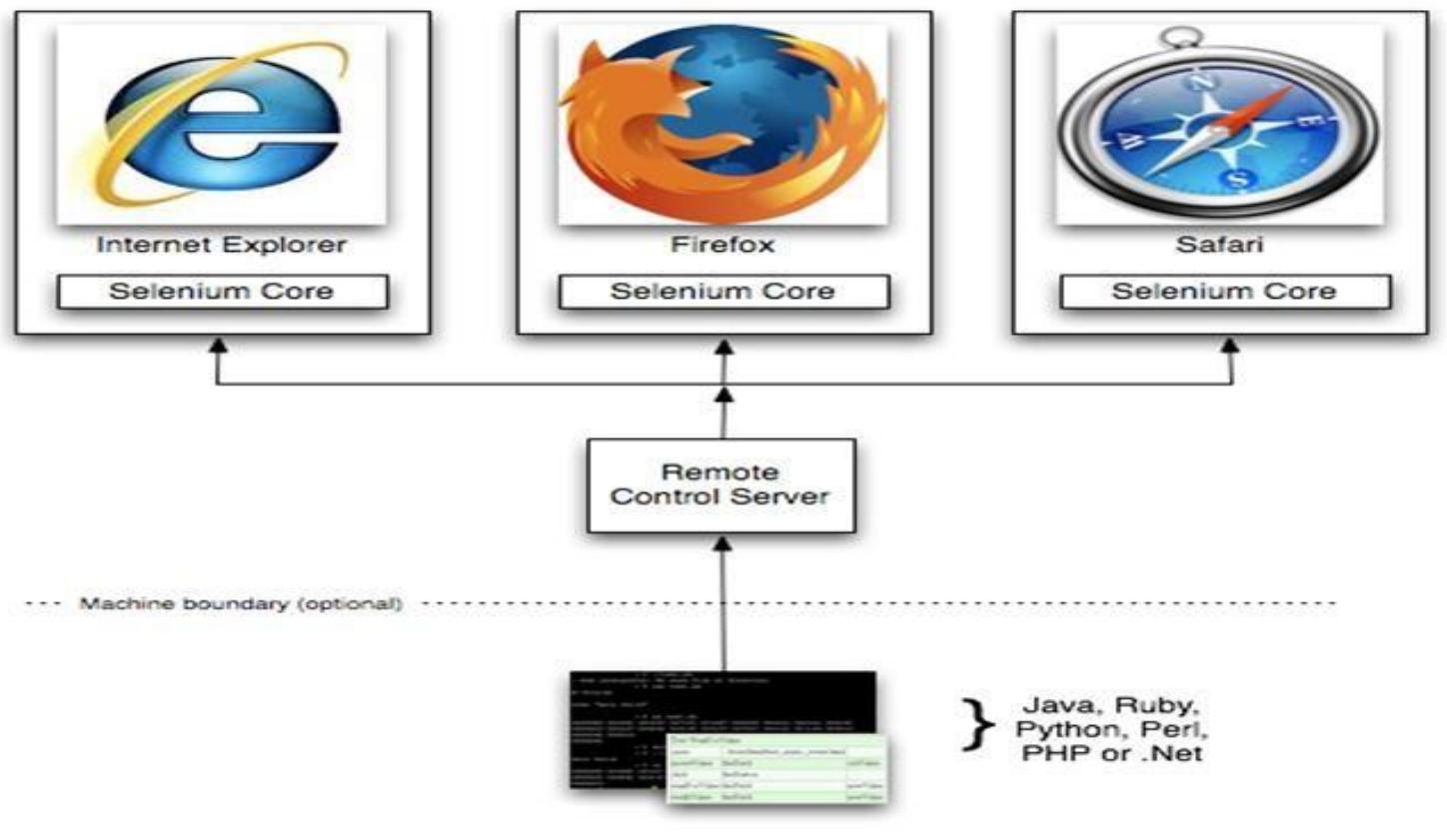
- Open Source
- Supports all browsers like IE, Firefox, Mozilla, Safari
- Supports all Operating Systems.
- Supports all programming languages Java,Ruby,C# and Python.
- Run multiple tests at a time.

Selenium RC Architecture

- Selenium Remote Control is a test tool that allows you to write automated web application UI tests in any programming language against any HTTP website using any mainstream JavaScript-enabled browser.
- Selenium RC comes in two parts:
 - A server which automatically launches and kills browsers, and acts as a HTTP proxy for web requests from them.
 - Client libraries for your favourite computer language.

Selenium RC Architecture

Below is a simplified architectural representation.



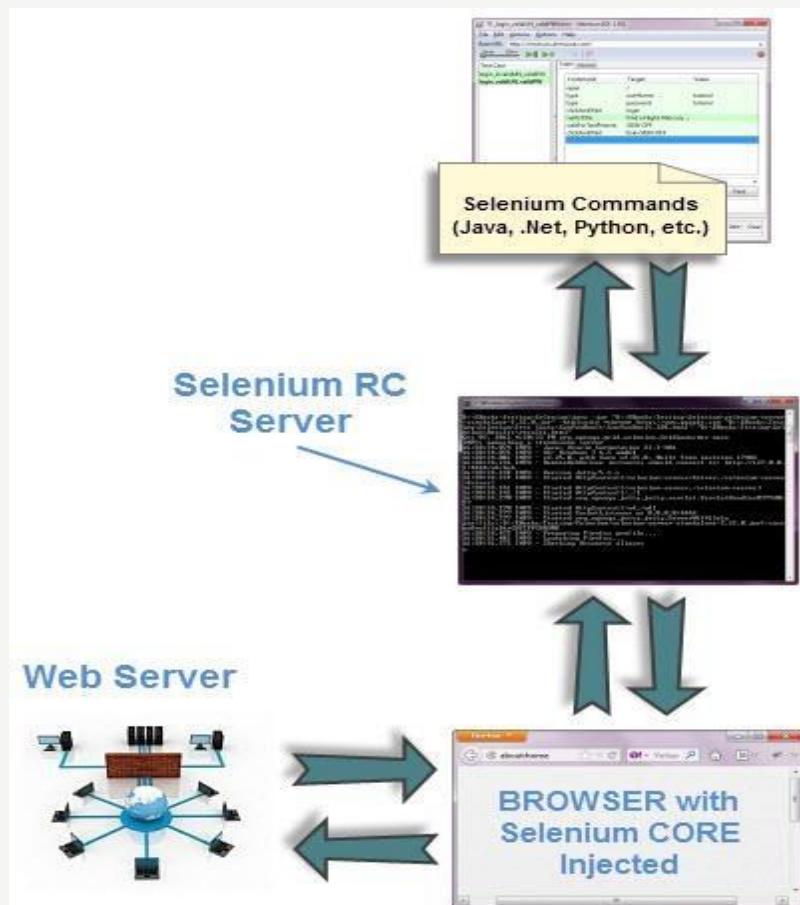
Selenium RC Architecture

Selenium RC's architecture is way more complicated.

- You first need to launch a separate application called Selenium Remote Control (RC) Server before you can start testing
- The Selenium RC Server acts as a "middleman" between your Selenium commands and your browser
- When you begin testing, Selenium RC Server "injects" a JavaScript program called Selenium Core into the browser.
- Once injected, Selenium Core will start receiving instructions relayed by the RC Server from your test program. When the instructions are received, Selenium Core will execute them as JavaScript commands.

Selenium RC Architecture

- The browser will obey the instructions of Selenium Core, and will relay its response to the RC Server.
- The RC Server will receive the response of the browser and then display the results to you.
- RC Server will fetch the next instruction from your test script to repeat the whole cycle.



Web driver Architecture

- WebDriver is a web automation framework that allows you to execute your tests
- against different browsers, not just Firefox (unlike Selenium IDE).
- WebDriver also enables you to use a programming language in creating your test scripts(not possible in Selenium IDE).
- You can now use conditional operations like if-then-else or switch-case.
- You can also perform looping like do-while.
- Following programming languages are supported by WebDriver: Java, .Net, PHP, Python, Perl, Ruby.

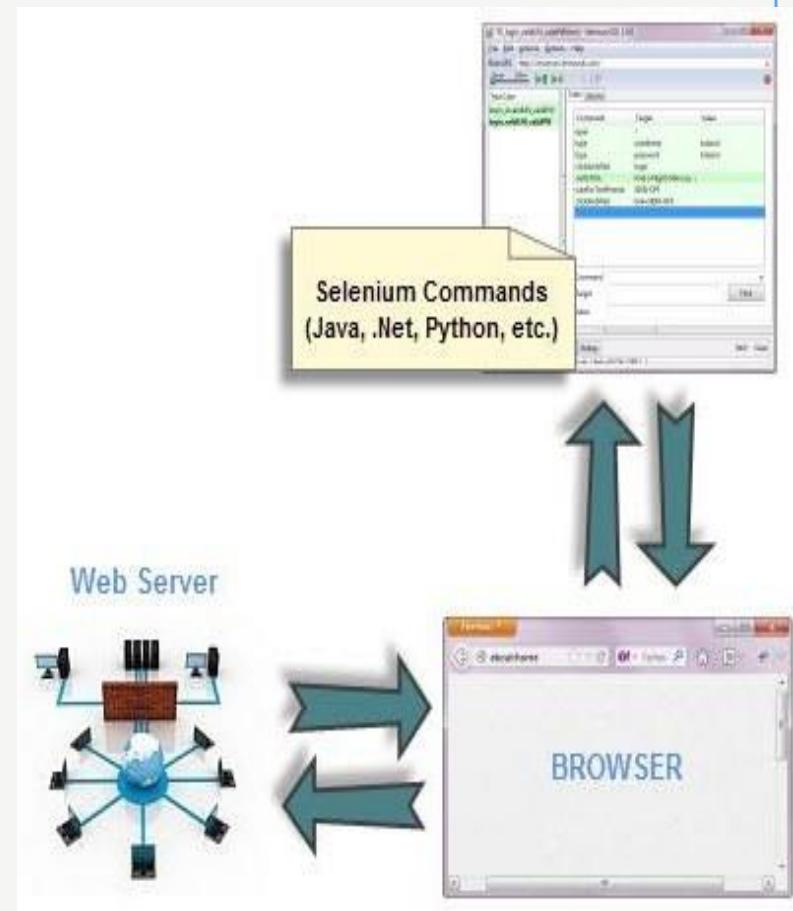
Web driver Architecture



Web driver Architecture

Web Driver's architecture is simpler than Selenium RC's.

- It controls the browser from the OS level
- All you need are your programming language's IDE (which contains your Selenium commands) and a browser.



Web driver Architecture

Web Driver	Selenium RC
<p>WebDriver is faster than Selenium RC since it speaks directly to the browser uses the browser's own engine to control it.</p>	<p>Selenium RC is slower since it uses a Javascript program called Selenium Core. This Selenium Core is the one that directly controls the browser, not you.</p>
<p>This is what happens in WebDriver</p> 	<p>Selenium RC Server</p>  <p>This is what happens in Selenium RC</p>

Web driver Architecture

Brief explanation about advantages of web driver

- Selenium uses JavaScript to automate web pages. This lets it interact very tightly with web content, and was one of the first automation tools to support Ajax and other heavily dynamic pages.
- However, this also means Selenium runs inside the JavaScript sandbox. This means you need to run the Selenium-RC server to get around the same-origin policy, which can sometimes cause issues with browser setup.
- WebDriver on the other hand uses native automation from each language. While this means it takes longer to support new browsers/languages, it does offer a much closer 'feel' to the browser.
- If you're happy with WebDriver, stick with it, it's the future. There are limitations and bugs right now, but if they're not stopping you, go for it.

Selenium Benefits Over WebDriver

- Supports many browsers and many languages, WebDriver needs native implementations for each new language/browser combo.
- Very mature and complete API
- Currently (Sept 2010) supports JavaScript alerts and confirms better

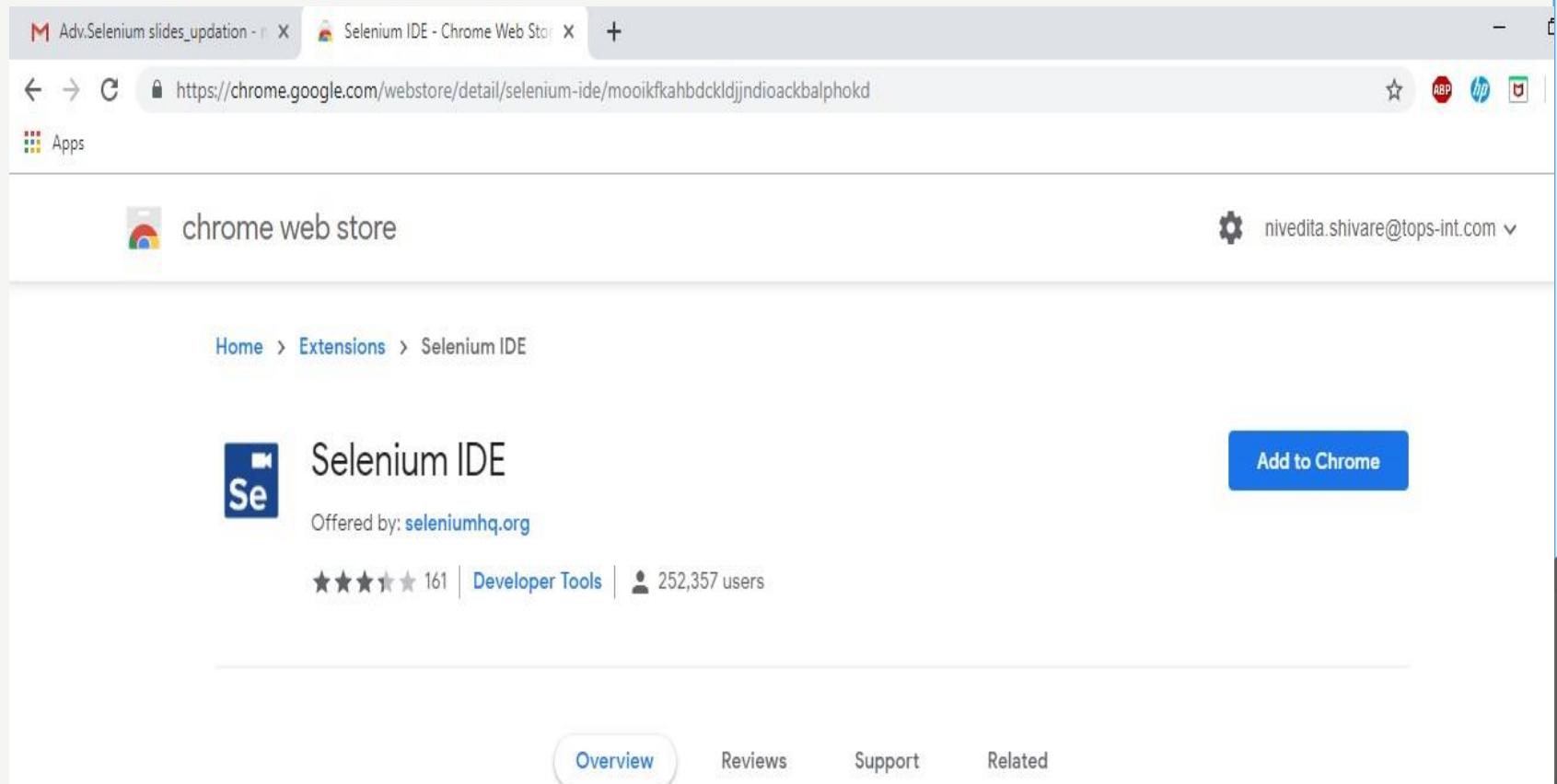
Benefits of WebDriver Compared to Selenium

- Native automation faster and a little less prone to error and browser configuration
- Does not Requires Selenium-RC Server to be running
- Access to headless HTML Unit can allow really fast tests
- Great API

Selenium IDE

- Download and Installation
- Record and playback techniques
- Modifying the script using IDE
- Object, CSS, XPath, Elements Identify Process
- Validate the locator value using IDE
- Commenting Code
- Fire Bug and Fire path

Add extension of Selenium IDE(Chrome)



The screenshot shows a Chrome browser window with the address bar displaying "Selenium IDE - Chrome Web Store". The URL is <https://chrome.google.com/webstore/detail/selenium-ide/mooikfahbdckldjjndioackbalphokd>. The page content is from the Chrome Web Store, showing the Selenium IDE extension. The extension icon features a blue square with white letters 'Se' and a video camera icon. The title is "Selenium IDE" and it is offered by "seleniumhq.org". It has a rating of 4.5 stars from 161 reviews, is categorized as "Developer Tools", and has 252,357 users. A prominent blue button on the right says "Add to Chrome". Below the main listing are navigation links for "Overview", "Reviews", "Support", and "Related".

Process #1: Recording a test script

Scenario

Open “<https://accounts.google.com>”.

Assert Title of the application

Enter a valid username and password and submit the details to login.

Verify that the user is re-directed to the Home page.

Step 1 – Launch the Firefox and open Selenium IDE from the menu bar.

Step 2 – Enter the address of application under test (“<https://accounts.google.com>”) inside the Base URL textbox.

Selenium IDE Commands

Each Selenium IDE test step can chiefly be split into following three components:

- Command
- Target
- Value

Command	Target	Value
type	id=Passwd	TestSelenium

Action needs to be performed The web element to interact with String that needs to be entered in the web element

Types of Selenium IDE commands:

There are three flavors of Selenium IDE commands. Each of the test step in Selenium IDE falls under any of the following category.

- Actions
- Accessors
- Assertions

Actions:

Actions are those commands which interact directly with the application by either altering its state or by pouring some test data.

For Example, “type” command lets the user to interact directly with the web elements like text box. It allows them to enter a specific value in the text box and as when the value is entered; it is showed on the UI as well.

Another example is “click” command. “click” command lets the user to manipulate with the state of the application.

In case of failure of an action type command, the test script execution halts and rest of the test steps would not be executed.

Accessors

Accessors are those commands which allows user to store certain values to a user defined variable. These stored values can be later on used to create assertions and verifications.

For example, “storeAllLinks” reads and stores all the hyperlinks available within a web page into a user defined variable. Remember the variable is of array type if there are multiple values to store.

Assertions

Similar to Accessors as they do not interact with the application directly. Assertions are used to verify the current state of the application with an expected state.

Forms of Assertions:

- **assert** – the “assert” command makes sure that the test execution is terminated in case of failure.
- **verify** – the “verify” command lets the Selenium IDE to carry on with the test script execution even if the verification is failed.
- **waitFor** – the “waitFor” command waits for a certain condition to be met before executing the next test step. The conditions are like page to be loaded, element to be present. It allows the test execution to proceed even if the condition is not met within the stipulated waiting period.

Commonly used Selenium IDE Commands

Command	Description	Arguments
open	Opens a specified URL in the browser.	1
assertTitle, VerifyTitle	Returns the current page title and compares it with the specified title	1
assertElementPresent, verifyElementPresent	Verify / Asserts the presence of an element on a web page.	1
type, typeKeys, sendKeys	Enters a value (String) in the specified web element.	2

Commonly used Selenium IDE Commands

Command	Description	Arguments
Click, clickAt, clickAndWait	Clicks on a specified web element within a web page.	1
waitForPageToLoad	Sleeps the execution and waits until the page is loaded completely.	1
waitForElement Present	Sleeps the execution and waits until the specified element is present	1
chooseOkOnNext Confirmation, chooseCancelOn NextConfirmation	Click on "OK" or "Cancel" button when next confirmation box appears.	0

Introduction to Firebug

Firebug is a Mozilla Firefox add-on. This tool helps us in identifying or to be more particular inspecting HTML, CSS and JavaScript elements on a web page. It helps us identifying the elements uniquely on a webpage.

The elements can be found uniquely based on their locator types which we would be discussing later...

How to Install Firebug?

For the ease of understanding, we would bifurcate the installation process into the following steps.

Step -1: Launch the Mozilla Firefox browser and navigate to this Firebug add-on download page. The URL takes us to Firefox add-ons section.

Step -2: Click on the “Add to Firefox” button present on the webpage. Refer the following figure for the same.

Installation of Firebug

For the ease of understanding, we would bifurcate the installation process into the following steps.

Step -1: Launch the Mozilla Firefox browser and navigate to this Firebug add-on download page. The URL takes us to Firefox add-ons section.

Step -2: Click on the “Add to Firefox” button present on the webpage. Refer the following figure for the same.

Creating Selenium Script using Firebug

Unlike Selenium IDE, In Firebug, we create automated test scripts manually by adding multiple test steps to form a logical and consistent test script.

Let us follow a progressive approach and understand the process step by step.

Scenario:

Open “<https://accounts.google.com>”.

Assert Title of the application

Enter an invalid username and invalid password and submit the details to login.

Module – 2 [Core Java]

- Introduction to Java
- Features of Java
- Data Type in Java
- Modifier in Java
- Operators
- Array
- OOPs
- File I/O
- Collection
- Sterilization
- Exception Handling

Core Java

Introduction:

- Java programming language was originally developed by Sun Microsystems.
- It is initiated by James Gosling and released in 1995 as core component of Sun Microsystems' Java platform (Java 1.0 [J2SE]).
- Sun Microsystems was acquired by the Oracle Corporation.
- Now it is part of Oracle.
- Java is guaranteed to be Write Once, Run Anywhere.
- Platforms available :
 - J2SE for Standard Edition
 - J2EE for Enterprise Applications,
 - J2ME for Mobile Applications.

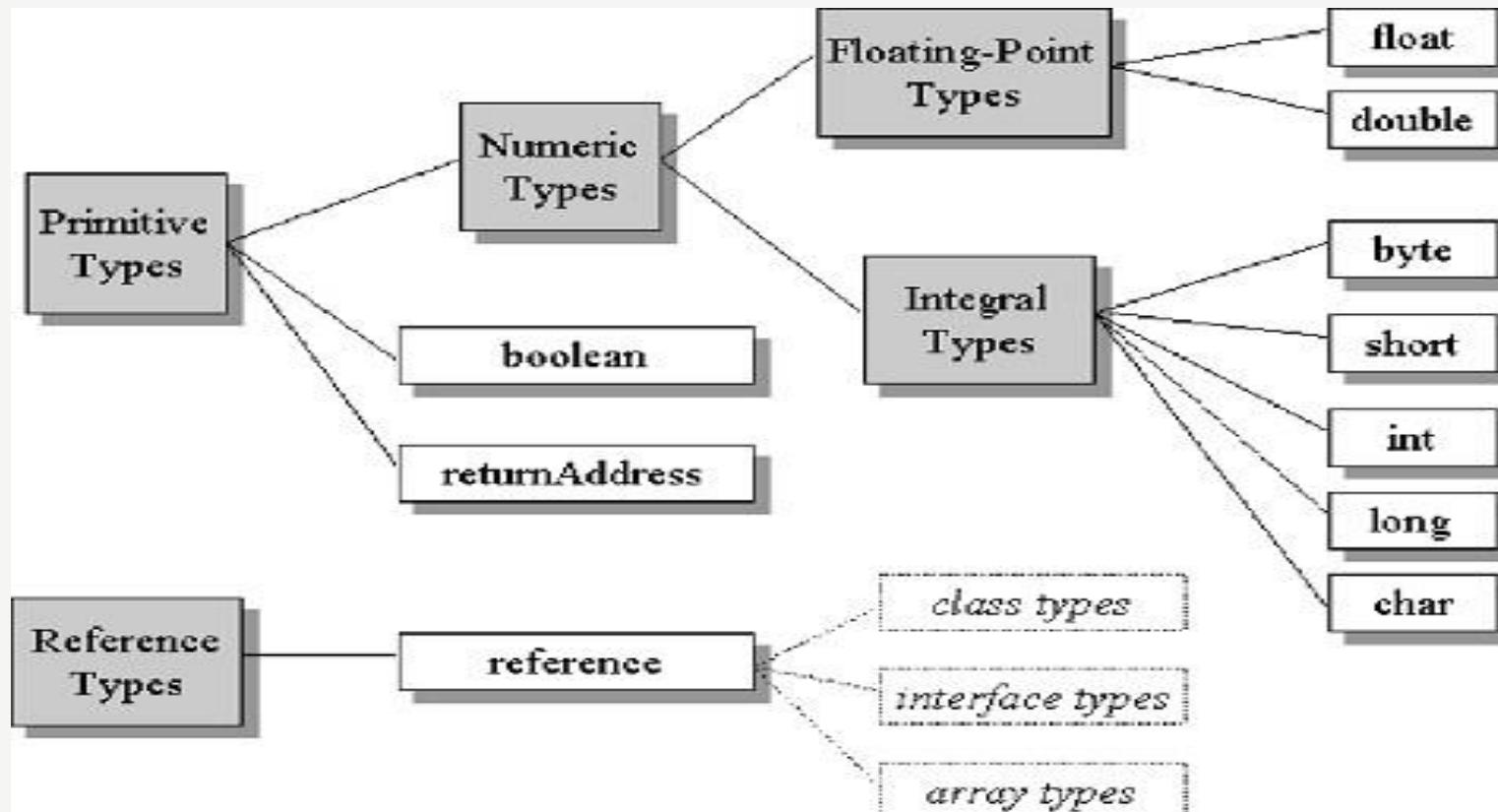
Features of Java

- Java is simple
- Java is object-oriented
- Java is distributed
- Java is interpreted
- Java is robust/powerful
- Java is secure
- Java is architecture-neutral
- Java is portable
- Java's performance
- Java is multithreaded
- Java is dynamic

Data types in Java

- Variables are nothing but reserved memory locations to store values.
- Based on the data type of a variable, the operating system allocates memory and decides what can be stored in the reserved memory.
- There are two data types available in Java:
 - Primitive Data Types
 - Reference/Object Data Types

Data types in Java



Data types in Java

Datatypes	Description
byte	8-bit signed two's complement integer. Range (-128 (- 2^7) to 127 (inclusive)($2^7 - 1$)). Default value is 0
Short	16-bit signed two's complement integer. Range -32,768 (- 2^{15}) to 32,767 (inclusive) ($2^{15} - 1$). Default value is 0.
Int	32-bit signed two's complement integer. Range -2,147,483,648. (2^{31}) to 2,147,483,647(inclusive). ($2^{31} - 1$). Default value is 0.
Long	64-bit signed two's complement integer. Range -9,223,372,036,854,775,808. (- 2^{63}) to 9,223,372,036,854,775,807 (inclusive). ($2^{63} - 1$). Default value is 0L.

Data types in Java

Datatypes	Description
Float	single-precision 32-bit IEEE 754 floating point. Default value is 0.0f.
Double	double-precision 64-bit IEEE 754 floating point. Default value is 0.0d.
Boolean	boolean data type represents one bit of information either true or false. Default value is false.
char	single 16-bit Unicode character. Range '\u0000' (or 0) to '\uffff' (or 65,535 inclusive).

Data types in Java

In Java a reference data type is a variable that can contain the reference or an address of dynamically created object. The reference data types are arrays, classes and interfaces.

- **Class type**

e.g.

```
public class Person
```

```
{ .....
```

```
Person p=new Person(); //p is of class type variable.
```

```
}
```

- **Array Type**

An array is a special kind of object that contains values called elements.

The java array enables the user to store the values of the same type in contiguous memory allocations.

e.g.

```
int [] a = new int [10];
```

```
String [] cities = , "Ahmedabad", "Mumbai", "Pune" -;
```

Keywords in Java

Keywords or Reserved words used for some internal process or represent some predefined actions. These words are therefore not allowed to use as a variable names or objects. Doing this will result into a **compile time error**.

57 keywords

Keywords in Java				
abstract	default	if	private	this
assert	do	implements	protected	throw
boolean	double	import	public	throws
break	else	instanceof	return	transient
byte	enum	int	short	try
case	extends	interface	static	void
catch	final	long	strictfp	volatile
char	finally	native	super	while
class	float	new	switch	
continue	for	package	synchronized	

Modifiers in Java

Modifier	Same Class	Same Package	Within Subclass outside the package	World
public	Y	Y	Y	Y
protected	Y	Y	Y	N
no modifier	Y	Y	N	N
private	Y	N	N	N

Assignment Operator(=) in Java

- **The Simple Assignment Operator**

One of the most common operators that you'll encounter is the simple assignment operator "`=`". It assigns the value on its right to the operand on its left:

```
int cadence = 0;
```

```
int speed = 0;
```

```
int gear = 1;
```

This operator can also be used on objects to assign *object references*.

Operators in Java

Java provides a rich set of operators to manipulate variables. We can divide all the Java operators into the following groups:

- Arithmetic Operators
- Relational Operators
- Bitwise Operators
- Logical Operators
- Assignment Operators
- Misc Operators

The Arithmetic Operators:

Arithmetic operators are used in mathematical expressions in the same way that they are used in algebra. The following table lists the arithmetic operators:

Assume integer variable A holds 10 and variable B holds 20, then:

Arithmetic Operators in Java

Operator	Description	Example
+	Addition - Adds values on either side of the operator	A + B will give 30
-	Subtraction - Subtracts right hand operand from left hand operand	A - B will give -10
*	Multiplication - Multiplies values on either side of the operator	A * B will give 200
%	Modulus - Divides left hand operand by right hand operand and returns remainder	B % A will give 0
/	Division - Divides left hand operand by right hand operand	B / A will give 2
++	Increment - Increases the value of operand by 1	B++ gives 21
--	Decrement - Decreases the value of operand by 1	B-- gives 19

Relational Operators in Java

- There are following relational operators supported by Java language
- Assume variable A holds 10 and variable B holds 20, then:

Operator	Description	Example
<code>==</code>	Checks if the values of two operands are equal or not, if yes then condition becomes true.	$(A == B)$ is not true.
<code>!=</code>	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.	$(A != B)$ is true.
<code>></code>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	$(A > B)$ is not true.

Bitwise Operators in Java

- Java defines several bitwise operators, which can be applied to the integer types, long, int, short, char, and byte.
- Bitwise operator works on bits and performs bit-by-bit operation. Assume if a = 60; and b = 13; now in binary format they will be as follows:

a = 0011 1100

b = 0000 1101

a&b = 0000 1100

a|b = 0011 1101

a^b = 0011 0001

~a = 1100 0011

Logical Operators in Java

- The following table lists the logical operators:
- Assume Boolean variables A holds true and variable B holds false, then:

Operator	Description	Example
<code>&&</code>	Called Logical AND operator. If both the operands are non-zero, then the condition becomes true.	$(A \&\& B)$ is false.
<code> </code>	Called Logical OR Operator. If any of the two operands are non-zero, then the condition becomes true.	$(A B)$ is true.
<code>!</code>	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make it false.	$!(A \&\& B)$ is true.

Assignment Operators in Java

Operator	Description	Example
=	Simple assignment operator, Assigns values from right side operands to left side operand	$C = A + B$ will assign value of $A + B$ into C
+=	Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand	$C += A$ is equivalent to $C = C + A$
-=	Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand	$C -= A$ is equivalent to $C = C - A$
*=	Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result to left operand	$C *= A$ is equivalent to $C = C * A$

Assignment Operators in Java

Operator	Description	Example
<code>/=</code>	Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand	<code>C /= A</code> is equivalent to <code>C = C / A</code>
<code>%=</code>	Modulus AND assignment operator, It takes modulus using two operands and assign the result to left operand	<code>C %= A</code> is equivalent to <code>C = C % A</code>
<code><=></code>	Left shift AND assignment operator	<code>C <=> 2</code> is same as <code>C = C << 2</code>
<code>>=></code>	Right shift AND assignment operator	<code>C >=> 2</code> is same as <code>C = C >> 2</code>
<code>&=</code>	Bitwise AND assignment operator	<code>C &= 2</code> is same as <code>C = C & 2</code>
<code>^=</code>	bitwise exclusive OR and assignment operator	<code>C ^= 2</code> is same as <code>C = C ^ 2</code>

Misc. Operator in Java

Conditional Operator

Conditional operator is also known as the ternary operator. This operator consists of three operands and is used to evaluate Boolean expressions. The goal of the operator is to decide which value should be assigned to the variable. The operator is written as:

variable x = (expression) ? value if true : value if false

Introduction to Eclipse IDE

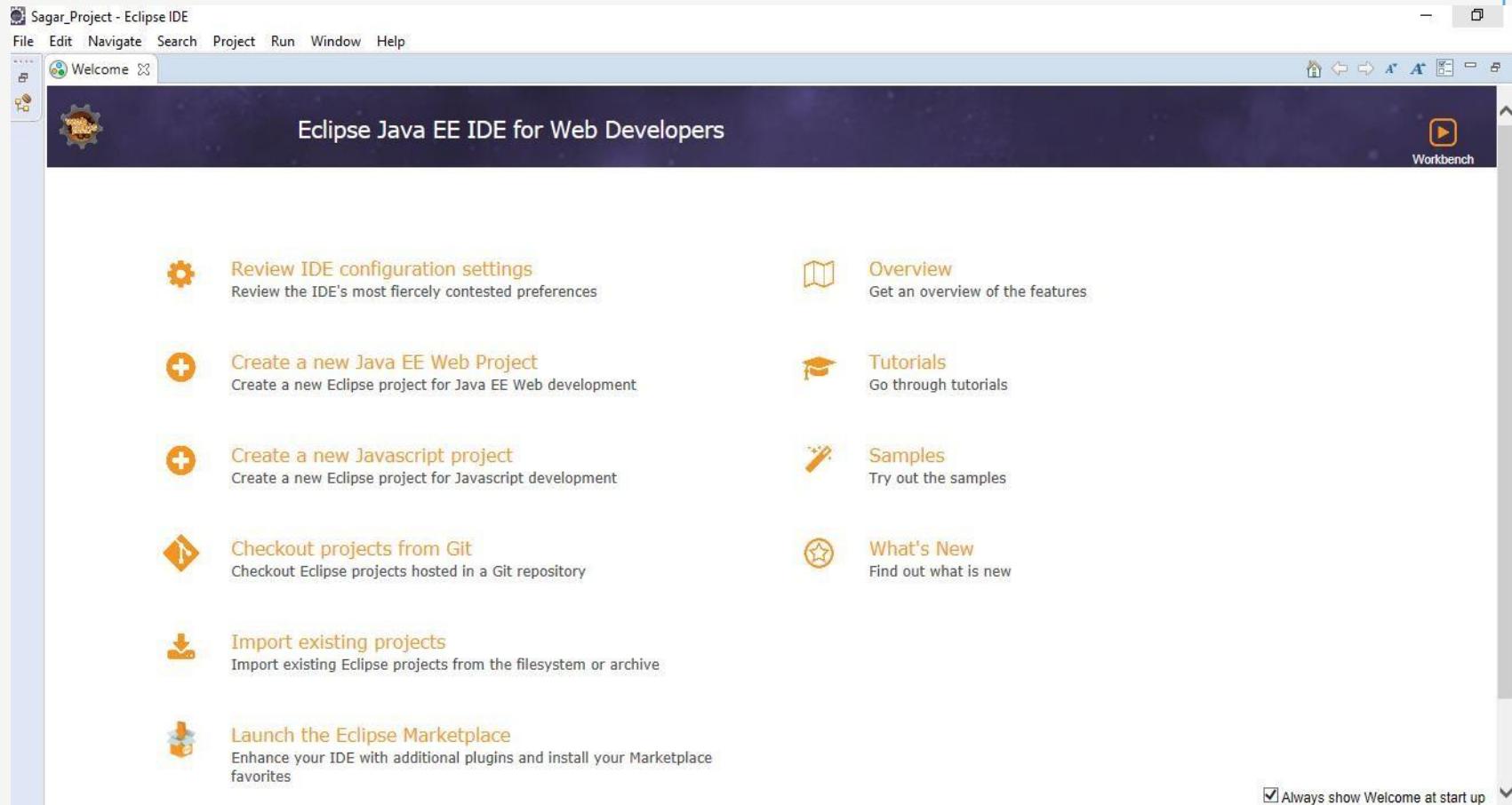
Java IDE Tools

- To write your Java programs, you will need a text editor. There are even more sophisticated IDEs available in the market.
- **Notepad:** On Windows machine you can use any simple text editor like Notepad, TextPad.
- **Netbeans:** is a Java IDE that is open-source.
- **Eclipse:** is also a Java IDE developed by the eclipse open-source community.

About Eclipse IDE

- Most people know **Eclipse** as an **integrated development environment (IDE)** for Java.
- Today it is the leading development environment for Java with a market share of approximately 65%.
- The Eclipse IDE can be extended with additional software components.
- Eclipse calls these software components plug-ins. Several Open Source projects and companies have extended the Eclipse IDE

Eclipse IDE



The screenshot shows the Eclipse IDE Workbench interface. The title bar reads "Sagar_Project - Eclipse IDE". The menu bar includes File, Edit, Navigate, Search, Project, Run, Window, and Help. The toolbar has icons for New Project, Open Project, Save, Undo, Redo, Cut, Copy, Paste, Find, Replace, and others. The main area is titled "Eclipse Java EE IDE for Web Developers" and features a "Workbench" icon in the top right corner. The "Welcome" view is displayed, listing several options:

- Review IDE configuration settings**: Review the IDE's most fiercely contested preferences.
- Create a new Java EE Web Project**: Create a new Eclipse project for Java EE development.
- Create a new Javascript project**: Create a new Eclipse project for Javascript development.
- Checkout projects from Git**: Checkout Eclipse projects hosted in a Git repository.
- Import existing projects**: Import existing Eclipse projects from the filesystem or archive.
- Launch the Eclipse Marketplace**: Enhance your IDE with additional plugins and install your Marketplace favorites.
- Overview**: Get an overview of the features.
- Tutorials**: Go through tutorials.
- Samples**: Try out the samples.
- What's New**: Find out what is new.

At the bottom right of the welcome view, there is a checkbox labeled "Always show Welcome at start up" with a checked mark.

Java program Structure

Introduction of

- Class, Object , Class Members
- Constructor

Class, Object and members

- **Class**

- A class can be defined as a template/blue print that describes the behaviors/states that object of its type support.



Class, Object and members

- Objects are key to understanding object-oriented technology.
- Examples of real-world objects:
 - your dog,
 - your desk,
 - your television set,
 - your bicycle.
- Real-world objects share two characteristics:
 - They all have state and behavior.
 - Example:
 - A dog has states - color, name, breed as well as behaviors -wagging, barking, eating.
 - An object is an instance of a class.

Class, Object and members

- An **object** is an instance of a class created using a new operator.
- The new operator returns a reference to a new instance of a class.
- Eg Person me=new Person();

Class Members

- When we create a class its totally incomplete without defining any member of this class.

Class, Object and members

- **Field:** field is nothing but the property of the class or object which we are going to create .
 - for example if we are creating a class called computer then they have property like model, mem_size, hd_size, os_type etc
- **Method:** method is nothing but the operation that an object can perform it define the behavior of object how an object can interact with outside world .
 - For example : startMethod (), shutdownMethod ().

Refer Example here-

https://github.com/TopsCode/Automation-Testing/blob/master/Module-2/2.0%20Class%20and%20Object/2.0.1%20Class_ObjectDemo.java

Constructors

- Every class has a constructor. If we do not explicitly write a constructor for a class the Java compiler builds a default constructor for that class.
- Each time a new object is created, at least one constructor will be invoked.
- The main rule of constructors is that they should have the same name as the class.
- A class can have more than one constructor.
- Constructor declarations look like method declarations—except that they use the name of the class and have no return type.

Constructors

- They are called or invoked when an object of class is created and can't be called explicitly.
- E.g. public class Person{
 //attribute declaration
 Person(){
 }
 //method declaration }
- Like other methods in java constructor can be overloaded i.e. we can create as many constructors in our class as desired.
- Number of constructors depends on the information about attributes of an object we have while creating objects.

Constructors

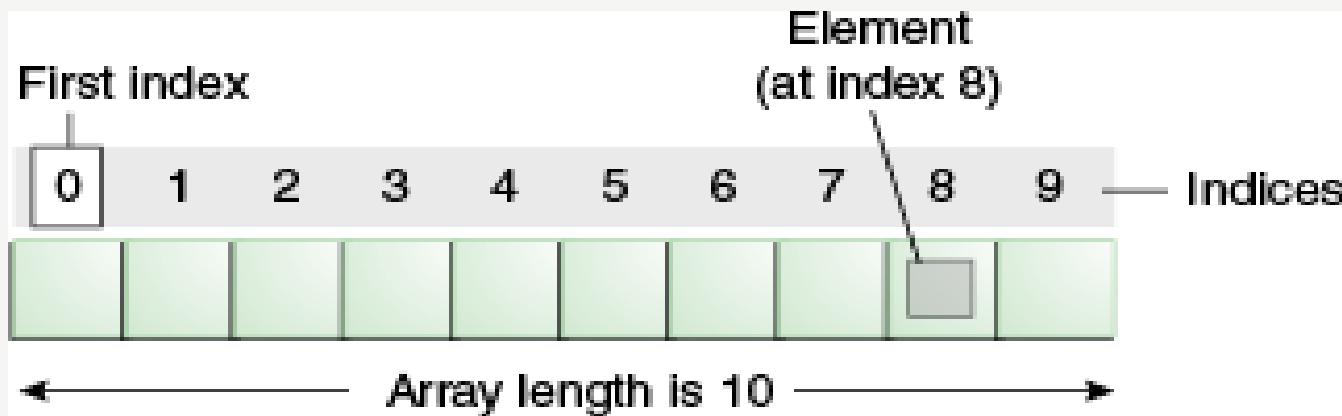
Refer Example here-

<https://github.com/TopsCode/Automation-Testing/blob/master/Module-2/2.1%20Constructor/2.1.1%20ConstructorDemo.java>

Array

- **Introduction:**

- An array is a container object that holds a fixed number of values of a single type.
- The length of an array is established when the array is created.
- After creation, its length is fixed.



Array

Advantages of Array:

- Arrays are most appropriate for storing a fixed amount of data which will be accessed in an unpredictable fashion.
- Arrays that has become very important on modern architectures is that iterating through an array has good locality of reference.
- Arrays are much faster than iterating through a linked list of the same size, which tends to jump around in memory.
- Arrays also are among the most compact data structures; storing 100 integers in an array takes only 100 times the space required to store an integer

Array

Array Types

There are 2 types of array available in java.

1) Primitive type,

- 1. byte b; // byte is a primitive type
- 2. byte[] arrayOfBytes; // byte[] is an array type: array of byte
- 3. byte[][] arrayArrayOfBytes; // byte[][] is another type: array of byte[]

2) Class/Object type.

Point[] points;

Array

Primitive Types and Reference type Arrays.

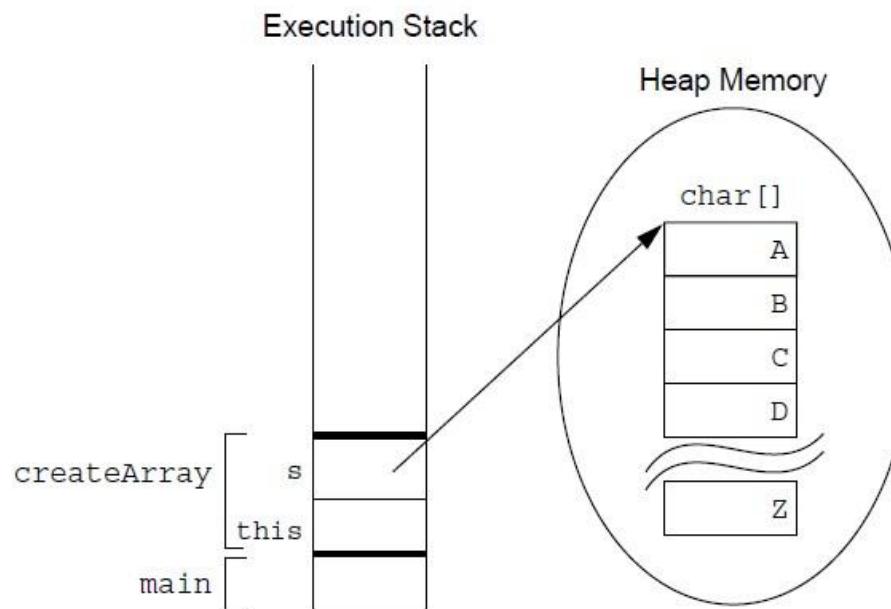
- char s[];
- Point p[];
- char[] s;
- Point[] p;

Primitive Array

```
public char[] createArray() {  
    char[] s= new char[26];  
    for ( int i=0; i<26; i++ ) {  
        s[i] = (char) ('A' + i);  
    }  
    return s;}  
    
```

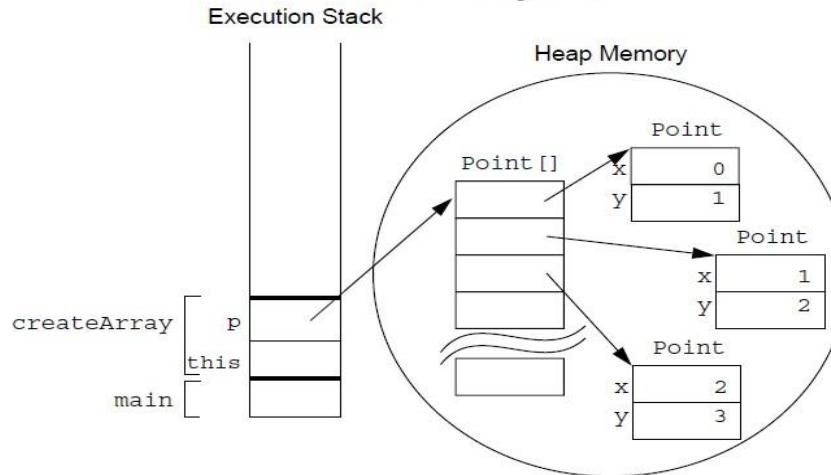
Array

Creating an Array of Character Primitives



Array

Creating an Array of Character Primitives With Point Objects



Array

Refer Example here-

https://github.com/TopsCode/Automation-Testing/blob/master/Module-2/2.2%20Array/2.2.1%20Array_Demo.java

String in Java

- The String class represents character strings.
- Strings are constant;
- Their values cannot be changed after they are created.
- It's immutable/fixed class.

Methods of String class

Function Name	Use
char charAt(int index)	Returns the char value at the specified index. An index ranges from 0 to length() - 1
public void getChars(int srcBegin, int srcEnd, char[] dst, int dstBegin)	Copies characters from this string into the destination character array.

Methods of String class

Function Name	Use
<code>public void getBytes(int srcBegin, int srcEnd, byte[] dst, int dstBegin)</code>	Copies characters from this string into the destination byte array.
<code>public boolean equalsIgnoreCase(String anotherString)</code>	Compares this String to another String, ignoring case considerations.

Methods of String class

Function Name	Use
public String substring(int beginIndex)	Returns a new string that is a substring of this string. The substring begins with the character at the specified index and extends to the end of this string.
public String concat(String str)	Concatenates the specified string to the end of this string.
public String replace(char oldChar, char newChar)	Returns a new string resulting from replacing all occurrences of oldChar in this string with newChar.

Methods of String class

Function Name	Use
public String trim()	Returns a copy of the string, with leading and trailing whitespace omitted.
public char[] toCharArray()	Converts this string to a new character array.
public static String valueOf(char c)	Returns the string representation of the char argument.

Methods of String class

Refer Example here-

<https://github.com/TopsCode/Automation-Testing/blob/master/Module-2/2.3%20String/2.3.1%20StringDemo.java>

StringBuffer and StringBuilder

Strings objects of type StringBuffer and Stringbuilder can be modified over and over again with out leaving behind a lot of new unused objects.

StringBuffer	StringBuilder
StringBuffer methods are thread safe.	StringBuilders methods are not thread safe(not Synchronised).
Less efficient than StringBuilder.	More efficient than StringBuffer.
When the appliacation needs to run on multiple threads then it is better to use StringBuffer.	When the application needs to be run only in a single thread then it is better to use StringBuilder.

StringBuffer and StringBuilder

StringBuffer:Refer Example here-

<https://github.com/TopsCode/Automation-Testing/blob/master/Module-2/2.3%20String/2.3.3%20StringBufferDemo.java>

StringBuilder:Refer Example here-

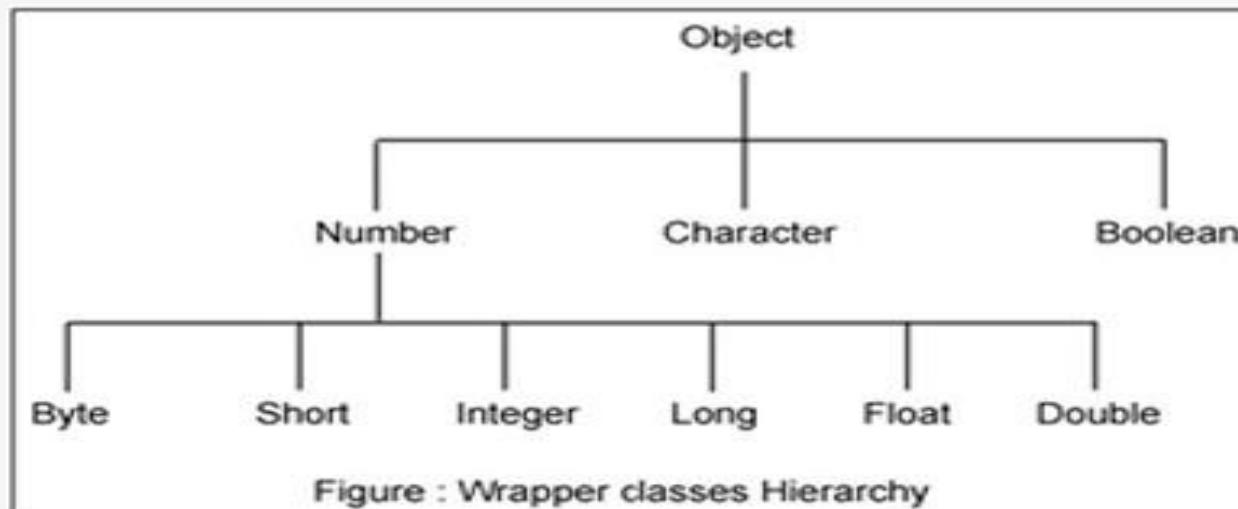
<https://github.com/TopsCode/Automation-Testing/blob/master/Module-2/2.3%20String/2.3.2%20StringBuilderDemo.java>

Wrapper Class

- Each of Java's eight primitive data types has a class dedicated to it.

Several possible reasons for wrapper classes availability,

- For the null value is possible,
- To include in a Collection,
- To treat generically / polymorphically as an Object along with other Objects



Boxing

- Some Terms:

- **Boxing**

- Boxing/wrapping, is the process of placing a primitive type within an object so that the primitive can be used as a reference object.

- **Autoboxing**

- Autoboxing is the term for getting a reference type out of a value type just through type conversion (either implicit or explicit).
 - The compiler automatically supplies the extra source code which creates the object.
 - `Integer i = new Integer(9);`
 - `Integer l = 9;`

Boxing

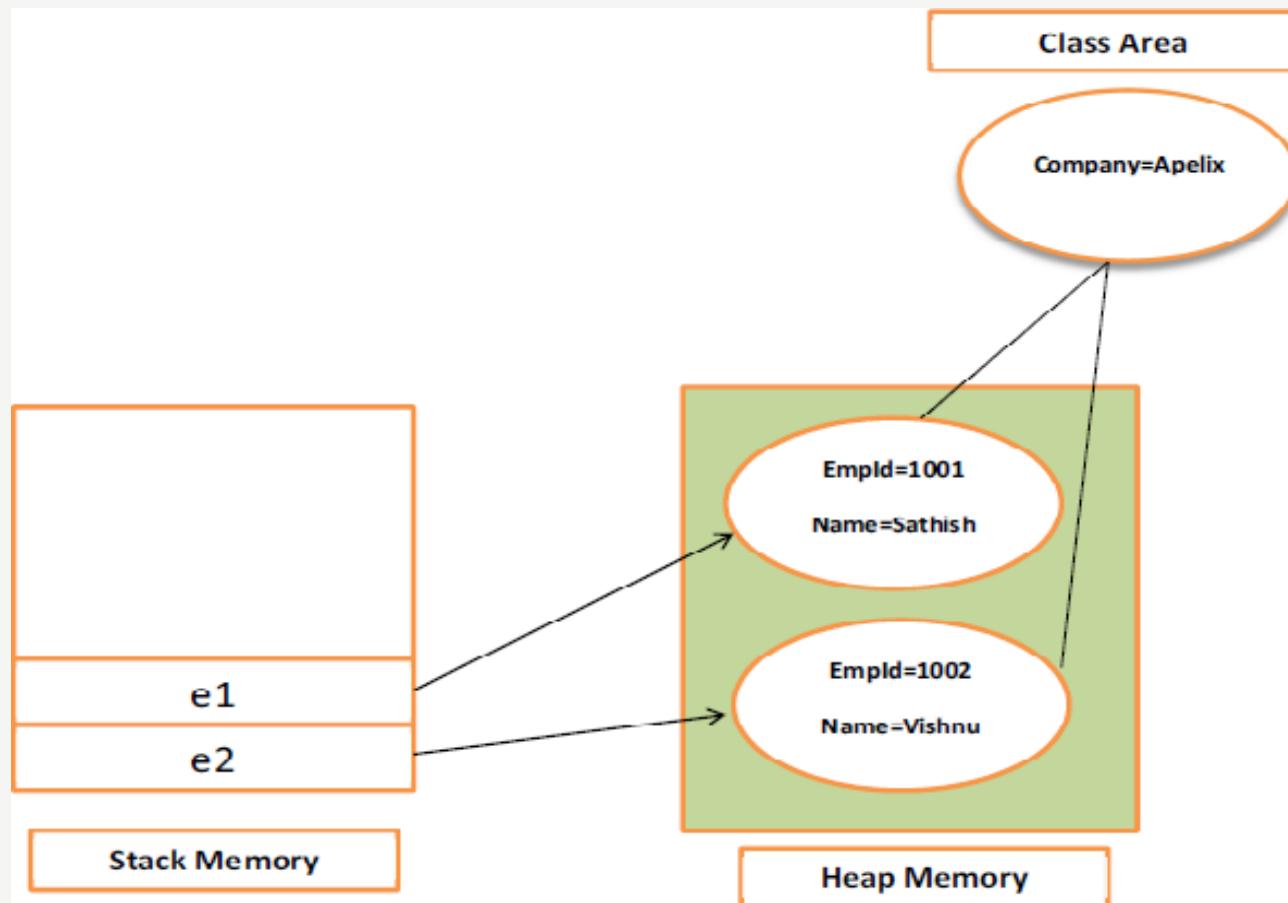
● Unboxing

- Unboxing refers to getting the value which is associated to a given object, just through type conversion (either implicit or explicit).
- The compiler automatically supplies the extra source code which retrieves the value out of that object, either by invoking some method on that object, or by other means.
- Integer k = new Integer(4);
- int l = k.intValue();
- int m = k;

Refer Example here-

<https://github.com/TopsCode/Automation-Testing/blob/master/Module-2/2.4%20Wrapper%20Class/2.4.1%20WrapperClassDemo.java>

Keyword-static



static variable

- Java instance variables are given separate memory for storage.
- If there is a need for a variable to be common to all the objects of a single java class, then the static modifier should be used in the variable declaration.
- Any java object that belongs to that class can modify its static variables.
- Also, an instance is not a must to modify the static variable and it can be accessed using the java class directly.
- Static variables can be accessed by java instance methods also.

static method

- static methods are also common to classes and not tied to a java instance.
- static methods should be invoked with using the class name though it can be invoked using an object.

ClassName.methodName(arguments)

or

objectName.methodName(arguments)

- General use for java static methods is to access static fields.
- Static methods can be accessed by java instance methods.
- Java static methods cannot access instance variables or instance methods directly.
- Java static methods cannot use the “this” keyword.

static initialize block

- A class can contain code in a static block that does not exist within a method body.
- Static block code executes once only, when the class is loaded.
- Usually, a static block is used to initialize static (class) attributes.

Refer Example here-

https://github.com/TopsCode/Automation-Testing/blob/master/Module-2/2.5%20Keywords/2.5.1%20Keyword_staticDemo.java

Keyword-final

- **final** modifier can be used with
Class,
Variable &
Method.
- **final class**

This class can not be extended.

```
public final class MyFinalClass
{...
}
// forbidden
public class subclass extends MyFinalClass
{...
}
```

Keyword-final

final variable

- A final variable is a constant.
- A variable that is declared as final and not initialized is called a blank final variable.
- A blank final variable forces the constructors to initialise it.

Keyword-final

final method

- You cannot override a final method.
- A blank final method variable must be set in the method body before being used.

Refer Example here-

https://github.com/TopsCode/Automation-Testing/blob/master/Module-2/2.5%20Keywords/2.5.2%20Keyword_finalDemo.java

OOPS

- Object Oriented Programming System
- Object means a real word entity such as pen, chair, table etc.
- Object-Oriented Programming is a methodology or paradigm to design a program using classes and objects.
- It simplifies the software development and maintenance by providing some concepts:

OOP Concepts

- Object
- Class
- Inheritance
- Polymorphism
- Abstraction
- Encapsulation

OOP Concepts

Object	Any entity that has state and behavior is known as an object. For example: chair, pen, table, keyboard, bike etc. It can be physical and logical.
Class	Collection of objects is called class. It is a logical entity.

Refer Example here-

https://github.com/TopsCode/Automation-Testing/blob/master/Module-2/2.6%20OOPS/2.6.1%20Class%20and%20Object/2.6.1.1%20Class_ObjectDemo.java

OOP Concepts

Encapsulation	<p>Binding (or wrapping) code and data together into a single unit is known as encapsulation. For example: capsule, it is wrapped with different medicines.</p> <p>A java class is the example of encapsulation. Java bean is the fully encapsulated class because all the data members are private here.</p>

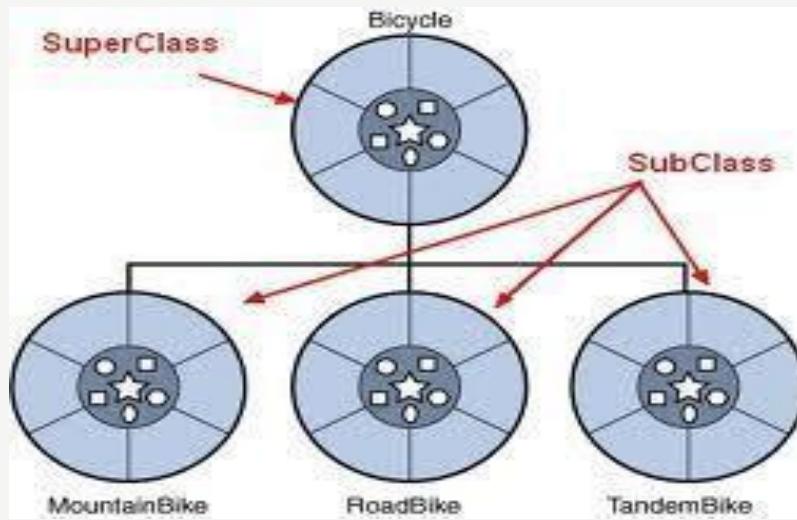
[Refer Example here-](#)

<https://github.com/TopsCode/Automation-Testing/blob/master/Module-2/2.6%20OOPS/2.6.2%20Encapsulation/2.6.2.1%20EncapsulationDemo.java>

OOP Concepts

Inheritance

- Inheritance is a mechanism wherein a new class is derived from an existing class.
- In Java, classes may inherit or acquire the properties and methods of other classes.
- A class derived from another class is called a sub class, whereas the class from which a subclass is derived is called a super class.



OOP Concepts

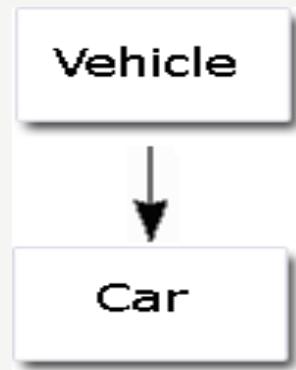
Inheritance-Use

Reusability	facility to use public methods of base class without rewriting the same.
Extensibility	extending the base class logic as per business logic of the derived class.
Data hiding	base class can decide to keep some data private so that it cannot be altered by the derived class.
Overriding	With inheritance, we will be able to override the methods of the base class so that meaningful implementation of the base class method can be designed in the derived class.

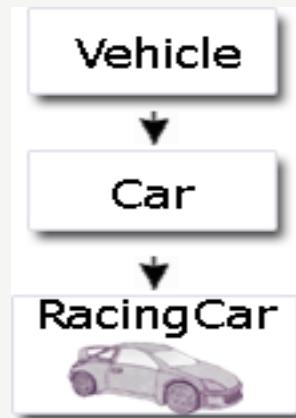
OOP Concepts

Inheritance Types-

Simple Inheritance



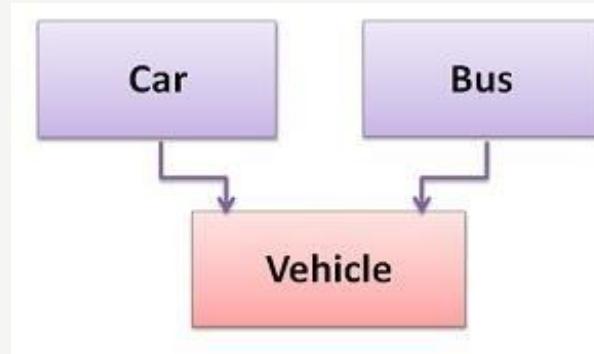
Multilevel Inheritance



OOP Concepts

Inheritance Types-

Multiple Inheritance



OOP Concepts

Simple Inheritance

- When a subclass is derived simply from its parent class then this mechanism is known as simple inheritance.
- In case of simple inheritance there is only a sub class and its parent class.
- It is also called single inheritance or one level inheritance.

Refer Example here-

<https://github.com/TopsCode/Automation-Testing/blob/master/Module-2/2.6%20OOPS/2.6.3%20Inheritance/2.6.3.1%20SimpleInheritanceDemo.java>

OOP Concepts

- Multilevel Inheritance
 - It is the enhancement of the concept of inheritance. When a subclass is derived from a derived class then this mechanism is known as the multilevel inheritance.
 - The derived class is called the subclass or child class for it's parent class and this
- parent class works as the child class for it's just above (parent) class.
- Multilevel inheritance can go up to any number of level.
- [Refer Example here-](#)
- <https://github.com/TopsCode/Automation-Testing/blob/master/Module-2/2.6%20OOPS/2.6.3%20Inheritance/2.6.3.2%20MultilevelInheritanceDemo.java>

OOP Concepts

Multiple Inheritance

- Java does not support multiple Inheritance.
- The mechanism of inheriting the features of more than one base class into a single class is known as multiple inheritance.
- Java does not support multiple inheritance but the multiple inheritance can be achieved by using the interface.
- In Java Multiple Inheritance can be achieved through use of Interfaces by implementing more than one interfaces in a class.

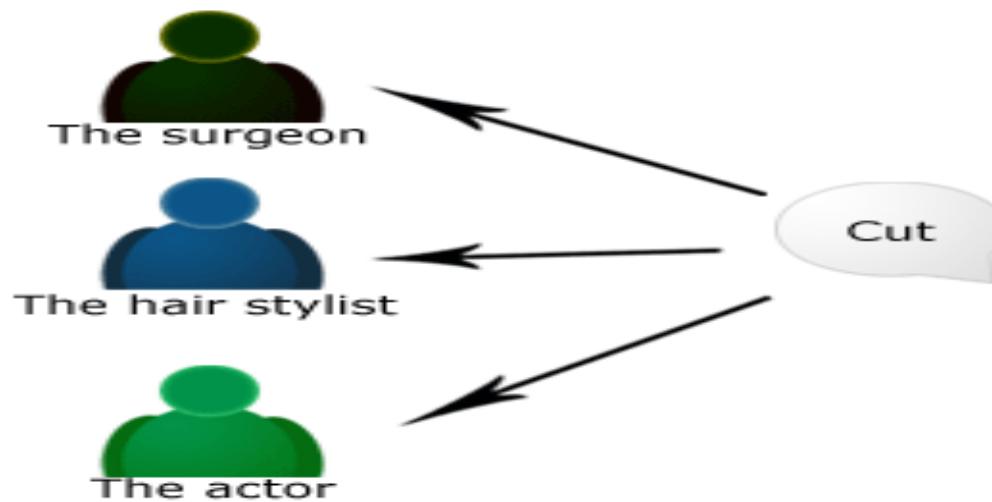
OOP Concepts

Polymorphism	<p>When one task is performed by different ways i.e. known as polymorphism. For example: to converse the customer differently, to draw something e.g. shape or rectangle etc. In java, we use method overloading and method overriding to achieve polymorphism.</p> <p>Another example can be to speak something e.g. cat speaks meaw, dog barks woof etc.</p>

- Types of Polymorphism
 - Method Overloading
 - Method Overriding

OOP Concepts

IF ANY BODY SAYS "CUT" TO THESE PEOPLE



The surgeon would begin to make an incision.

The hair stylist would begin to cut someone's hair.

The actor would abruptly stop acting out the current scene, awaiting directorial guidance.

OOP Concepts

Polymorphism

- Polymorphism is the ability of an object to take on many forms.
- Any Java object that can pass more than one IS-A test is considered to be polymorphic.
- For e.g.

```
public interface Vegetarian{ }  
public class Animal{ }  
public class Deer extends Animal implements Vegetarian{}
```

OOP Concepts

Polymorphism

- The Deer class is considered to be polymorphic since this has multiple inheritance.
- Following are true for the above example:
 - A Deer IS-A Animal
 - A Deer IS-A Vegetarian
 - A Deer IS-A Deer
 - A Deer IS-A Object
- Polymorphism is essentially considered into two versions.
 - Compile time polymorphism /static binding/method overloading.
 - Runtime polymorphism/dynamic binding/method overriding.

OOP Concepts

Method overloading	Method overriding
In case of method overloading in Java, Signature of method changes .	while in case of method overriding signature of method (including return type, number of method parameters, type of parameters and order of parameters)remain same.
Java is that You can overload method in one class .	While overriding can only be done on subclass.
You can overload static, final and private method in Java.	While you can not override static, final or private method in Java.
Overloaded method in Java is bonded by static binding .	Overridden methods are subject to dynamic binding.

OOP Concepts

Method Overloading:Refer Example here-

<https://github.com/TopsCode/Automation-Testing/blob/master/Module-2/2.6%20OOPS/2.6.4%20Polymorphism/2.6.4.1%20MethodOverloadingDemo.java>

Method Overriding:Refer Example here-

<https://github.com/TopsCode/Automation-Testing/blob/master/Module-2/2.6%20OOPS/2.6.4%20Polymorphism/2.6.4.2%20MethodOverridingDemo.java>

OOP Concepts

Abstraction	<p>Hiding internal details and showing functionality is known as abstraction. For example: phone call, we don't know the internal processing.</p> <p>In java, we use abstract class and interface to achieve abstraction.</p>

Interface

- An interface is a collection of abstract methods.
- A class implements an interface, thereby inheriting the abstract methods of the interface.
- Class describes the attributes and behaviours of an object.
- An interface contains behaviours that a class implements.

OOP Concepts

Class vs. Interface

- You cannot instantiate an interface.
- An interface does not contain any constructors.
- All of the methods in an interface are abstract.
- An interface cannot contain instance fields. The only fields that can appear in an interface must be declared both static and final.
- An interface is not extended by a class; it is implemented by a class.
- An interface is not extended by a class; it is implemented by a class.
- An interface can extend multiple interfaces.

OOP Concepts

Abstract Class : Refer Example here-

https://github.com/TopsCode/Automation-Testing/blob/master/Module-2/2.6%20OOPS/2.6.5%20Abstraction/2.6.5.1%20Abstraction_AbstractClassDemo.java

Interface: Refer Example here-

https://github.com/TopsCode/Automation-Testing/blob/master/Module-2/2.6%20OOPS/2.6.5%20Abstraction/2.6.5.2%20Abstraction_InterfaceDemo.java

Exception in Java

- **Introduction**
- **Types of Exception**
- **try catch and finally block**
- **Multi catch Exceptions**
- **throw and throws keywords**
- **Method overriding with Exceptions**
- **Custom Exceptions**

Exception in Java

Introduction

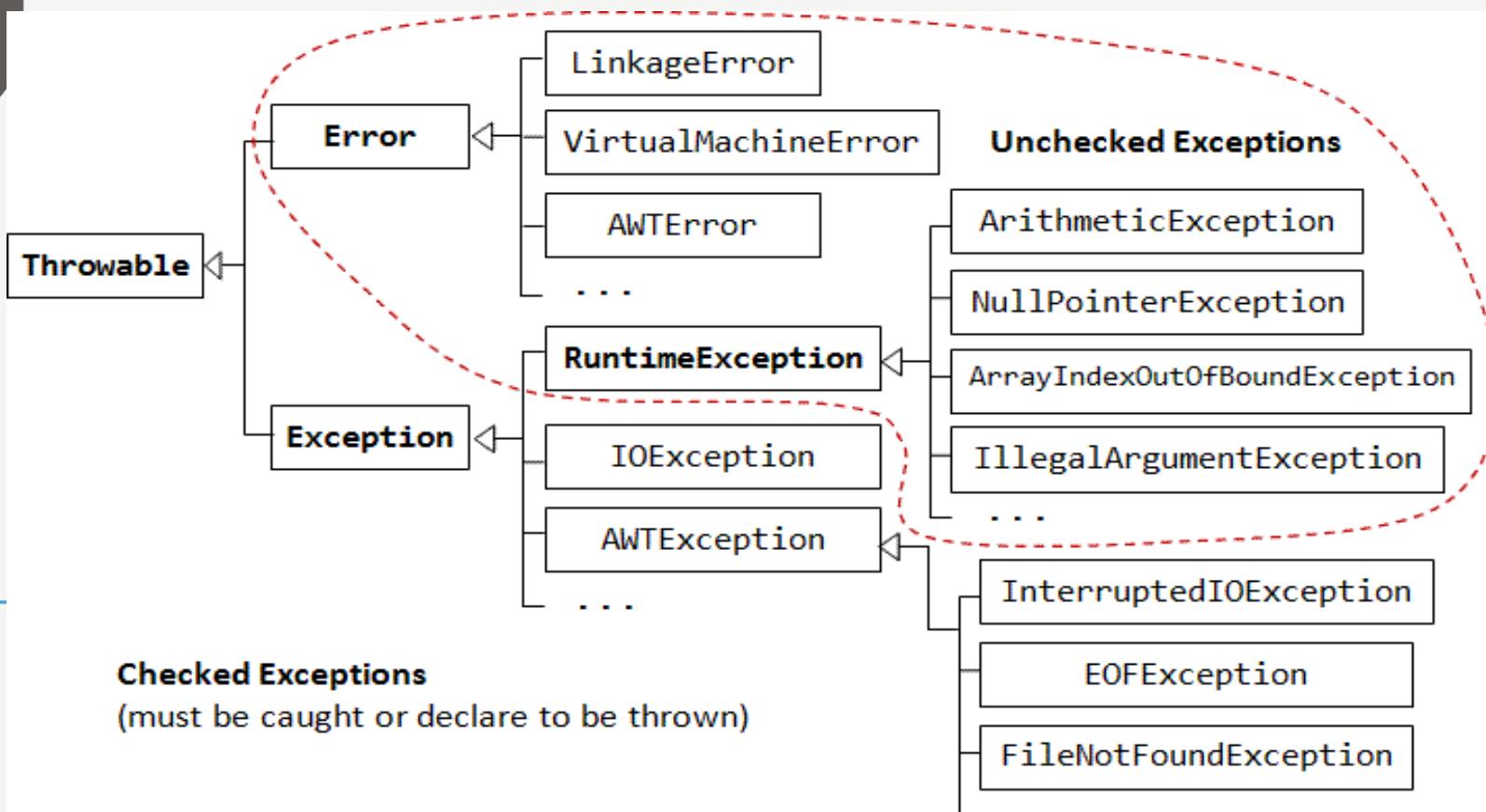
- An exception is a problem that arises during the execution of a program.
- An exception can occur for many different reasons:
 - A user has entered invalid data.
 - A file that needs to be opened cannot be found.
 - A network connection has been lost in the middle of communications or the JVM has run out of memory.

Exception in Java

- There are three categories of exceptions:

Exception	Description
Checked exceptions:	A checked exception is an exception that is typically a user error or a problem that cannot be foreseen by the programmer. For example, if a file is to be opened, but the file cannot be found, an exception occurs. These exceptions cannot simply be ignored at the time of compilation.
Runtime exceptions:	A runtime exception is an exception that occurs that probably could have been avoided by the programmer. As opposed to checked exceptions, runtime exceptions are ignored at the time of compilation.
Errors:	These are not exceptions at all, but problems that arise beyond the control of the user or the programmer. For example, if a stack overflow occurs, an error will arise. They are also ignored at the time of compilation.

Exception in Java



Exception in Java

Exception Handling

try block

- The first step in constructing an exception handler is to enclose the code that might throw an exception within a try block.
- try
 - {
 - *code*
 - }
 - *catch and finally blocks . . .*

Exception in Java

catch block

- You associate exception handlers with a try block by providing one or more catch blocks directly after the try block.

```
try {  
}  
  
catch (ExceptionType name) {  
}
```

Exception in Java

finally block

- The finally keyword is used to create a block of code that follows a try block.
- A finally block of code always executes, whether or not an exception has occurred.

Refer Example here-

<https://github.com/TopsCode/Automation-Testing/blob/master/Module-2/2.7%20Exception/2.4.1%20ExceptionDemo.java>

Exception in Java

Multiple catch blocks

```
try
{ //Protected code }
catch(ExceptionType1 e1)
{ //Catch block }
catch(ExceptionType2 e2)
{ //Catch block }
catch(ExceptionType3 e3)
{ //Catch block }
```

Exception in Java

- If an exception occurs in the protected code, the exception is thrown to the first catch block in the list.
- If the data type of the exception thrown matches ExceptionType1, it gets caught there.
- If not, the exception passes down to the second catch statement.

Refer Example here-

<https://github.com/TopsCode/Automation-Testing/blob/master/Module-2/2.7%20Exception/2.4.2%20MultiplaCatchDemo.java>

Exception in Java

throw & throws keyword

- If a method does not handle a checked exception, the method must declare it using the **throws** keyword.
- The throws keyword appears at the end of a method's signature.
- You can throw an exception, either a newly instantiated one or an exception that you just caught, by using the **throw** keyword

Exception in Java

throw	throws
throw is use to explicitly throw the exceptions.	throws is use to declare the exceptions.
throw is followed by instance	throws is followed by class
throw is used within method.	throws is used with the method signature.
You can not throw multiple exception .	you can declare multiple exceptions using throws.

Exception in Java

Custom Exception

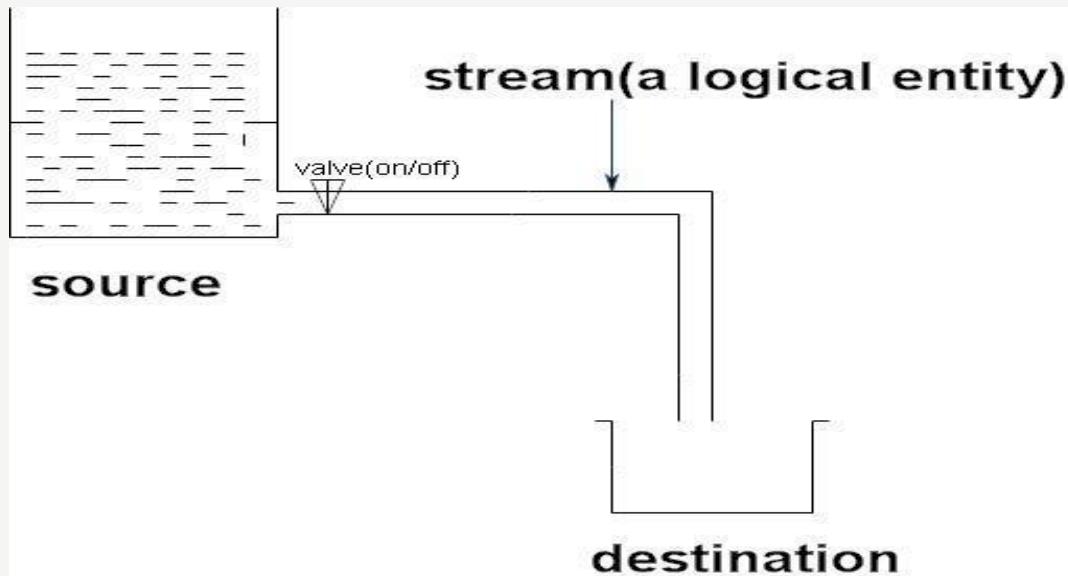
- You can create your own exceptions in Java.
- For that,
 - All exceptions must be a child of Throwable.
 - If you want to write a checked exception that is automatically enforced by the Handle or Declare Rule, you need to extend the **Exception** class.
 - If you want to write a runtime exception, you need to extend the **RuntimeException** class.
 - Eg class MyException extends Exception{ }

Refer Example here-

<https://github.com/TopsCode/Automation-Testing/blob/master/Module-2/2.7%20Exception/2.4.3%20User%20Exception/2.4.3.1%20BankDemo.java>

File IO in Java

- What is Stream and Types of Stream
- File Input and Output Stream and its Methods
- File class
- Command Line Arguments.

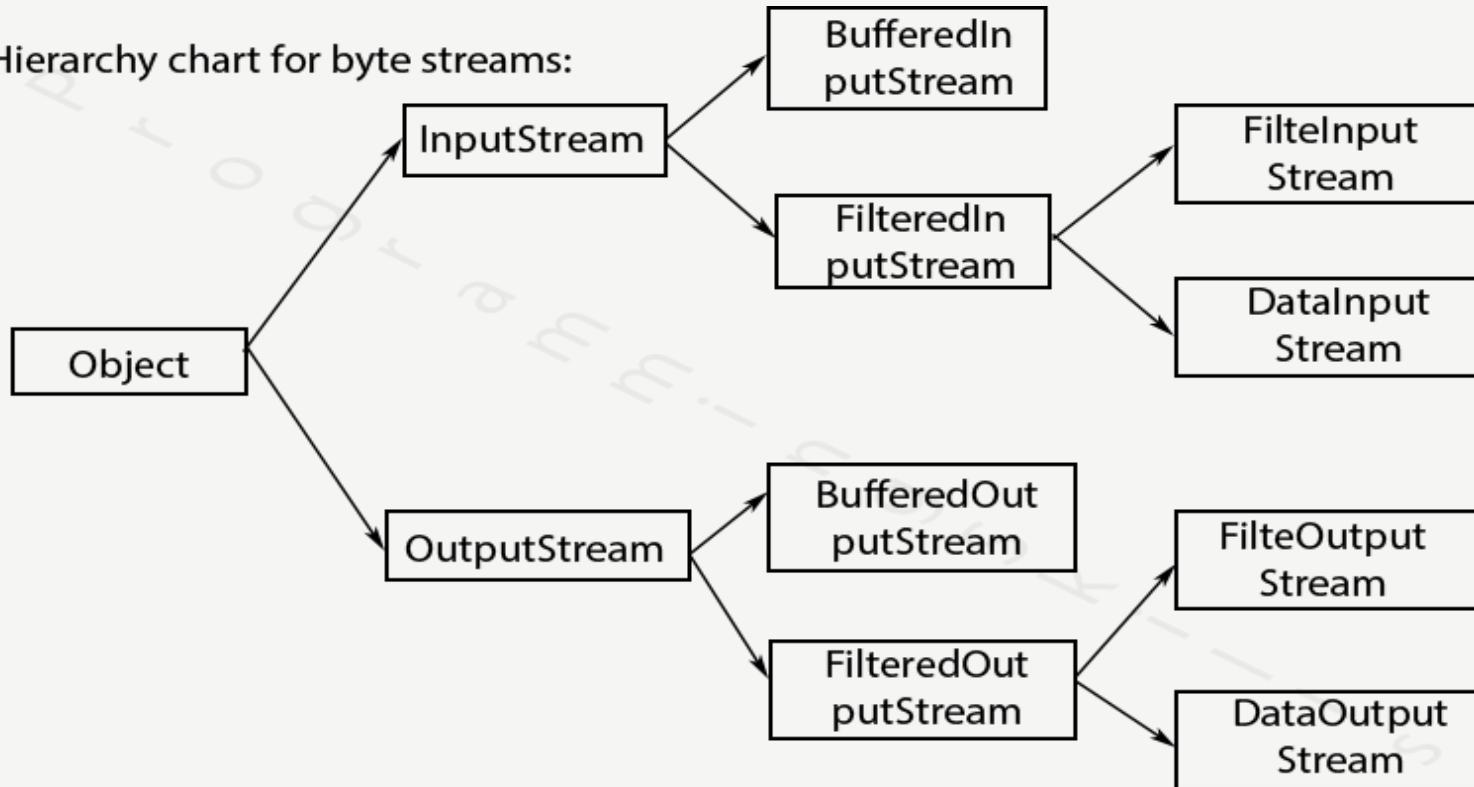


File IO in Java

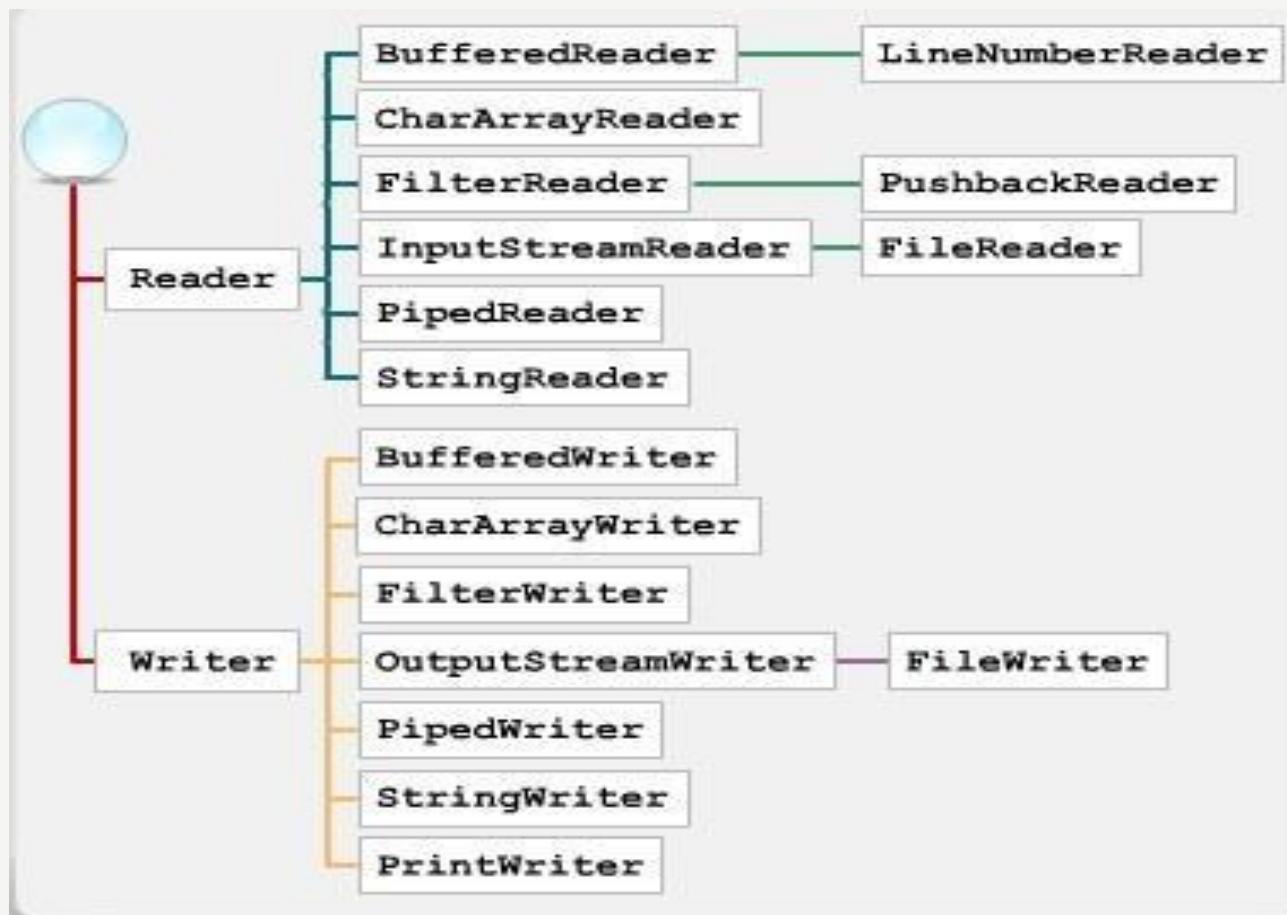
- A stream can be defined as sequence of data from one place to another.
- A stream can represent many different kinds of sources and destinations, including disk files, devices, other programs, and memory arrays.
- Streams are used read/write data from file resides on network or hard disk or other storage device.
- An input stream/Source Stream initiates the flow of data, used to read data from Source.
- An output stream/Sink Stream terminates flow of data, used for writing data to a destination.

File IO in Java

Hierarchy chart for byte streams:



File IO in Java



InputStream Methods

Method details

`public int available() throws IOException`

Returns the number of bytes that can be read (or skipped over) from this input stream without blocking by the next caller of a method for this input stream.

`public void close() throws IOException`

Closes this input stream and releases any system resources associated with the stream.

`public void mark(int readlimit)`

Marks the current position in this input stream.

`public void reset() throws IOException`

Repositions this stream to the position at the time the mark method was last called on this input stream.

InputStream Methods

Method details

public boolean markSupported()	Tests if this input stream supports the mark and reset methods.
public abstract int read() throws IOException	Reads the next byte of data from the input stream. The value byte is returned as an int in the range 0 to 255. If no byte is available because the end of the stream has been reached, the value - 1 is returned.

InputStream Methods

Method details

`public int read(byte[] b) throws
IOException`

Reads some number of bytes from the input stream and stores them into the buffer array b.

`public int read(byte[] b, int off, int len)
throws IOException`

Reads up to len bytes of data from the input stream into an array of bytes. An attempt is made to read as many as len bytes. The number of bytes actually read is returned as an integer.

OutputStream Methods

Method details

public abstract void **write(int b)** throws IOException

Writes the specified byte to this output stream.

public void **write(byte[] b)** throws IOException

Writes b.length bytes from the specified byte array to this output stream.
The general contract for write(b) is that it should have exactly the same effect as the call write(b, 0, b.length).

public void **write(byte[] b, int off, int len)** throws IOException

Writes len bytes from the specified byte array starting at offset off to this output stream.

public void **flush()** throws IOException

Flushes this output stream and forces any buffered output bytes to be written out.

InputStream & OutputStream

Refer Example here-

<https://github.com/TopsCode/Automation-Testing/blob/master/Module-2/2.8%20File%20Handling/2.8.1%20InputStreamDemo.java>

Reader & Writer

- Java IO's Reader and Writer work much like the InputStream and OutputStream with the exception that Reader and Writer are character based.
- They are intended for reading and writing text.
- The InputStream and OutputStream were byte based.
- **Reader**
- The Reader is the baseclass of all Reader's in the Java IO API. Subclasses include a BufferedReader, PushbackReader etc.

Reader & Writer

Methods	Description
abstract void close()	Closes the stream and releases any system resources associated with it.
void mark(int readAheadLimit)	Marks the present position in the stream.
boolean markSupported()	Tells whether this stream supports the mark() operation.
Int read()	Reads a single character.
Int read(char[] cbuf)	Reads characters into an array.

Reader & Writer

Methods	Description
Int read(char[] cbuf,int off,int len)	Reads characters into a portion of an array.
Int read(CharBuffer targert)	Attempts to read characters into the specified character buffer.
void reset()	Resets the stream.
long skip()	Skips characters.

Reader & Writer

Writer

- You will normally use a Writer subclass rather than a Writer directly. Subclasses of Writer include OutputStreamWriter, CharArrayWriter, FileWriter, plus many others

Methods	Description
Writer append(char c)	Appends the specified character to this writer.
Writer append(CharSequence seq)	Appends the specified character sequence to this writer.
Writer append(CharSequence seq,int start,int end)	Appends a subsequence of the specified character sequence to this writer.
void close()	Closes the stream, flushing it first.

Reader & Writer

Methods	Description
void flush()	Flushes the stream.
void write(char[] cbuf)	Writes an array of characters.
void write(char[] cbuf,int off,int len)	Writes a portion of an array of characters.
void write(int c)	Writes a single character.
void write(String str)	Writes a string.
Void write(String str,int off,int len)	Writes a portion of a string.

Refer Example here-

<https://github.com/TopsCode/Automation-Testing/blob/master/Module-2/2.8%20File%20Handling/2.8.2%20ReaderWriterClassDemo.java>

File Class

- Java File class represents the files and directory pathnames in an abstract manner.
- This class is used for creation of files and directories, file searching, file deletion etc.
- The File object represents the actual file/directory on the disk.
- There are following constructors to create a File object:
 - `File(File parent, String child);`
 - `File(String pathname)`
 - `File(String parent, String child)`
 - `File(URI uri)`

File Class

Method details

public boolean **createNewFile()**
throws IOException

Atomically creates a new, empty file named by this abstract pathname if and only if a file with this name does not yet exist.

public boolean **delete()**

Deletes the file or directory denoted by this abstract pathname. If this pathname denotes a directory, then the directory must be empty in order to be deleted.

public String[] **list()**

Returns an array of strings naming the files and directories in the directory denoted by this abstract pathname.

public boolean **mkdir()**

Creates the directory named by this abstract pathname.

File Class

Method details

public long length()	Returns the length of the file denoted by this abstract pathname. The return value is unspecified if this pathname denotes a directory.
public boolean isFile()	Tests whether the file denoted by this abstract pathname is a normal file.
public boolean isDirectory()	Tests whether the file denoted by this abstract pathname is a directory.
public boolean exists()	Tests whether the file or directory denoted by this abstract pathname exists.
public boolean canWrite()	Tests whether the application can modify to the file denoted by this abstract pathname.
public boolean canRead()	Tests whether the application can read the file denoted by this abstract pathname.

File Class

Method details

public String getName()	Returns the name of the file or directory denoted by this abstract pathname.
public String getParent()	Returns the pathname string of this abstract pathname's parent, or null if this pathname does not name a parent directory.
public String getPath()	Converts this abstract pathname into a pathname string.

Refer Example here-

<https://github.com/TopsCode/Automation-Testing/blob/master/Module-2/2.8%20File%20Handling/2.8.3%20FileDemo.java>

Java Serialization & Deserialization

Refer Example here-

https://github.com/TopsCode/Automation-Testing/blob/master/Module-2/2.8%20File%20Handling/2.8.4%20Serialization_DeserializationDemo.java

Collection in Java

- Collection Framework Introduction
- Collection API
 - ArrayList
 - HashSet
 - Iterator
 - HashMap
 - Vector and Enumeration
- Generics
 - Generics Example
- Comparator and Comparable.
- Vector vs ArrayList
- Hashmap vs Hashtable
- Comparator vs Comparable

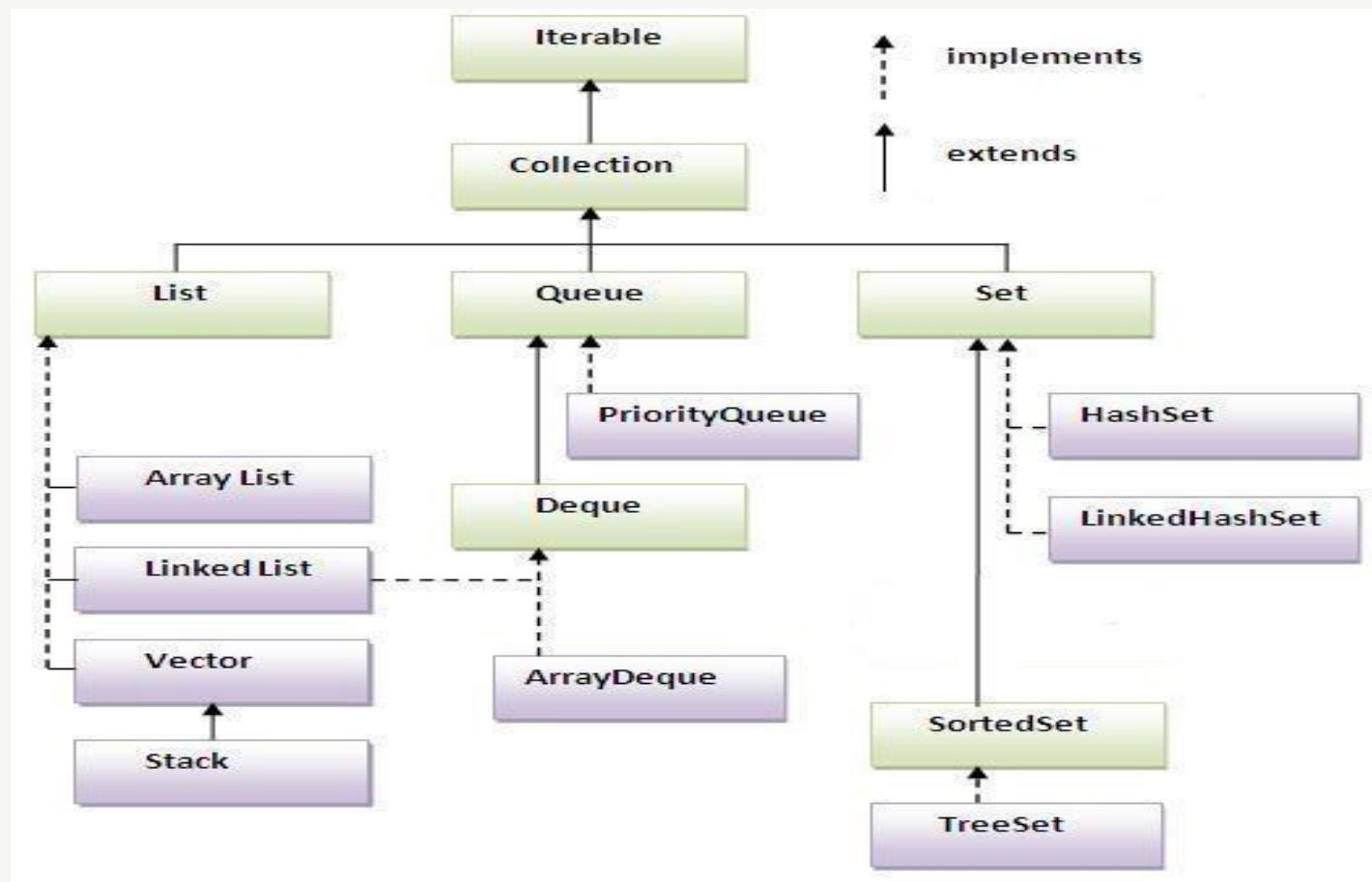
Collection in Java

- Towards this end, the entire collections framework is designed around a set of standard interfaces. Several standard implementations such as **LinkedList**, **HashSet**, and **TreeSet**, of these interfaces are provided that you may use as-is and you may also implement your own collection, if you choose.
- A collections framework is a unified architecture for representing and manipulating collections. All collections frameworks contain the following:
 - **Interfaces:** These are abstract data types that represent collections. Interfaces allow collections to be manipulated independently of the details of their representation. In object-oriented languages, interfaces generally form a hierarchy.

Collection in Java

- **Implementations, i.e., Classes:** These are the concrete implementations of the collection interfaces. In essence, they are reusable data structures.
- **Algorithms:** These are the methods that perform useful computations, such as searching and sorting, on objects that implement collection interfaces. The algorithms are said to be polymorphic: that is, the same method can be used on many different implementations of the appropriate collection interface.
- In addition to collections, the framework defines several map interfaces and classes. Maps store key/value pairs. Although maps are not *collections* in the proper use of the term, but they are fully integrated with collections.

Collection API



Collection in Java

ArrayList

- The ArrayList class extends AbstractList and implements the List interface. ArrayList supports dynamic arrays that can grow as needed.
- Standard Java arrays are of a fixed length. After arrays are created, they cannot grow or shrink, which means that you must know in advance how many elements an array will hold.
- Array lists are created with an initial size. When this size is exceeded, the collection is automatically enlarged. When objects are removed, the array may be shrunk.

Refer Example here-

<https://github.com/TopsCode/Automation-Testing/blob/master/Module-2/2.9%20Collection/2.9.1%20ArrayListDemo.java>

Collection in Java

HashSet

- HashSet extends AbstractSet and implements the Set interface. It creates a collection that uses a hash table for storage.
- A hash table stores information by using a mechanism called hashing. In hashing, the informational content of a key is used to determine a unique value, called its hash code.
- The hash code is then used as the index at which the data associated with the key is stored. The transformation of the key into its hash code is performed automatically.

Refer Example here-

<https://github.com/TopsCode/Automation-Testing/blob/master/Module-2/2.9%20Collection/2.9.2%20HashSetDemo.java>

Collection in Java

Iterator Interface

- Iterator interface will be used for the fetching the element from the collections.
- The easiest way to do this is to employ an iterator, which is an object that implements either the Iterator or the ListIterator interface.
- Iterator enables you to cycle through a collection, obtaining or removing elements. ListIterator extends Iterator to allow bidirectional traversal of a list, and the modification of elements.
- Before you can access a collection through an iterator, you must obtain one. Each of the collection classes provides an iterator() method that returns an iterator to the start of the collection.

Collection in Java

Iterator Interface

- By using this iterator object, you can access each element in the collection, one element at a time.
- In general, to use an iterator to cycle through the contents of a collection, follow these steps:
 - Obtain an iterator to the start of the collection by calling the collection's iterator() method.
 - Set up a loop that makes a call to hasNext(). Have the loop iterate as long as hasNext() returns true.
 - Within the loop, obtain each element by calling next().

Collection in Java

HashMap

- The HashMap class uses a hashtable to implement the Map interface. This allows the execution time of basic operations, such as get() and put(), to remain constant even for large sets.

Refer Example here-

<https://github.com/TopsCode/Automation-Testing/blob/master/Module-2/2.9%20Collection/2.9.3%20HashMapDemo.java>

Collection in Java

Generics

- Java **Generic** methods and generic classes enable programmers to specify, with a single method declaration, a set of related methods or, with a single class declaration, a set of related types, respectively.
- Generics also provide compile-time type safety that allows programmers to catch invalid types at compile time.
- Using Java Generic concept, we might write a generic method for sorting an array of objects, then invoke the generic method with Integer arrays, Double arrays, String arrays and so on, to sort the array elements.

Collection in Java

Generics Method

- You can write a single generic method declaration that can be called with arguments of different types. Based on the types of the arguments passed to the generic method, the compiler handles each method call appropriately. Following are the rules to define Generic Methods:
 - All generic method declarations have a type parameter section delimited by angle brackets (< and >) that precedes the method's return type (< E > in the next example).
 - Each type parameter section contains one or more type parameters separated by commas. A type parameter, also known as a type variable, is an identifier that specifies a generic type name.

Collection in Java

Generics Method

- The type parameters can be used to declare the return type and act as placeholders for the types of the arguments passed to the generic method, which are known as actual type arguments.
- A generic method's body is declared like that of any other method. Note that type parameters can represent only reference types, not primitive types (like int, double and char).

Collection in Java

Collection Framework Without Generic

```
ArrayList myDogList = new ArrayList();
```



row type conversion by compiler

object

object

object

Collection

Collection in Java

Comparator and Comparable

- **Comparator** is capable of comparing two different objects. The method required for implementation is *compare()*.
- **Comparable** is implemented by a class in order to be able to compare object of itself with some other objects. The class itself must implement the interface in order to be able to compare its instance(s). The method required for implementation is *compareTo()*.

Collection in Java

Difference between comparable and comparator

Parameter	Comparable	Comparator
Sorting logic	Sorting logic must be in same class whose objects are being sorted. Hence this is called natural ordering of objects	Sorting logic is in separate class. Hence we can write different sorting based on different attributes of objects to be sorted. E.g. Sorting using id, name etc.
Implementation	Class whose objects to be sorted must implement this interface. e.g Country class needs to implement comparable to collection of country object by id	Class whose objects to be sorted do not need to implement this interface. Some other class can implement this interface. E.g.- CountrySortByIdComparator class can implement Comparator interface to sort collection of country object by id

Collection in Java

Difference between comparable and comparator

Parameter	Comparable	Comparator
Sorting method	<pre>int compareTo(Object o1)</pre> <p>This method compares this object with o1 object and returns a integer. Its value has following meaning.</p>	<pre>int compare(Object o1, Object o2)</pre> <p>This method compares o1 and o2 objects. and returns a integer. Its value has following meaning.</p>
Calling method	<pre>Collections.sort(List)</pre> <p>Here objects will be sorted on the basis of CompareTo method</p>	<pre>Collections.sort(List, Comparator)</pre> <p>Here objects will be sorted on the basis of Compare method in Comparator</p>
	Java.lang.Comparable	Java.util.Comparator

Comparable and Comparator

Comparable Interface:Refer Example here-

<https://github.com/TopsCode/Automation-Testing/blob/master/Module-2/2.9%20Collection/2.9.4%20ComparableDemo.java>

Comparator Interface:Refer Example here-

<https://github.com/TopsCode/Automation-Testing/blob/master/Module-2/2.9%20Collection/2.9.5%20ComparatorDemo.java>

Module – 3 [Web Driver]

- Web Driver Introduction
- Browser Tools
- Web Table
- Junit
- TestNG
- Working with Multiple Browser

Selenium WebDriver

- Selenium suite is comprised of 4 basic components; Selenium IDE, Selenium RC, WebDriver, Selenium Grid.
- WebDriver allows user to perform web based automation testing. WebDriver is a different tool altogether that has various advantages over Selenium RC.
- WebDriver supports a wide range of web browsers, programming languages and test environments.
- WebDriver directly communicates with the web browser and uses its native compatibility to automate.

Selenium WebDriver

- WebDriver's support doesn't only limit in the periphery of traditional user actions. Instead it supports efficient handling mechanisms for complex user actions like dealing with dropdowns, Ajax calls, switching between windows, navigation, handling alerts etc.
- WebDriver enables user to perform web based mobile testing. To support the same, WebDriver introduces AndroidDriver and IphoneDriver.
- WebDriver is faster than other tools of Selenium Suite because it makes direct calls to browser without any external intervention.

Available Drivers

- There are a number of driver classes available in WebDriver, each catering a specific web browser. Each browser has a different driver implementation in WebDriver.
- In WebDriver, a few of the browsers can be automated directly where as some of the web browsers require an external entity to be able to automate and execute the test script. This external entity is known as Driver Server. Thus, user is required to download the Driver Server for different web browsers.
- Notice that there is a separate Driver Server for each of the web browser and user cannot use one Driver Server for web browsers other than the one it is designated for.

Below is the list of available web browsers and their corresponding Server Drivers.

Web-Browser	Driver Server
Mozilla Firefox	No (No external server is required to spin the Firefox browser)
Google Chrome	Yes (ChromeDriver)
Internet Explorer	Yes (Internet Explorer Driver Server)
Opera	Yes (OperaDriver)
Safari	Yes (SafariDriver)
HTML Unit	No (No external entity is required to spin the HTML Unit)

Script Creation

For script creation, we would be using “Learning_Selenium” project created in the previous tutorial and “gmail.com” as the application under test (AUT).

Code Walkthrough

```
import org.openqa.selenium.WebDriver;  
  
import org.openqa.selenium.firefox.FirefoxDriver;  
  
import org.openqa.selenium.WebElement;  
  
import org.openqa.selenium.By;
```

Object Instantiation

```
WebDriver driver = new FirefoxDriver();
```

We create a reference variable for WebDriver interface and instantiate it using FirefoxDriver class. A default Firefox profile will be launched which means that no extensions and plugins would be loaded with the Firefox instance and that it runs in the safe mode.

Launching the Web browser

```
driver.get(appUrl);
```

A `get()` method is called on the WebDriver instance to launch a fresh web browser instance. The string character sequence passed as a parameter into the `get()` method redirects the launched web browser instance to the application URL.

Maximize Browser Window

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

The *maximize()* method is used to maximize the browser window soon after it is redirected to the application URL.

Fetch the page Title

```
driver.getTitle();
```

The `getTitle()` method is used to fetch the title of the current web page. Thus, the fetched title can be loaded to a string variable.

Comparison between Expected and Actual Values:

```
if (expectedTitle.equals(actualTitle)) {
```

```
    System.out.println("Verification Successful - The correct title is displayed on the web  
page."); }
```

```
Else
```

```
{
```

```
    System.out.println("Verification Failed - An incorrect title is displayed on the web  
page.");}
```

The above code uses the conditional statement java constructs to compare the actual value and the expected value. Based on the result obtained, the print statement would be executed.

WebElement Instantiation

WebElement username = driver.findElement(By.id("Email"));

In the above statement, we instantiate the WebElement reference with the help of “*driver.findElement(By.id("Email"))*”. Thus, username can be used to reference the Email textbox on the user interface every time we want to perform some action on it.

Clear Command

```
username.clear();
```

The clear() method/command is used to clear the value present in the textbox if any. It also clears the default placeholder value.

sendKeys Command

```
username.sendKeys("TestSelenium ");
```

The *sendKeys()* method/command is used to enter/type the specified value (within the parentheses) in the textbox. Notice that the *sendKeys()* method is called on the WebElement object which was instantiated with the help of element property corresponding to the UI element.

The above block of code enters the string “TestSelenium” inside the Email textbox on the Gmail application.

sendKeys is one of the most popularly used commands across the WebDriver scripts.

Click Command

SignInButton.click();

Like *sendKeys()*, *click()* is another excessively used command to interact with the web elements. *Click()* command/method is used to click on the web element present on the web page. The above block of code clicks on the “Sign in” button present on the Gmail application.

Notes:

Unlike *sendKeys()* method, *click()* methods can never be parameterized.

At times, clicking on a web element may load a new page altogether. Thus to sustain such cases, *click()* method is coded in a way to wait until the page is loaded.

Close the Web Browser

driver.close();

The close() is used to close the current browser window.

Terminate the Java Program

System.exit(0);

The Exit() method terminates the Java program forcefully. Thus, remember to close all the browser instances prior terminating the Java Program.

Refer Example here-

https://github.com/TopsCode/Automation-Testing/blob/master/Module-3/3.0%20WebDriver_Sample_Program/3.0.1%20WebDriverFirstDemo.java

Test Execution

The test script or simply the java program can be executed in the following ways:

- #1.** Under the Eclipse's menu bar, there is an icon to execute the test script. Refer the following figure.

Locator Types

Locator Type	Syntax	Description
id	driver.findElement (By.id("ID_of_Element"))	Locate by value of the "id" attribute
className	driver.findElement (By.className ("Class_of_Element"))	Locate by value of the "class" attribute
linkText	driver.findElement (By.linkText("Text"))	Locate by value of the text of the hyperlink
partialLinkText	driver.findElement (By.partialLinkText ("PartialText"))	Locate by value of the sub-text of the hyperlink

Locator Types

Locator Type	Syntax	Description
name	driver.findElement (By.name ("Name_of_Element"))	Locate by value of the "name" attribute
xpath	driver.findElement (By.xpath("Xpath"))	Locate by value of the xpath
cssSelector	driver.findElement (By.cssSelector ("CSS Selector"))	Locate by value of the CSS selector
tagName	driver.findElement (By.tagName("input"))	Locate by value of its tag name

Some Important Commands

get()	It automatically opens a new browser window and fetches the page that you specify inside its parentheses.
getTitle()	Needs no parameters Fetches the title of the current page Leading and trailing white spaces are trimmed Returns a null string if the page has no title
getPageSource()	Needs no parameters Returns the source code of the page as a String value
getCurrentUrl()	Needs no parameters Fetches the string representing the current URL that the browser is looking at

Some Important Commands

getText()	Fetches the inner text of the element that you specify

Navigate commands

navigate().to()	<p>It automatically opens a new browser window and fetches the page that you specify inside its parentheses. It does exactly the same thing as the get() method.</p>
navigate().refresh()	<p>Needs no parameters. It refreshes the current page.</p>
navigate().back()	<p>Needs no parameters Takes you back by one page on the browser's history.</p>
navigate().forward()	<p>Needs no parameters Takes you forward by one page on the browser's history.</p>

Closing and Quitting Browser Windows

<code>close()</code>	Needs no parameters It closes only the browser window that WebDriver is currently controlling.
<code>quit()</code>	Needs no parameters It closes all windows that WebDriver has opened.

Selenium-WebDriver's Drivers

WebDriver is the name of the key interface against which tests should be written, but there are several implementations. These include:

HtmlUnit Driver	<p>This is currently the fastest and most lightweight implementation of WebDriver.</p> <p>As the name suggests, this is based on HtmlUnit. HtmlUnit is a java based implementation of a WebBrowser without a GUI.</p> <p>For any language binding (other than java) the Selenium Server is required to use this driver.</p>
	<pre>WebDriver driver = new HtmlUnitDriver();</pre>

Selenium-WebDriver's Drivers

HtmlUnit Driver	
Pros	Cons
<ul style="list-style-type: none">1. Fastest implementation of WebDriver2. A pure Java solution and so it is platform independent.3. Supports JavaScript	Emulates other browsers' JavaScript behaviour

Selenium-WebDriver's Drivers

Firefox Driver	
	<p>Controls the Firefox browser using a Firefox plugin. The Firefox Profile that is used is stripped down from what is installed on the machine to only include the Selenium WebDriver.xpi (plugin). A few settings are also changed by default (see the source to see which ones) Firefox Driver is capable of being run and is tested on Windows, Mac, Linux. Currently on versions 3.6, 10, latest - 1, latest</p>
	<pre>WebDriver driver = new FirefoxDriver();</pre>

Selenium-WebDriver's Drivers

Pros	Cons
<p>Runs in a real browser and supports JavaScript</p> <p>Faster than the <u>Internet Explorer Driver</u></p>	<p>Slower than the <u>HtmlUnit Driver</u></p>

Selenium-WebDriver's Drivers

Internet Explorer Driver	
	This driver is controlled by a .dll and is thus only available on Windows OS. Each Selenium release has its core functionality tested against versions 6, 7 and 8 on XP, and 9 on Windows7.
	WebDriver driver = new InternetExplorerDriver();

Selenium-WebDriver's Drivers

Pros	Cons
Runs in a real browser and supports JavaScript with all the quirks your end users see.	Obviously the Internet Explorer Driver will only work on Windows! Comparatively slow (though still pretty snappy :) XPath is not natively supported in most versions. Sizzle is injected automatically which is significantly slower than other browsers and slower when comparing to CSS selectors in the same browser. CSS is not natively supported in versions 6 and 7. Sizzle is injected instead. CSS selectors in IE 8 and 9 are native, but those browsers don't fully support CSS3

Selenium-WebDriver's Drivers

ChromeDriver	
	<p>ChromeDriver is maintained / supported by the Chromium project itself. WebDriver works with Chrome through the chromedriver binary (found on the chromium project's download page). You need to have both chromedriver and a version of chrome browser installed. chromedriver needs to be placed somewhere on your system's path in order for WebDriver to automatically discover it. The Chrome browser itself is discovered by chromedriver in the default installation path. These both can be overridden by environment variables. Please refer to the wiki for more information.</p>
	<pre>WebDriver driver = new ChromeDriver();</pre>

Selenium-WebDriver's Drivers

Pros	Cons
<p>Runs in a real browser and supports JavaScript</p> <p>Because Chrome is a Webkit-based browser, the <u>ChromeDriver</u> may allow you to verify that your site works in Safari. Note that since Chrome uses its own V8 JavaScript engine rather than Safari's Nitro engine, JavaScript execution may differ.</p>	<p>Slower than the <u>HtmlUnit Driver</u></p>

Accessing Forms using Selenium WebDriver

Accessing Form Elements

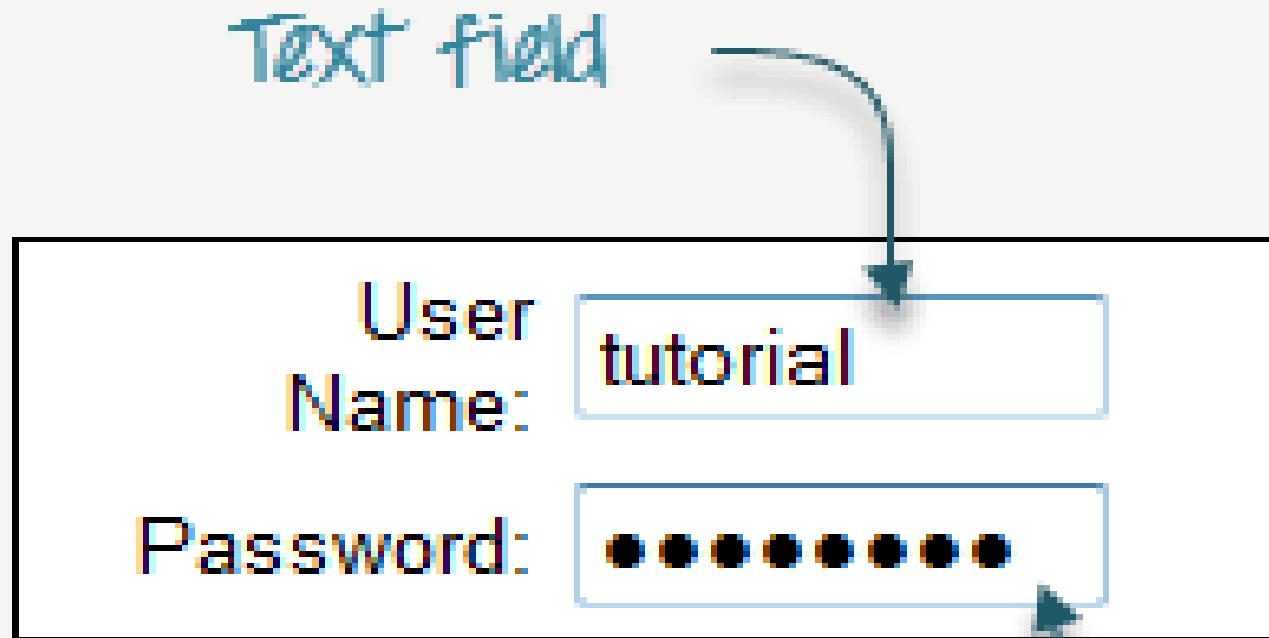
Input Box

Input boxes refer to either of these two types:

Text Fields- text boxes that accept typed values and show them as they are.

Password Fields- text boxes that accept typed values but mask them as a series of special characters (commonly dots and asterisks) to avoid sensitive values to be displayed.

Accessing Forms using Selenium WebDriver



Accessing Forms using Selenium WebDriver

Entering Values in Input Boxes

The **sendKeys()** method is used to enter values into input boxes.

this will type the text "tutorial" onto the element with name="username"

```
driver.findElement(By.name("username")).sendKeys("tutorial");
```

User Name:

Accessing Forms using Selenium WebDriver

Deleting Values in Input Boxes

The **clear()** method is used to delete the text in an input box. **This method does not need any parameter**. The code snippet below will clear out the text "tutorial" in the User Name text box.

```
driver.findElement(By.name("userName")).clear();
```

[Refer Example here-](#)

https://github.com/TopsCode/Automation-Testing/blob/master/Module-3/3.0%20WebDriver_Sample_Program/3.0.2%20WebDriver_Id.java

Accessing Forms using Selenium WebDriver

Radio Button

Toggling a radio button on is done using the **click()** method.

```
driver.findElement(By.cssSelector("input[value='Business']")).click();
```



[Refer Example here-](#)

https://github.com/TopsCode/Automation-Testing/blob/master/Module-3/3.0%20WebDriver_Sample_Program/3.0.3%20WebDriver_Radio.java

Accessing Forms using Selenium WebDriver

Check Box

Toggling a check box on/off is also done using the `click()` method.

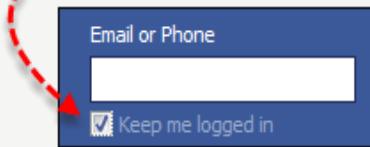
The code below will click on Facebook's "Keep me logged in" check box twice and then output the result as TRUE when it is toggled on, and FALSE if it is toggled off.

Refer Example here-

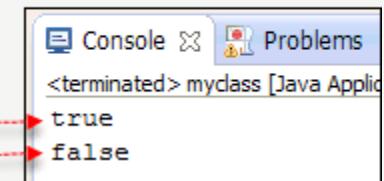
https://github.com/TopsCode/Automation-Testing/blob/master/Module-3/3.0%20WebDriver_Sample_Program/3.0.4%20WebDriver_Check.java

Accessing Forms using Selenium WebDriver

```
public static void main(String[] args) {  
    WebDriver driver = new FirefoxDriver();  
    String baseURL = "http://www.facebook.com";  
  
    driver.get(baseURL);  
    WebElement chkFBPersist = driver.findElement(By.id("persist_box"));  
    for(int i=0; i<2; i++){  
        chkFBPersist.click();  
        System.out.println(chkFBPersist.isSelected());  
    }  
    driver.quit();  
}
```



First click - checkbox
was toggled on



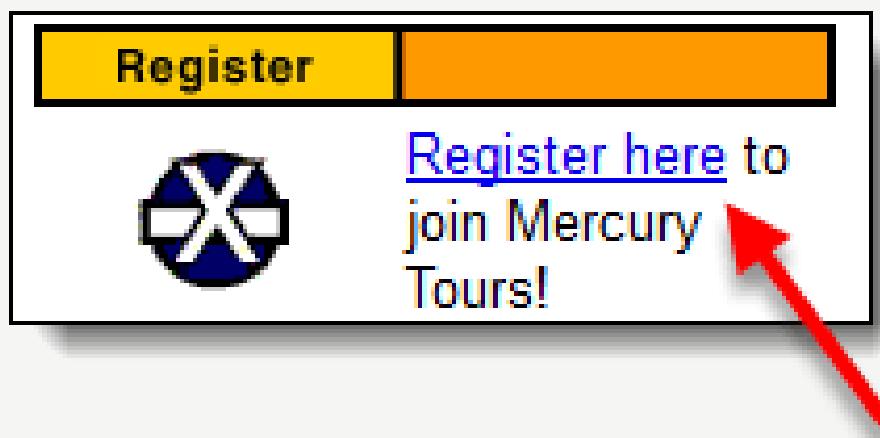
Second click - checkbox
was toggled off

Accessing Forms using Selenium WebDriver

Links

Links also are accessed by using the **click()** method.

Consider the below link found in Mercury Tours' homepage.



Accessing Forms using Selenium WebDriver

You can access this link using linkText() or partialLinkText() together with click(). Either of the two lines below will be able to access the "Register here" link shown above.

```
driver.findElement(By.linkText("Register here")).click();
```

```
driver.findElement(By.partialLinkText("here")).click();
```

Accessing Forms using Selenium WebDriver

Drop-Down Box

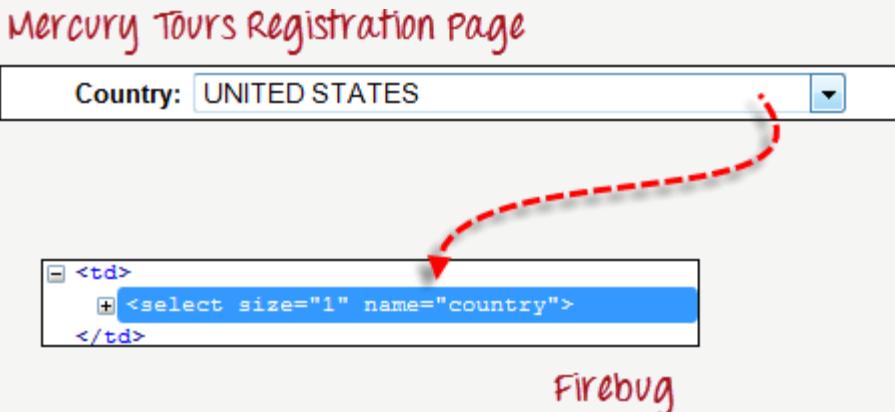
Before we can control drop-down boxes, we must do following two things :

Import the package **org.openqa.selenium.support.ui.Select**

Instantiate the drop-down box as a "Select" object in WebDriver

As an example, go to Mercury Tours' Registration page

(<http://newtours.demoaut.com/mercuryregister.php>) and notice the "Country" drop-down box there.



Accessing Forms using Selenium WebDriver

Selecting Items in a Multiple SELECT element

We can also use the **selectByVisibleText()** method in selecting multiple options in a multi SELECT element. As an example, we will take <http://jsbin.com/osebed/2> as the base URL. It contains a drop-down box that allows multiple selections at a time.

page source

```
<select id="fruits" multiple="">
  <option value="banana">Banana</option>
  <option value="apple">Apple</option>
  <option value="orange">Orange</option>
  <option value="grape">Grape</option>
</select>
```

HTML page



Accessing Forms using Selenium WebDriver

Submitting a Form

The **submit()** method is used to submit a form. This is an alternative to clicking the form's submit button. You can use submit() on any element within the form, not just on the submit button itself.

```
driver.findElement(By.name("userName")).sendKeys("tutorial");
driver.findElement(By.name("password")).sendKeys("tutorial");
driver.findElement(By.name("password")).submit();
```

submit() was used on the Password text box instead of on the sign-in button.

Refer Example here-

https://github.com/TopsCode/Automation-Testing/blob/master/Module-3/3.0%20WebDriver_Sample_Program/3.0.5%20WebDriver_Select.java

Accessing Forms using Selenium WebDriver

When `submit()` is used, WebDriver will look up the DOM to know which form the element belongs to, and then trigger its submit function.

Element	Command	Description
Input Box	<code>sendKeys()</code>	used to enter values onto text boxes
	<code>clear()</code>	used to clear text boxes of its current value
Check Box, Radio Button,	<code>click()</code>	used to toggle the element on/off

Accessing Forms using Selenium WebDriver

Drop-Down Box	<i>selectByVisibleText()/ deselectByVisibleText()</i>	selects/deselects an option by its displayed text
	<i>selectByValue()/ deselectByValue()</i>	selects/deselects an option by the value of its "value" attribute
	<i>selectByIndex()/ deselectByIndex()</i>	selects/deselects an option by its index
	<i>isMultiple()</i>	returns TRUE if the drop-down element allows multiple selection at a time; FALSE if otherwise
	<i>deselectAll()</i>	deselects all previously selected options

Accessing Forms using Selenium WebDriver

Submit Button	<i>submit()</i>	
Links	<i>click()</i>	used to click on the link and wait for page load to complete before proceeding to the next command.

Accessing Links & Tables using Selenium Webdriver

Accessing Links

Links Matching a Criterion

Links can be accessed using an exact or partial match of their link text. The examples below provide scenarios where multiple matches would exist, and would explain how WebDriver would deal with them.

Exact Match

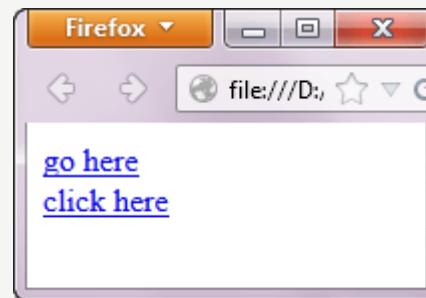
Accessing links using their exact link text is done through the By.linkText() method. However, if there are two links that have the very same link text, this method will only access the first one. Consider the HTML code below

Accessing Links & Tables using Selenium Webdriver

Partial Match

Accessing links using a portion of their link text is done using the **By.partialLinkText()** method. If you specify a partial link text that has multiple matches, only the first match will be accessed. Consider the HTML code below.

```
<html>
  <head>
    <title>Partial Match</title>
  </head>
  <body>
    <a href="http://www.google.com">go here</a>
    <br>
    <a href="http://www.fb.com">click here</a>
  </body>
</html>
```

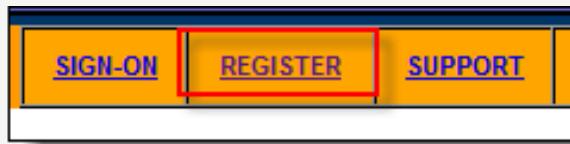


Accessing Links & Tables using Selenium Webdriver

Case-sensitivity

The parameters for **By.linkText()** and **By.partialLinkText()** are both case-sensitive, meaning that capitalization matters. For example, in Mercury Tours' homepage, there are two links that contain the text "egis" - one is the "REGISTER" link found at the top menu, and the other is the "Register here" link found at the lower right portion of the page.

The link at the top menu



The link at the lower right portion of the page



Accessing Links & Tables using Selenium Webdriver

All Links

One of the common procedures in web testing is to test if all the links present within the page are working. This can be conveniently done using a combination of the **Java for-each loop** and the **By.tagName("a")** method. The WebDriver code below checks each link from the Mercury Tours homepage to determine those that are working and those that are still under construction.

Accessing Links & Tables using Selenium Webdriver

Links Outside and Inside a Block

The latest HTML5 standard allows the `<a>` tags to be placed inside and outside of block-level tags like `<div>`, `<p>`, or `<h1>`. The "By.linkText()" and "By.partialLinkText()" methods can access a link located outside and inside these block-level elements. Consider the HTML code below.

```
<body>
  <p>
    <a href="http://www.google.com">Inside a block-level tag.</a>
  </p>

  <br>
  <a href="http://www.fb.com">
    <div>
      <span>Outside a block-level tag.</span>
    </div>
  </a>
</body>
```



Accessing Links & Tables using Selenium Webdriver

Accessing Image Links

Image links are images that act as references to other sites or sections within the same page. Since they are images, we cannot use the By.linkText() and By.partialLinkText() methods because image links basically have no link texts at all. In this case, we should resort to using either By.cssSelector or By.xpath. The first method is more preferred because of its simplicity.

In the example below, we will access the "Facebook" logo on the upper left portion of Facebook's Password Recovery page.

Accessing Links & Tables using Selenium Webdriver



We will use By.cssSelector and the element's "title" attribute to access the image link. And then we will verify if we are taken to Facebook's homepage.

Accessing Links & Tables using Selenium Webdriver

Reading a Table

There are times when we need to access elements (usually texts) that are within HTML tables. However, it is very seldom for a web designer to provide an id or name attribute to a certain cell in the table. Therefore, we cannot use the usual methods such as "By.id()", "By.name()", or "By.cssSelector()". In this case, the most reliable option is to access them using the "By.xpath()" method.

Accessing Links & Tables using Selenium Webdriver

XPath Syntax

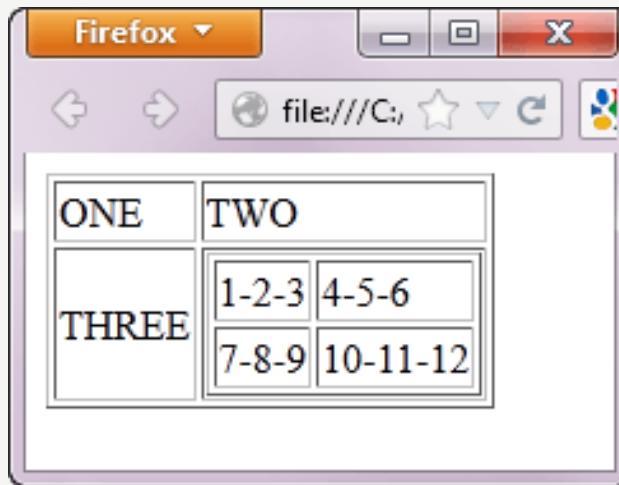
Consider the HTML code below.

```
<html>
  <head>
    <title>Sample</title>
  </head>
  <body>
    <table border="1">
      <tbody>
        <tr>
          <td>first cell</td>
          <td>second cell</td>
        </tr>
        <tr>
          <td>third cell</td>
          <td>fourth cell</td>
        </tr>
      </tbody>
    </table>
  </body>
</html>
```

Accessing Links & Tables using Selenium Webdriver

Accessing Nested Tables

The same principles discussed above applies to nested tables. **Nested tables are tables located within another table.** An example is shown below.



Accessing Links & Tables using Selenium Webdriver

Using Attributes as Predicates

If the element is written deep within the HTML code such that the number to use for the predicate is very difficult to determine, we can use that element's unique attribute instead.

In the example below, the "New York to Chicago" cell is located deep into Mercury Tours homepage's HTML code.

Specials	
Atlanta to Las Vegas	\$398
Boston to San Francisco	\$513
Los Angeles to Chicago	\$168
New York to Chicago	\$198
Phoenix to San Francisco	\$213

Accessing Links & Tables using Selenium Webdriver

In this case, we can use the table's unique attribute (`width="270"`) as the predicate. **Attributes are used as predicates by prefixing them with the @ symbol.** In the example above, the "New York to Chicago" cell is located in the first `<td>` of the fourth `<tr>`, and so our XPath should be as shown below.

```
By.xpath("//table[@width=\"270\"]/tbody/tr[4]/td")
```

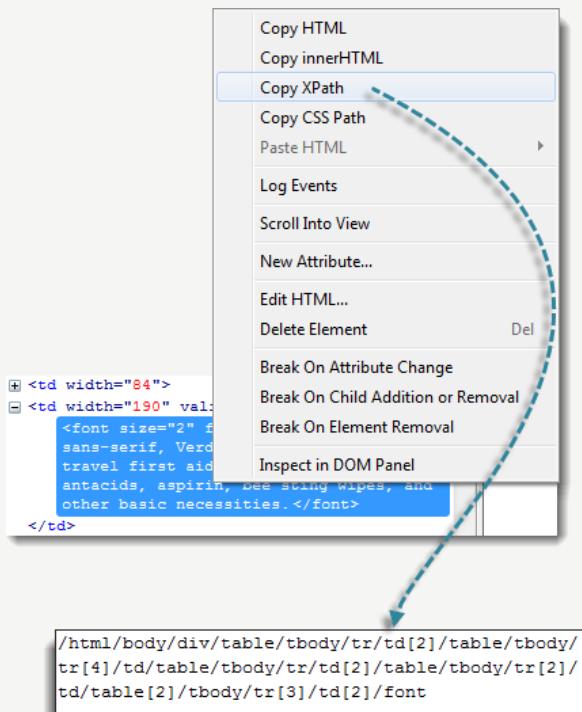


use the escape characters here

Accessing Links & Tables using Selenium Webdriver

Shortcut: Use Firebug

If the number or attribute of an element is extremely difficult or impossible to obtain, the quickest way to generate the XPath code is thru Firebug.



Accessing Links & Tables using Selenium Webdriver

delete these

this is the first
"parent" table element

```
/html/body/div/table/tbody/tr/td[2]/table/tbody/  
tr[4]/td/table/tbody/tr/td[2]/table/tbody/tr[2]/  
td/table[2]/tbody/tr[3]/td[2]/font
```

The remaining portion of the code, trimmed
and prefixed with "://"

```
//table/tbody/tr/td[2]/table/tbody/tr[4]/td/table/  
tbody/tr/td[2]/table/tbody/tr[2]/td/table[2]/tbody/  
tr[3]/td[2]/font
```

```
By.xpath("//table/tbody/tr/td[2]"  
+ "/table/tbody/tr[4]/td"  
+ "/table/tbody/tr/td[2]"  
+ "/table/tbody/tr[2]/td"  
+ "/table[2]/tbody/tr[3]/td[2]/font"))
```

When pasted onto the By.xpath() method

How to write Dynamic XPath in Selenium

Relative XPath method

Using single attribute

// tagname[@attribute-name='value1'+

Example

// a *@href='http://www.google.com'+

//input*@id='name'+

//input*@name='username'+

//img*@alt='sometext'+

How to write Dynamic XPath in Selenium

Using multiple attribute	<code>//tagname*(@attribute1='value1'+'attribute2='value2'+ //a*(@id='id1'+'@name='namevalue1'+ //img*(@src=""+'@href=""+</code>
Using contains method	Syntax <code>//tagname*contains(@attribute,'value1')+ //input*contains(@id,'')+ //input*contains(@name,'')+ //a*contains(@href,'')+ //img*contains(@src,'')+ //div*contains(@id,'')+ //p*contains(text(),'Hello World')+</code>

How to write Dynamic XPath in Selenium

Using starts-with method	//tagname[starts-with(@attribute-name,'')+ //id[starts-with(@id,'')+ //a[starts-with(@href='')+ //img[starts-with(@src='')+ //div[starts-with(@id='')+ //input[starts-with(@id='')+ //button[starts-with(@id,'')+

How to write Dynamic XPath in Selenium

Using Following node	Xpath/following::again-ur-regular-path //input* @id=""+/following::input*1+ //a* @href=""+/following::a*1+ //img* @src=""+/following::img*1+
Using preceding node	Xpath/preceding::again-ur-regular-path //input* @id=""+/ preceding::input*1+ //a[@href=""+/ preceding::a*1+ //img[@src=""+/ preceding::img[1]

Accessing Links & Tables using Selenium Webdriver

Refer Example here-

https://github.com/TopsCode/Automation-Testing/blob/master/Module-3/3.0%20WebDriver_Sample_Program/3.0.6%20WebDriver_Table.java

Implicit Wait

Selenium WebDriver has borrowed the idea of **implicit waits** from **Watir**. This means that we can tell Selenium that we would like it to wait for a certain amount of time before throwing an **exception** that it cannot find the element on the page. We should note that implicit waits will be in place for the entire time the browser is open. This means that any search for elements on the page could take the time the implicit wait is set for.

Refer Example here-

https://github.com/TopsCode/Automation-Testing/blob/master/Module-3/3.0%20WebDriver_Sample_Program/3.0.7%20ImplicitWaitDemo.java

Explicit Wait

It is more extendible in the means that you can set it up to wait for any condition you might like. Usually, you can use some of the prebuilt **ExpectedConditions** to wait for elements to become clickable, visible, invisible, etc.

Fluent Wait

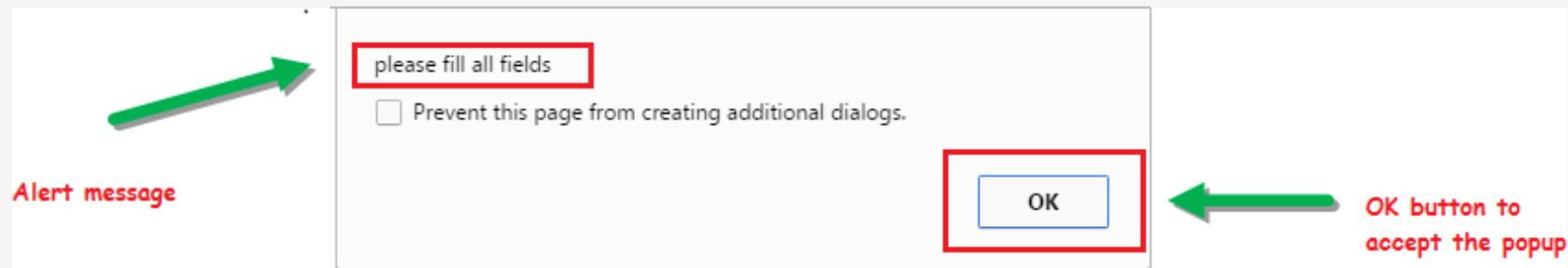
Each **FluentWait** instance defines the maximum amount of time to wait for a condition, as well as the frequency with which to check the condition. Furthermore, the user may configure the wait to ignore specific types of exceptions whilst waiting, such as **NoSuchElementExceptions** when searching for an element on the page.

Handling alerts box, Verifying Alert texts and popups

What is Alert?

Alert is a small message box which displays on-screen notification to give the user some kind of information or ask for permission to perform certain kind of operation. It may be also used for warning purpose.

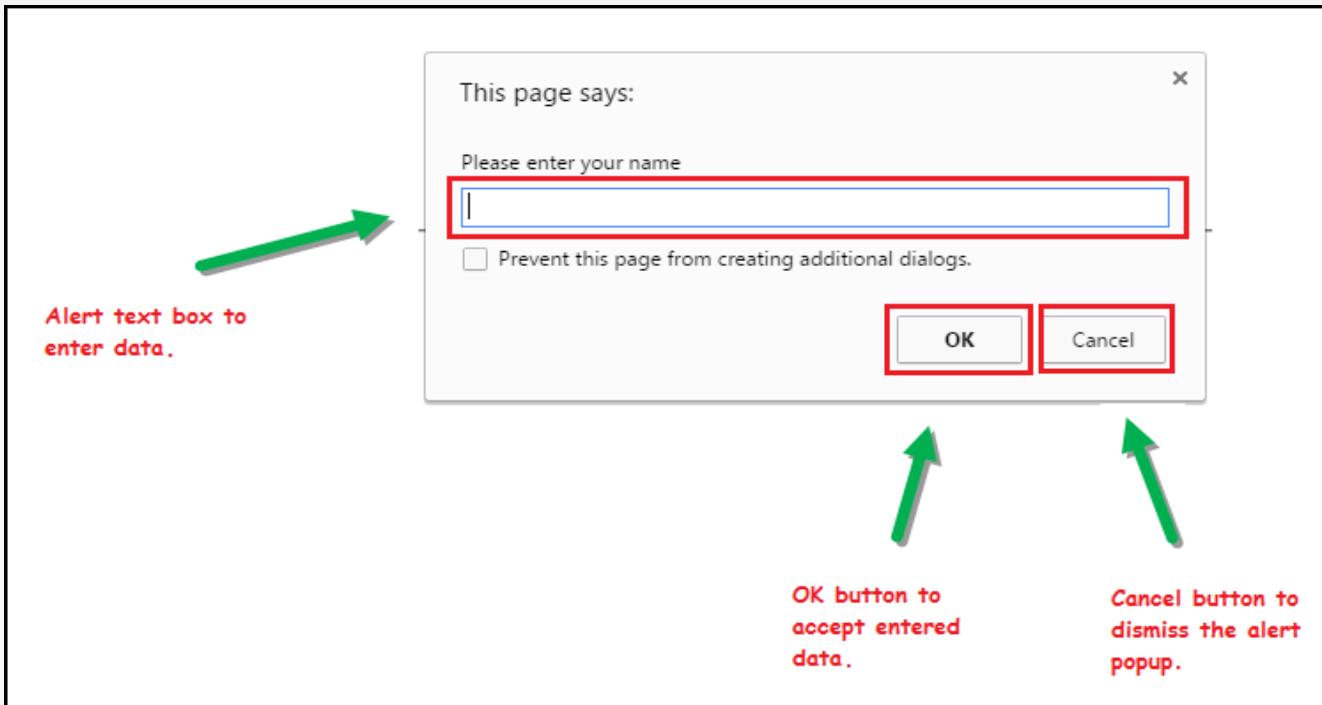
1) Simple Alert-



Handling alerts box, Verifying Alert texts and popups

2) Prompt Alert-

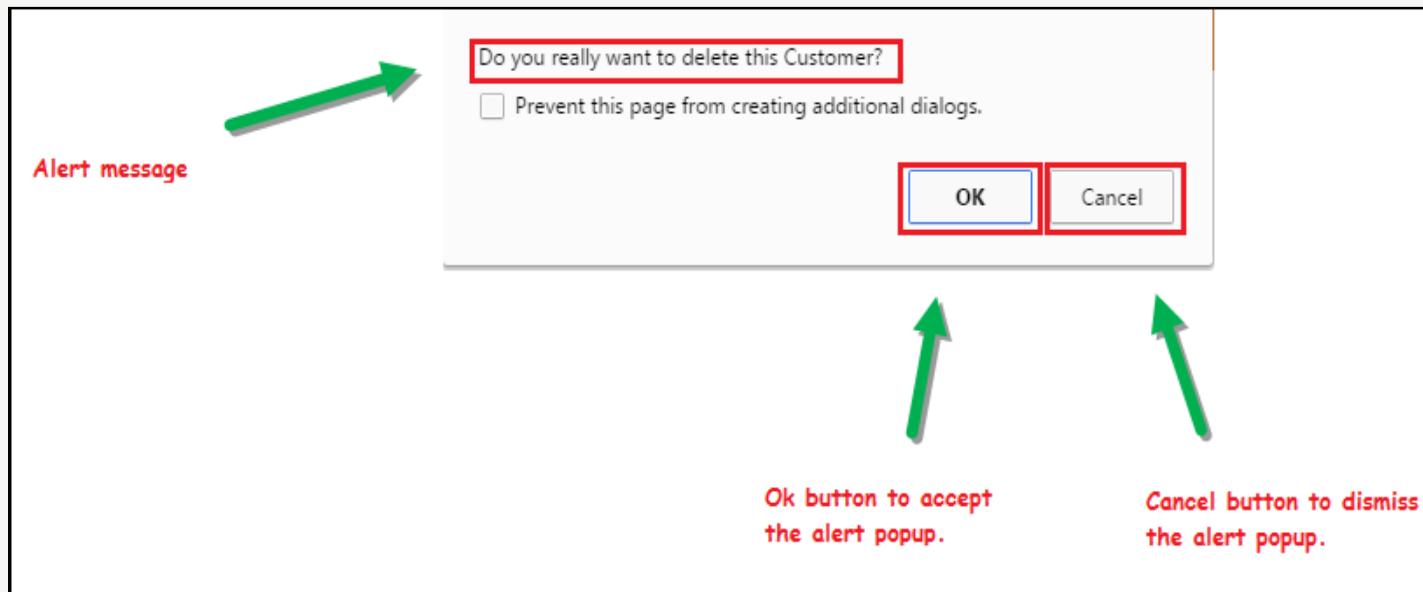
This Prompt Alert asks some input from the user and selenium webdriver can enter the text using sendkeys(" input.... ").



Handling alerts box, Verifying Alert texts and popups

3) Confirmation Alert.

This confirmation alert asks permission to do some type of operation.



Handling alerts box, Verifying Alert texts and popups

How to handle Alert in Selenium WebDriver

Alert interface provides the below few methods which are widely used in Selenium Webdriver.

1) **void dismiss() // To click on the 'Cancel' button of the alert.**

```
driver.switchTo().alert().dismiss();
```

2) **void accept() // To click on the 'OK' button of the alert.**

```
driver.switchTo().alert().accept();
```

Handling alerts box, Verifying Alert texts and popups

How to handle Alert in Selenium WebDriver

3) **String getText() // To capture the alert message.**

```
driver.switchTo().alert().getText();
```

4) **void sendKeys(String stringToSend) // To send some data to alert box.**

```
driver.switchTo().alert().sendKeys("Text");
```

You can see a number of Alert methods are displayed as shown in below screen suggested by Eclipse.

We can easily switch to alert from the main window by using Selenium's **.switchTo()** method.

Handling alerts box, Verifying Alert texts and popups

How to handle Selenium Pop-up window using Webdriver

In automation, when we have multiple windows in any web application, the activity may need to switch control among several windows from one to other in order to complete the operation. After completion of the operation, it has to return to the main window i.e. parent window.

In selenium web driver there are methods through which we can handle multiple windows.

Driver.getWindowHandles();

Handling Alert and Popups Example

Refer Example for Handling Alert here-

https://github.com/TopsCode/Automation-Testing/blob/master/Module-3/3.0%20WebDriver_Sample_Program/3.0.10%20WebDriver_HandlingAlerts.java

Refer Example for Handling Popups here-

https://github.com/TopsCode/Automation-Testing/blob/master/Module-3/3.0%20WebDriver_Sample_Program/3.0.11%20WebDriver_HandlingPopUps.java

Mouse Click & Keyboard Event: Action Class in Selenium Webdriver

Handling special keyboard and mouse events are done using the **Advanced User Interactions API**. It contains the **Actions** and the **Action** classes that are needed when executing these events.

Method	Description
<code>clickAndHold()</code>	Clicks (without releasing) at the current mouse location.
<code>contextClick()</code>	Performs a context-click at the current mouse location. (Right Click Mouse Action)
<code>doubleClick()</code>	Performs a double-click at the current mouse location.
<code>dragAndDrop(source, target)</code>	Performs click-and-hold at the location of the source element, moves to the location of the target element, then releases the mouse.

Parameters:

source- element to emulate button down at.

target- element to move to and release the mouse at.

Mouse Click & Keyboard Event: Action Class in Selenium Webdriver

dragAndDropBy(source, x-offset, y-offset) Performs click-and-hold at the location of the source element, moves by a given offset, then releases the mouse.

Parameters:

source- element to emulate button down at.

xOffset- horizontal move offset.

yOffset- vertical move offset.

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)

Mouse Click & Keyboard Event: Action Class in Selenium Webdriver

moveByOffset(x-offset, y-offset) Moves the mouse from its current position (or 0,0) by the given offset.

Parameters:

x-offset- horizontal offset. A negative value means moving the mouse left.

y-offset- vertical offset. A negative value means moving the mouse down.

moveToElement(toElement) Moves the mouse to the middle of the element.

Parameters:

toElement- element to move to.

release() Releases the depressed left mouse button at the current mouse location

sendKeys(onElement, charsequence) Sends a series of keystrokes onto the element.

Parameters:

onElement - element that will receive the keystrokes, usually a text field

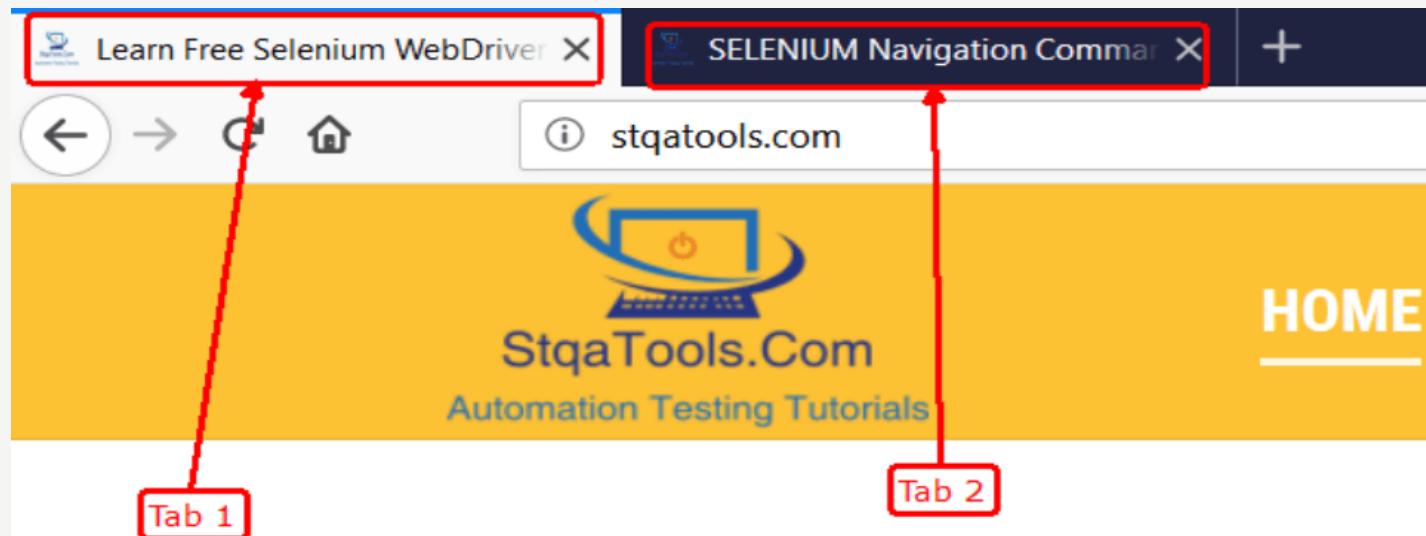
charsequence - any string value representing the sequence of keystrokes to be sent

Mouse and Keyboard actions Example

https://github.com/TopsCode/Automation-Testing/blob/master/Module-3/3.0%20WebDriver_Sample_Program/3.0.12%20WebDriver_KeyboardMouseAction.java

Handling Multiple Tabs in Selenium Webdriver

1. Tab is Similar to Window there is no difference with regards to Selenium.
2. New tab content is the same as automating the [window](#).
3. WebDriver Software Automation Testing Tool does not have any built-in method that we can use to open new tabs.
4. We can do the same thing to open a new tab in the Selenium WebDriver.



Handling Multiple Tabs in Selenium Webdriver Example

Ways to Window Tab Handle Using Selenium:

Switch between two tabs using switchTo()

https://github.com/TopsCode/Automation-Testing/blob/master/Module-3/3.0%20WebDriver_Sample_Program/3.0.13%20WebDriver_HAndlingMultipleTabs_switchTo.java

Switch between two tabs using sendKeys Actions

https://github.com/TopsCode/Automation-Testing/blob/master/Module-3/3.0%20WebDriver_Sample_Program/3.0.14%20WebDriver_Handlin gMultipleTabs_Actions.java

How to Identify Web Elements Using Selenium Xpath and Other Locators

What is Locator?

- Locator can be termed as an address that identifies a web element uniquely within the webpage. Locators are the HTML properties of a web element which tells the Selenium about the web element it need to perform action on.

How to Identify Web Elements Using Selenium Xpath and Other Locators

There is a diverse range of web elements. The most common amongst them are:

- Text box
- Button
- Drop Down
- Hyperlink
- Check Box
- Radio Button

How to Identify Web Elements Using Selenium Xpath and Other Locators

Types of Locators:

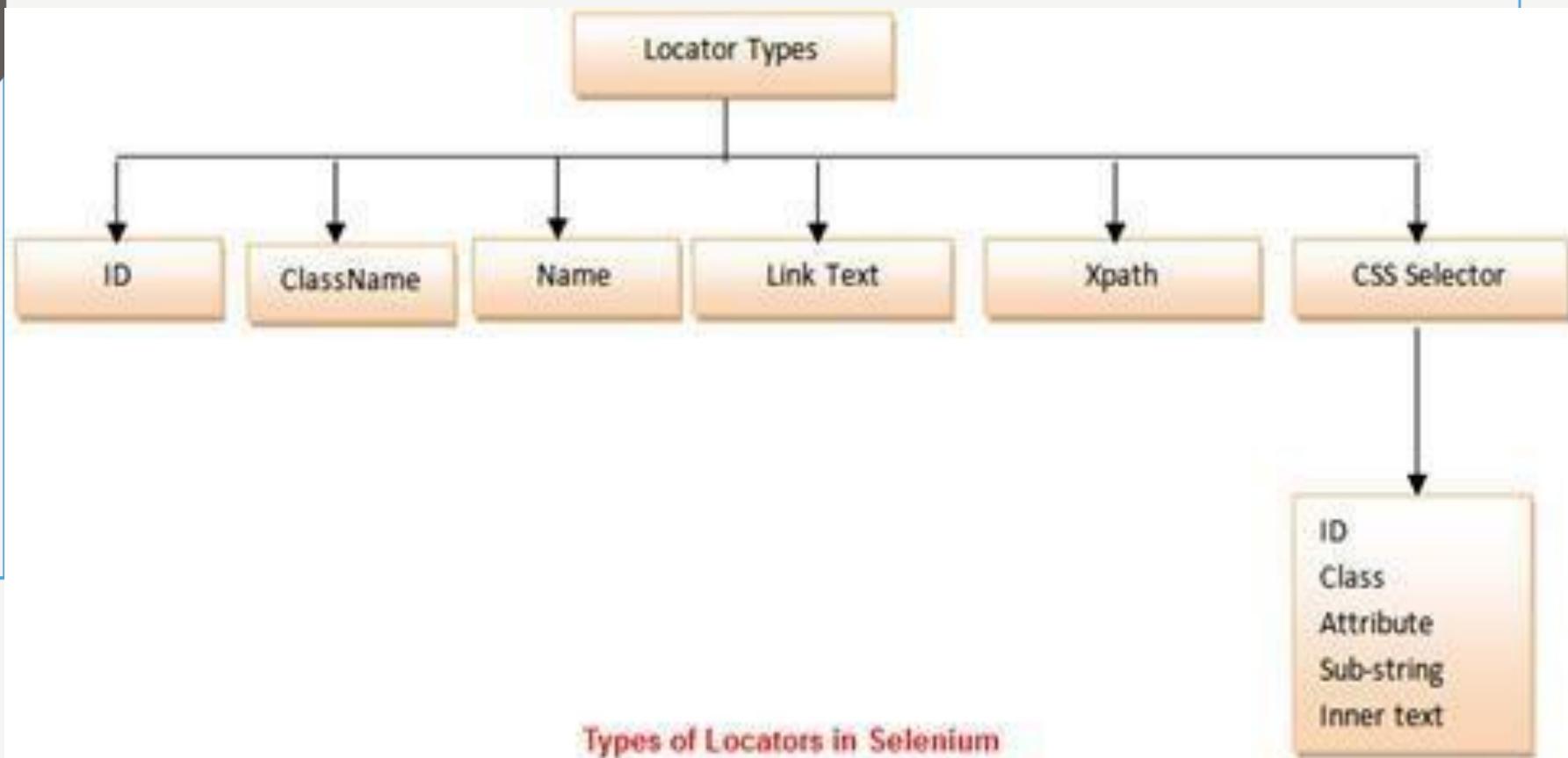
Identifying these elements has always been a very tricky subject and thus it requires an accurate and effective approach.

Thereby, we can assert that more effective the locator, more stable will be the automation script.

Essentially every Selenium command requires locators to find the web elements.

Thus, to identify these web elements accurately and precisely we have different types of locators.

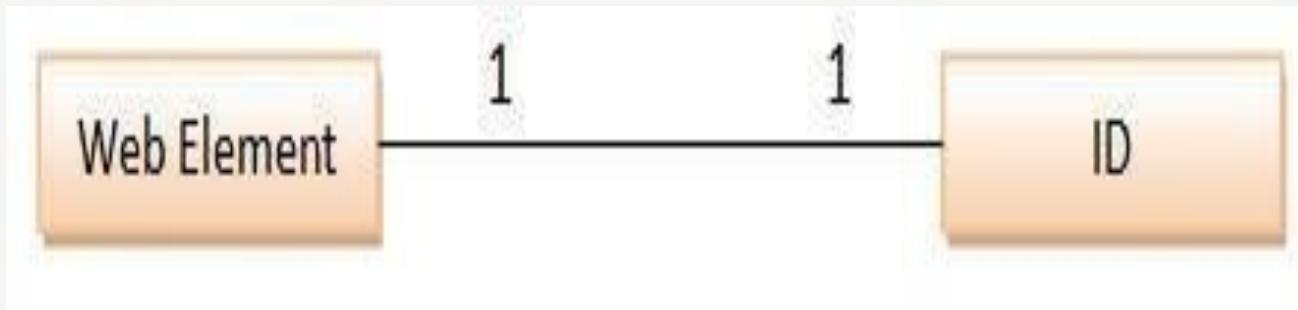
How to Identify Web Elements Using Selenium Xpath and Other Locators



How to Identify Web Elements Using Selenium Xpath and Other Locators

Using ID as a Locator

- The best and the most popular method to identify web element is to use ID. The ID of each element is alleged to be unique.



How to Identify Web Elements Using Selenium Xpath and Other Locators

Using ClassName as a Locator

There is only a subtle difference between using ID as a locator and using classname as a locator.

In this sample, we would access “Need Help?” hyperlink present at the bottom of the login form at gmail.com.

How to Identify Web Elements Using Selenium Xpath and Other Locators

Using name as a Locator

Locating a web element using name is very much analogous to previous two locator types. The only difference lies in the syntax.

In this sample, we would access “Password” text box present in the login form at gmail.com.

Syntax: name = name of the element, In our case, the name is “Passwd”.

Verify the locator value

- **Step 1:** Type “name= Passwd” in the target box and click on the Find Button. Notice that the “Password” textbox would be highlighted.

Using Link Text as a Locator

All the hyperlinks on a web page can be identified.

([Some Text](#)) :-

Used to create the hyperlinks on a web page and the text between opening and closing of anchor tags constitutes the link text

Using Link Text as a Locator

In this sample, we would access “Create an account” link present at the bottom of the login form at gmail.com.

Finding a link text of a web element using Firebug

- **Step 1:** Locate / inspect the web element (“Create an account” link in our case) by right clicking on the web element whose locator value we need to inspect and clicking on the option “Inspect Element with Firebug”.
- **Step 2:** Be cognizant about the text present within the `<a> ` tags and take a note of it. Hence this text will be used to identify the link on a web page uniquely.

Using Xpath as a Locator

Xpath is used to locate a web element based on its XML path. XML stands for Extensible Markup Language and is used to store, organize and transport arbitrary data. It stores data in a key-value pair which is very much similar to HTML tags. Both being mark up languages and since they fall under the same umbrella, xpath can be used to locate HTML elements.

Using Xpath as a Locator

Key Points:

The success rate of finding an element using Xpath is too high. Along with the previous statement, Xpath can find relatively all the elements within a web page. Thus, Xpaths can be used to locate elements having no id, class or name.

Creating a valid Xpath is a tricky and complex process. There are plug-ins available to generate Xpath but most of the times, the generated Xpaths fails to identify the web element correctly.

While creating xpath, user should be aware of the various nomenclatures and protocols.

Using Xpath as a Locator

Selenium Xpath Examples

Xpath Checker

Creating Xpath becomes a little simpler by using Xpath Checker. Xpath Checker is a firefox add-on to automatically generate Xpath for a web element. The add-on can be downloaded and installed like any other plug-in. The plug-in can be downloaded from "<https://addons.mozilla.org/en-US/firefox/addon/xpath-checker/>".

As soon as the plug-in is installed, it can be seen in the context menu by right clicking any element whose Xpath we want to generate.

Locator

Note that the Xpath Checker is available for other browsers as well.

But re-iterating the fact, that most of the times, the generated Xpaths fails to identify the web element rightly. Thus, it is recommended to create our own Xpath following the pre defined rules and protocols.

In this sample, we would access “Google” image present at the top of the login form at gmail.com.

Creating a Xpath of a web element

- **Step 1:** Type “//img*@class='logo'+” i.e. the locator value in the target box within the Selenium IDE.

X-Path Example

https://github.com/TopsCode/Automation-Testing/blob/master/Module-3/3.0%20WebDriver_Sample_Program/3.0.9%20WebDriver_Xpath.java

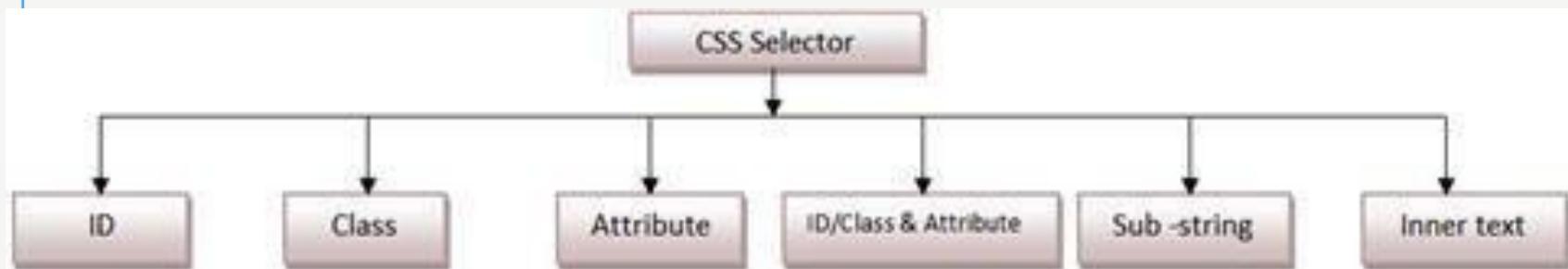
Using CSS Selector as a Locator:

CSS Selector is combination of an element selector and a selector value which identifies the web element within a web page. The composite of element selector and selector value is known as Selector Pattern.

Selector Pattern is constructed using HTML tags, attributes and their values. The central theme behind the procedure to create CSS Selector and Xpath are very much similar underlying the only difference in their construction protocol. Like Xpath, CSS selector can also locate web elements having no ID, class or Name.

So now gearing ahead, let us discuss the primitive types of CSS Selectors:

Using CSS Selector as a Locator:



CSS Selector: ID

In this sample, we would access “Email” text box present in the login form at Gmail.com.

The Email textbox has an ID attribute whose value is defined as “Email”. Thus ID attribute and its value can be used to create CSS Selector to access the email textbox.

Creating CSS Selector for web element

- **Step 1:** Locate / inspect the web element (“Email” textbox in our case) and notice that the html tag is “input” and value of ID attribute is “Email” and both of them collectively make a reference to the “Email Text box”. Hence the above data would be used to create CSS Selector.

Verify the locator value

Syntax: css=<HTML tag><#><Value of ID attribute>

HTML tag – It is tag which is used to denote the web element which we want to access.

– The hash sign is used to symbolize ID attribute. It is mandatory to use hash sign if ID attribute is being used to create CSS Selector.

Value of ID attribute – It is the value of an ID attribute which is being accessed. The value of ID is always preceded by a hash sign.

While specifying CSS Selector in the target text box of Selenium IDE, always remember to prefix it with “css=”. The sequence of the above artifacts is inalterable. If two or more web elements have the same HTML tag and attribute value, the first element marked in the page source will be identified.

CSS Selector: Class

We would access “Stay signed in” check box present below the login form at gmail.com.

The “Stay signed in” check box has a Class attribute whose value is defined as “remember”. Thus Class attribute and its value can be used to create CSS Selector to access the designated web element.

Locating an element using Class as a CSS Selector is very much similar to using ID, the lone difference lies in their syntax formation.

Creating CSS Selector for web element

- **Step 1:** Locate / inspect the web element (“Stay signed in” check box in our case) and notice that the html tag is “label” and value of ID attribute is “remember” and both of them collectively make a reference to the “Stay signed in check box”.

CSS Selector: Attribute

In this sample, we would access “Sign in” button present below the login form at gmail.com.

The “Sign in” button has a type attribute whose value is defined as “submit”. Thus type attribute and its value can be used to create CSS Selector to access the designated web element.

Creating CSS Selector for web element

- **Step 1:** Locate / inspect the web element (“Sign in” button in our case) and notice that the html tag is “input”, attribute is type and value of type attribute is “submit” and all of them together make a reference to the “Sign in” button.

Verify the locator value

Syntax

css=<HTML tag><[attribute=Value of attribute]>

Attribute – It is the attribute we want to use to create CSS Selector. It can value, type, name etc. It is recommended to choose an attribute whose value uniquely identifies the web element.

Value of attribute – It is the value of an attribute which is being accessed.

CSS Selector: ID/Class and attribute

In this sample, we would access “Password” text box present in the login form at gmail.com.

CSS Selector: Sub-string

CSS in Selenium allows matching a partial string and thus deriving a very interesting feature to create CSS Selectors using sub strings. There are three ways in which CSS Selectors can be created based on mechanism used to match the sub string.

Types of mechanisms

All the underneath mechanisms have symbolic significance.

- Match a prefix
- Match a suffix
- Match a sub string

Match a prefix

It is used to correspond to the string with the help of a matching prefix.

Syntax

`css=<HTML tag><[attribute^=prefix of the string]>`

`^` – Symbolic notation to match a string using prefix.

Prefix – It is the string based on which match operation is performed. The likely string is expected to start with the specified string.

For Example: Let us consider “Password textbox”, so the corresponding CSS Selector would be:

`css=input#Passwd*name^='Pass'`

Match a Suffix

Match a suffix

It is used to correspond to the string with the help of a matching suffix.

css=<HTML tag><[attribute\$=suffix of the string]>

– Symbolic notation to match a string using suffix.

Suffix – It is the string based on which match operation is performed. The likely string is expected to ends with the specified string.

For Example: Lets again consider “Password textbox”, so the corresponding CSS Selector would be:

css=input#Passwd[name\$='wd'+

Match a sub string

It is used to correspond to the string with the help of a matching sub string.

Syntax:-css=<HTML tag><[attribute*=sub string]>

* – Symbolic notation to match a string using sub string.

Sub string – It is the string based on which match operation is performed. The likely string is expected to have the specified string pattern.

For Example: Lets again consider “Password textbox”, so the corresponding CSS Selector would be:

css=input#Passwd[name\$='wd'+

CSS Selector: Inner text

Inner text helps us identify and create CSS Selector using a string pattern that the HTML Tag manifests on the web page.

Consider, “Need help?” hyperlink present below the login form at gmail.com.

The anchor tag representing the hyperlink has a text enclosed within. Thus this text can be used to create CSS Selector to access the designated web element.

CSS Selector: Inner text

Syntax

css=<HTML tag><:><contains><(text)>

:- The dot sign is used to symbolize contains method

Contains – It is the value of a Class attribute which is being accessed.

Text – The text that is displayed anywhere on the web page irrespective of its location.

This is one of the most frequently used strategies to locate web element because of its simplified syntax.

Owing to the fact that creating CSS Selector and Xpath requires a lot of efforts and practice, thus the process is only exercised by more sophisticated and trained users.

CSS Selector: Inner text

Matching Text Patterns

Like locators, *patterns* are a type of parameter frequently required by Selenese commands.

Examples of commands which require are **verifyTextPresent**, **verifyTitle**, **verifyAlert**, **assertConfirmation**, and **verifyPrompt**.

And as has been mentioned above, link locators can utilize a pattern. Patterns allow you to *describe*, via the use of special characters, what text is expected rather than having to specify that text exactly.

Globbing Patterns

There are three types of patterns: *globbing*, *regular expressions*, and *exact*.

Most people are familiar with globbing as it is utilized in filename expansion at a DOS or Unix/Linux command line such as ls *.c. In this case, globbing is used to display all the files ending with a .c extension that exist in the current directory. Globbing is fairly limited. Only two special characters are supported in the Selenium implementation:

- * which translates to “match anything,” i.e., nothing, a single character, or many characters.

Globbing Patterns

[] (*character class*) which translates to “match any single character found inside the square brackets.” A dash (hyphen) can be used as a shorthand to specify a range of characters (which are contiguous in the ASCII character set). A few examples will make the functionality of a character class clear:

[aeiou] matches any lowercase vowel

[0-9] matches any digit

[a-zA-Z0-9] matches any alphanumeric character

Globbing Patterns

In most other contexts, globbing includes a third special character, the ?. However, Selenium globbing patterns only support the asterisk and character class.

To specify a globbing pattern parameter for a Selenese command, you can prefix the pattern with a **glob:** label. However, because globbing patterns are the default, you can also omit the label and specify just the pattern itself.

Globbing Patterns

Below is an example of two commands that use globbing patterns. The actual link text on the page being tested was “Film/Television Department”; by using a pattern rather than the exact text, the **click** command will work even if the link text is changed to “Film & Television Department” or “Film and Television Department”. The glob pattern’s asterisk will match “anything or nothing” between the word “Film” and the word “Television”.

Command	Target
click	link=glob:Film*Television Department
verifyTitle	glob:*Film*Television*

Globbing Patterns

The actual title of the page reached by clicking on the link was “De Anza Film And Television Department - Menu”. By using a pattern rather than the exact text, the verifyTitle will pass as long as the two words “Film” and “Television” appear (in that order) anywhere in the page’s title.

For example, if the page’s owner should shorten the title to just “Film & Television Department,” the test would still pass.

Using a pattern for both a link and a simple test that the link worked (such as the verifyTitle above does) can greatly reduce the maintenance for such test cases.

Regular Expression Patterns

- *Regular expression* patterns are the most powerful of the three types of patterns that Selenese supports.
- Regular expressions are also supported by most high-level programming languages, many text editors, and a host of tools, including the Linux/Unix command-line utilities **grep**, **sed**, and **awk**.
- In Selenese, regular expression patterns allow a user to perform many tasks that would be very difficult otherwise.
- Whereas Selenese globbing patterns support only the * and [] (character class) features, Selenese regular expression patterns offer the same wide array of special characters that exist in JavaScript. Below are a subset of those special characters:

Exact Patterns

Exact Patterns

The **exact** type of Selenium pattern is of marginal usefulness. It uses no special characters at all. So, if you needed to look for an actual asterisk character (which is special for both globbing and regular expression patterns), the **exact** pattern would be one way to do that.

For example, if you wanted to select an item labeled “Real *” from a dropdown, the following code might work or it might not. The asterisk in the `glob:Real *` pattern will match anything or nothing. So, if there was an earlier select option labeled “Real Numbers,” it would be the option selected rather than the “Real *” option.

Exact Patterns

select//selectglob:Real *In order to ensure that the “Real *” item would be selected, the exact: prefix could be used to create an **exact** pattern as shown below:

select//selectexact:Real *But the same effect could be achieved via escaping the asterisk in a regular expression pattern:

select//selectregexp:Real *It’s rather unlikely that most testers will ever need to look for an asterisk or a set of square brackets with characters inside them (the character class for globbing patterns). Thus, globbing patterns and regular expression patterns are sufficient for the vast majority of us.

CSS Selector Example

https://github.com/TopsCode/Automation-Testing/blob/master/Module-3/3.0%20WebDriver_Sample_Program/3.0.8%20WebDriver_CSS.java

Commonly Used Selenium Commands

Open	opens a page using a URL.
click/clickAndWait	performs a click operation, and optionally waits for a new page to load.
verifyTitle/assertTitle	verifies an expected page title.
verifyTextPresent	verifies expected text is somewhere on the page.
verifyElementPresent	verifies an expected UI element, as defined by its HTML tag, is present on the page.
verifyText	verifies expected text and its corresponding HTML tag are present on the page.

Commonly Used Selenium Commands

verifyTable	verifies a table's expected contents.
waitForPageToLoad	pauses execution until an expected new page loads. Called automatically when clickAndWait is used.
waitForElementPresent	pauses execution until an expected UI element, as defined by its HTML tag, is present on the page.

Verifying Page Elements

Verifying UI elements on a web page is probably the most common feature of your automated tests. Selenese allows multiple ways of checking for UI elements. It is important that you understand these different methods because these methods define what you are actually testing.

Verifying Page Elements

For example, will you test that...

- an element is present somewhere on the page?
- specific text is somewhere on the page?
- specific text is at a specific location on the page?

For example, if you are testing a text heading, the text and its position at the top of the page are probably relevant for your test.

Verifying Page Elements

If, however, you are testing for the existence of an image on the home page, and the web designers frequently change the specific image file along with its position on the page, then you only want to test that *an image* (as opposed to the specific image file) exists *somewhere on the page*.

Assertion or Verification?

Choosing between “assert” and “verify” comes down to convenience and management of failures.

If you’re not on the correct page, you’ll probably want to abort your test case so that you can investigate the cause and fix the issue(s) promptly.

Verifying Page Elements

On the other hand, you may want to check many attributes of a page without aborting the test case on the first failure as this will allow you to review all failures on the page and take the appropriate action.

Effectively an “assert” will fail the test and abort the current test case, whereas a “verify” will fail the test and continue to run the test case.

Verifying Page Elements

The best use of this feature is to logically group your test commands, and start each group with an “assert” followed by one or more “verify” test commands.

An example follows:

Command	Target	Value
open	/download/	
assertTitle	Downloads	
verifyText	//h2	Downloads
assertTable	1.2.1	Selenium IDE
verifyTable	1.2.2	June 3, 2008
verifyTable	1.2.3	1.0 beta 2

Verifying Page Elements

The above example first opens a page and then “asserts” that the correct page is loaded by comparing the title with the expected value. Only if this passes will the following command run and “verify” that the text is present in the expected location.

The test case then “asserts” the first column in the second row of the first table contains the expected value, and only if this passed will the remaining cells in that row be “verified”.

Verifying Text Elements

verifyTextPresent

- The command `verifyTextPresent` is used to verify *specific text exists somewhere on the page*. It takes a single argument—the text pattern to be verified. For example:

Command	Target	Value
<code>verifyTextPresent</code>	Marketing Analysis	

Verifying Text Elements

This would cause Selenium to search for, and verify, that the text string “Marketing Analysis” appears somewhere on the page currently being tested. Use `verifyTextPresent` when you are interested in only the text itself being present on the page. Do not use this when you also need to test where the text occurs on the page.

`verifyElementPresent`

Use this command when you must test for the presence of a specific UI element, rather than its content. This verification does not check the text, only the HTML tag. One

Command	Target	Value
<code>verifyElementPresent</code>	<code>//div/p/img</code>	

Verifying Text Elements

This command verifies that an image, specified by the existence of an `` HTML tag, is present on the page, and that it follows a `<div>` tag and a `<p>` tag. The first (and only) parameter is a *locator* for telling the Selenese command how to find the element. Locators are explained in the next section.

`verifyElementPresent` can be used to check the existence of any HTML tag within the page. You can check the existence of links, paragraphs, divisions `<div>`, etc. Here are a few more examples.

Verifying Text Elements

Command	Target	Value
verifyElementPresent	//div/p	
verifyElementPresent	//div/a	
verifyElementPresent	id=Login	
verifyElementPresent	link=Go to Marketing Research	
verifyElementPresent	//a[2]	
verifyElementPresent	//head/title	

Verifying Text Elements

These examples illustrate the variety of ways a UI element may be tested. Again, locators are explained in the next section.

verifyText

Use `verifyText` when both the text and its UI element must be tested. `verifyText` must use a locator. If you choose an *XPath* or *DOM* locator, you can verify that specific text appears at a specific location on the page relative to other UI components on the page.

Command	Target	Value
<code>verifyText</code>	<code>//table/tr/td/div/p</code>	This is my text and it occurs right after the div inside the table.

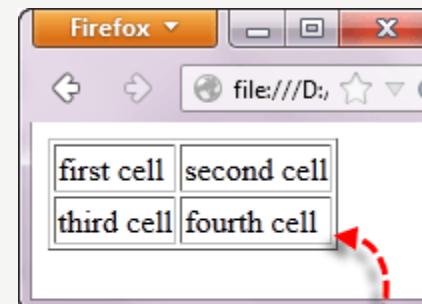
Handling Web Table in Selenium WebDriver

How to write XPath for Table

Consider the HTML code below.

```
<html>
  <head>
    <title>Sample</title>
  </head>
  <body>
    <table border="1">
      <tbody>
        <tr>
          <td>first cell</td>
          <td>second cell</td>
        </tr>
        <tr>
          <td>third cell</td>
          <td>fourth cell</td>
        </tr>
      </tbody>
    </table>
  </body>
</html>
```

We will use XPath to get the inner text of the cell containing the text "fourth cell."

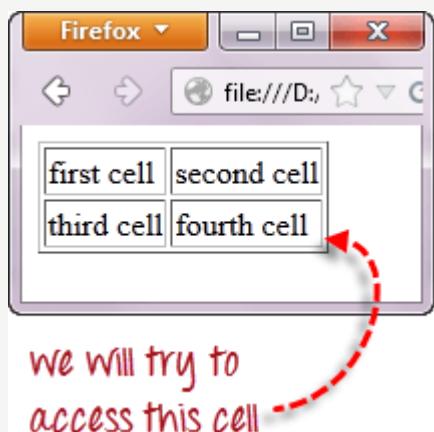


we will try to
access this cell

Handling Web Table in Selenium WebDriver

How to write XPath for Table

We will use [XPath](#) to get the inner text of the cell containing the text "fourth cell."

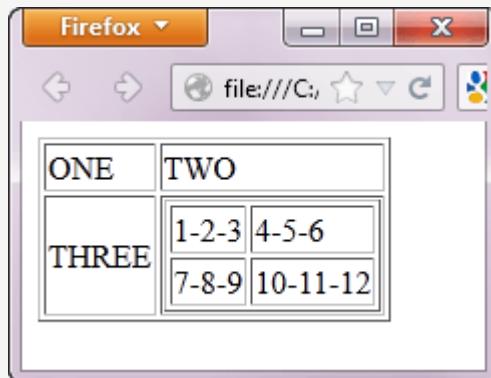


```
public static void main(String[] args) {  
    String baseUrl = "file:///C:/newhtml.html";  
    WebDriver driver = new FirefoxDriver();  
  
    driver.get(baseUrl);  
    String innerText = driver.findElement(  
        By.xpath("//table/tbody/tr[2]/td[2]")).getText();  
    System.out.println(innerText);  
    driver.quit();  
}
```

Handling Web Table in Selenium WebDriver

How to write XPath for Nested Tables-

The same principles discussed above applies to nested tables. Nested tables are tables located within another table. An example is shown here.



```
<html>
  <head>
    <title>Sample</title>
  </head>
  <body>
    <!--outer table-->
    <table border="1">
      <tbody>
        <tr>
          <td>ONE</td>
          <td>TWO</td>
        </tr>
        <tr>
          <td>THREE</td>
          <td>
            <!--inner table-->
            <table border="1">
              <tbody>
                <tr>
                  <td>1-2-3</td>
                  <td>4-5-6</td>
                </tr>
                <tr>
                  <td>7-8-9</td>
                  <td>10-11-12</td>
                </tr>
              </tbody>
            </table>
          </td>
        </tr>
      </tbody>
    </table>
  </body>
</html>
```

Handling Web Table in Selenium WebDriver

How to write XPath for Nested Tables

To access the cell with the text "4-5-6" using the "//parent/child" and predicate concepts from the previous section, we should be able to come up with the XPath code below.

The WebDriver code below should be able to retrieve the inner text of the cell which we are accessing.

The outer table

The inner table

//table/tbody/tr[2]/td[2]/table/tbody/tr/td[2]

```
public static void main(String[] args) {
    String baseUrl = "file:///C:/newhtml.html";
    WebDriver driver = new FirefoxDriver();

    driver.get(baseUrl);
    String innerText = driver.findElement(By
        .xpath("//table/tbody/tr[2]/td[2]/table/tbody/tr/td[2]"))
        .getText();
    System.out.println(innerText);
    driver.quit();
}
```

Handling Web Table in Selenium WebDriver Example

https://github.com/TopsCode/Automation-Testing/blob/master/Module-3/3.0%20WebDriver_Sample_Program/3.0.15%20Webdriver_WebTable.java

JUNIT

- JUnit is a unit testing framework for the Java programming language.
- JUnit is an open source framework which is used for writing & running tests.
- JUnit promotes the idea of "testing first" rather than implementing first.
- JUnit provides Annotation to identify the test methods and utility methods to assert expected results.
- JUnit has Test runners for running tests.
- JUnit tests can be organized into test suites which contains test cases and other test suites.
- JUnit Framework can be easily integrated with either of the followings:
 - Eclipse
 - Ant
 - Maven

JUnit Features

JUnit test framework provides following important features

- Fixtures
- Test suites
- Test runners
- JUnit classes

JUnit Set up

Refer this Link :

<https://github.com/TopsCode/Automation-Testing/blob/master/Module-3/3.1%20JUnit/3.1.1%20JUnit%20SetUp/3.1.1%20JUnit%20SetUp.docx>

JUnit Assert

```
public class Assert extends java.lang.Object
```

This class provides a set of assertion methods useful for writing tests.

Only failed assertions are recorded.

JUnit Assert class Methods

`void assertEquals(expected, result);`
test whether two arrays are equal to each other.

`void assertEquals(boolean expected, boolean actual)`
Check that two primitives/Objects are equal

`void assertFalse(boolean condition)`
Check that a condition is false

`void assertNotNull(Object object)`
Check that an object isn't null.

`void assertNotSame(boolean condition)`
tests if two object references not point to the same object

JUnit Assert class Methods

`void assertNull(Object object)`

Check that an object is null

`void assertSame(boolean condition)`

tests if two object references point to the same object

`void assertTrue(boolean condition)`

Check that a condition is true.

`void fail()`

Fails a test with no message.

[Link for Example](#)

<https://github.com/TopsCode/Automation-Testing/blob/master/Module-3/3.1%20JUnit/3.1.2%20JUnit%20Assert%20Class/3.1.2.1%20JUnit%20Assert%20Class.java>

JUnit TestCase Methods

public abstract class TestCase extends Assert implements Test

int countTestCases()

Counts the number of test cases.

TestResult createResult()

Creates a default TestResult object.

String getName()

Gets the name of a TestCase.

TestResult run()

A convenience method to run this test.

JUnit TestCase Methods

`void run(TestResult result)`

Runs the test case and collects the results in TestResult.

`void setName(String name)`

Sets the name of a TestCase.

`void setUp()`

Sets up the fixture, for example, open a database connection.

`void tearDown()`

Tears down the fixture, for example, close a database connection.

`String toString()`

Returns a string representation of the test case.

Link for Examples of Methods

<https://github.com/TopsCode/Automation-Testing/blob/master/Module-3/3.1%20JUnit/3.1.3%20JUnit%20Test%20Cases/3.1.3%20JUnit%20Test%20Cases.java>

TestResult Class

public class TestResult extends Object

Follow Link-

<https://github.com/TopsCode/Automation-Testing/blob/master/Module-3/3.1%20JUnit/3.1.4%20JUnit%20Test%20Result/3.1.4.1%20JUnit%20Test%20Result.java>

TestSuite Class

A TestSuite is a Composite of Tests. It runs a collection of test cases.

Links for Examples

<https://github.com/TopsCode/Automation-Testing/blob/master/Module-3/3.1%20JUnit/3.1.5%20JUnit%20Test%20Suite%20class/3.1.5.1%20Test%20Suite%20class.java>

JUnit Annotation

Annotations are meta-tags we can use to mark methods or classes.

We can use annotations from JUnit to mark methods. For example, we can use `@Before` to mark a method to run before the test cases.

Annotation	Description
<code>@Test</code>	marks the public void method as a test case.
<code>@Before</code>	Annotating a method with <code>@Before</code> causes that method to be run before each Test method.
<code>@After</code>	Annotating a method with <code>@After</code> causes that method to be run after the Test method.

Annotation

Annotation	Description
@BeforeClass	Annotating a method with @BeforeClass causes it to be run once before any of the test methods in the class.
@AfterClass	call the method after all tests have finished.
@Ignore	marks to ignore the test and that test will not be executed.

Links for Examples

<https://github.com/TopsCode/Automation-Testing/blob/master/Module-3/3.1%20JUnit/3.1.6%20JUnit%20Annotations/3.1.6.1%20JUnit%20Annotations.java>

JUnit Exceptions Test

Junit can trace the Exception.

We can test whether the code throws desired exception or not.

The expected parameter is used along with @Test annotation.

Links for Examples

<https://github.com/TopsCode/Automation-Testing/blob/master/Module-3/3.1%20JUnit/3.1.7%20JUnit%20Exceptions%20Test/3.1.7.1%20JUnit%20Exceptions%20Test.java>

JUnit Parameterized Test

unit 4 has introduced a new feature called Parameterized tests.

We can use Parameterized tests to run the same test using different values.

Links for Examples

<https://github.com/TopsCode/Automation-Testing/blob/master/Module-3/3.1%20JUnit/3.1.8%20JUnit%20Parameterized%20Test/3.1.8.1%20JUnit%20Parameterized%20Test.java>

TestNG(Testing of Next generation)

Introduction

- TestNG is a testing framework inspired from JUnit and NUnit. But introducing some new functionality that make it more powerful and easier to use.
- It is an open source automated testing framework; where NG of TestNG means Next Generation. TestNG is similar to JUnit but it is much more powerful than JUnit but still it's inspired by JUnit. It is designed to be better than JUnit, especially when testing integrated classes. Pay special thanks to Cedric Beust who is the creator of TestNG.
- This eliminates most of the limitations of the older framework. It provides developer the ability to write more flexible and powerful tests with help of easy annotations, grouping, sequencing & parametrizing.

Advantages of TestNG over JUnit

- Annotations are easier to understand
- Test cases can be grouped more easily
- Parallel testing is possible
- **Annotations in TestNG are lines of code that can control how the method below them will be executed.**
- They are always preceded by the @ symbol. A very early and quick example is the one shown below.

These are 2 examples of annotations

```
@Test(priority = 0)
public void goToHomepage() {
    driver.get(baseUrl);
    Assert.assertEquals(driver.getTitle(), "Welcome: Mercury Tours");
}

@Test(priority = 1)
public void logout() {
    driver.findElement(By.linkText("SIGN-OFF")).click();
    Assert.assertEquals("Sign-on: Mercury Tours", driver.getTitle());
}
```

The example above simply says that the method `goToHomepage()` should be executed first before `logout()` because it has a lower priority number

TestNG Test Suite

- In TestNG framework, we need to create **testng.xml** file to create and handle multiple test classes.
- This is the xml file where you will configure your test run, set test dependency, include or exclude any test, method, class or package and set priority etc.
- Implement Excel with TestNG Data Provider.

TestNG Annotations

- In the TestNG Introduction chapter we have came across different annotations used in TestNG Framework but so far we have used just three(Before, After & Test).
- All though these are the most frequently used annotations but who know how far you will go with your framework and may like to use other useful TestNG annotations.

TestNG Annotations

Sr.No.	Annotation & Description
1	@BeforeSuite The annotated method will be run only once before all tests in this suite have run.
2	@AfterSuite The annotated method will be run only once after all tests in this suite have run.
3	@BeforeClass The annotated method will be run only once before the first test method in the current class is invoked.
4	@AfterClass The annotated method will be run only once after all the test methods in the current class have run.
5	@BeforeTest The annotated method will be run before any test method belonging to the classes inside the <test> tag is run.

TestNG Annotations

Sr.No.	Annotation & Description
6	@AfterTest The annotated method will be run after all the test methods belonging to the classes inside the <test> tag have run.
7	@BeforeGroups The list of groups that this configuration method will run before. This method is guaranteed to run shortly before the first test method that belongs to any of these groups is invoked.
8	@AfterGroups The list of groups that this configuration method will run after. This method is guaranteed to run shortly after the last test method that belongs to any of these groups is invoked.
9	@BeforeMethod The annotated method will be run before each test method.
10	@AfterMethod The annotated method will be run after each test method.

TestNG Annotations

No.	Annotation & Description
11	@DataProvider Marks a method as supplying data for a test method. The annotated method must return an Object[][], where each Object[] can be assigned the parameter list of the test method. The @Test method that wants to receive data from this DataProvider needs to use a dataProvider name equals to the name of this annotation.
12	@Factory Marks a method as a factory that returns objects that will be used by TestNG as Test classes. The method must return Object[].
13	@Listeners Defines listeners on a test class.
14	@Parameters Describes how to pass parameters to a @Test method.
15	@Test Marks a class or a method as a part of the test.

TestNG Annotations

Benefits of Using Annotations-

- TestNG identifies the methods it is interested in, by looking up annotations. Hence, method names are not restricted to any pattern or format.
- We can pass additional parameters to annotations.
- Annotations are strongly typed, so the compiler will flag any mistakes right away.
- Test classes no longer need to extend anything (such as TestCase, for JUnit 3).

Link for Example

<https://github.com/TopsCode/Automation-Testing/blob/master/Module-3/3.2%20TestNG/3.2.4%20TestNG%20Annotations/3.2.4.1%20TestNG%20Annotations.java>

TestNG - Ignore a Test

- Sometimes, it happens that our code is not ready and the test case written to test that method/code fails.
- In such cases, annotation **@Test(enabled = false)** helps to disable this test case.
- If a test method is annotated with **@Test(enabled = false)**, then the test case that is not ready to test is bypassed.
- Now, let's see **@Test(enabled = false)** in action.

For Example refer this link-

<https://github.com/TopsCode/Automation-Testing/blob/master/Module-3/3.2%20TestNG/3.2.5%20TestNG%20Ignore%20a%20Test/3.2.5.1TestNG%20Ignore%20a%20Test.java>

TestNG - Dependency Test

- Sometimes, you may need to invoke methods in a test case in a particular order, or you may want to share some data and state between methods.
- This kind of dependency is supported by TestNG, as it supports the declaration of explicit dependencies between test methods.
- TestNG allows you to specify dependencies either with –
 - Using attribute *dependsOnMethods* in @Test annotations, OR.
 - Using attribute *dependsOnGroups* in @Test annotations.

For Example refer link-

<https://github.com/TopsCode/Automation-Testing/blob/master/Module-3/3.2%20TestNG/3.2.6.1%20TestNG%20Dependancy%20Test.java>

TestNG – Parameterized Test

- Another interesting feature available in TestNG is **parametric testing**.
- In most cases, you'll come across a scenario where the business logic requires a hugely varying number of tests.
- **Parameterized tests** allow developers to run the same test over and over again using different values.
- TestNG lets you pass parameters directly to your test methods in two different ways –
 - With testng.xml
 - With Data Providers

For Example Refer link-

<https://github.com/TopsCode/Automation-Testing/blob/master/Module-3/3.2%20TestNG/3.2.7%20TestNG%20Parameterized%20Test/3.2.7.1%20TestNG%20Parameterized%20Test.java>

TestNG Prioritizing & Sequencing

- There will be situations when you want to put number of tests under a single test class and like to run all in single shot.
- With the help of TestNG '@Test' annotations we can execute multiple tests in single TestNG file.

Link for Example

<https://github.com/TopsCode/Automation-Testing/blob/master/Module-3/3.2%20TestNG/3.2.8%20TestNG%20Prioritizing%20and%20Sequencing/3.2.8.1%20TestNG%20Prioritizing%20and%20Sequencing.java>

TestNG Data Providers

- When you need to pass complex parameters or parameters that need to be created from Java (complex objects, objects read from a property file or a database, etc...), in such cases parameters can be passed using Dataproviders.
- A Data Provider is a method annotated with @DataProvider.
- A Data Provider returns an array of objects.

How to do it...

Link

Follow this steps to do

<https://github.com/TopsCode/Automation-Testing/blob/master/Module-3/3.2%20TestNG/3.2.9%20TestNG%20Data%20Provider/3.2.9.1%20TestNG%20Data%20Provider.java>

TestNG Data Providers

Example Link

https://github.com/TopsCode/Automation-Testing/blob/master/Module-3/3.2%20TestNG/3.2.9%20TestNG%20Data%20Provider/3.2.9.2%20TestNG_Dataprovider_Example/test-output/index.html

Working with Multiple Browsers(Cross browser Testing)

Cross Browser Testing is a type of functional test to check that your web application works as expected in different browsers.

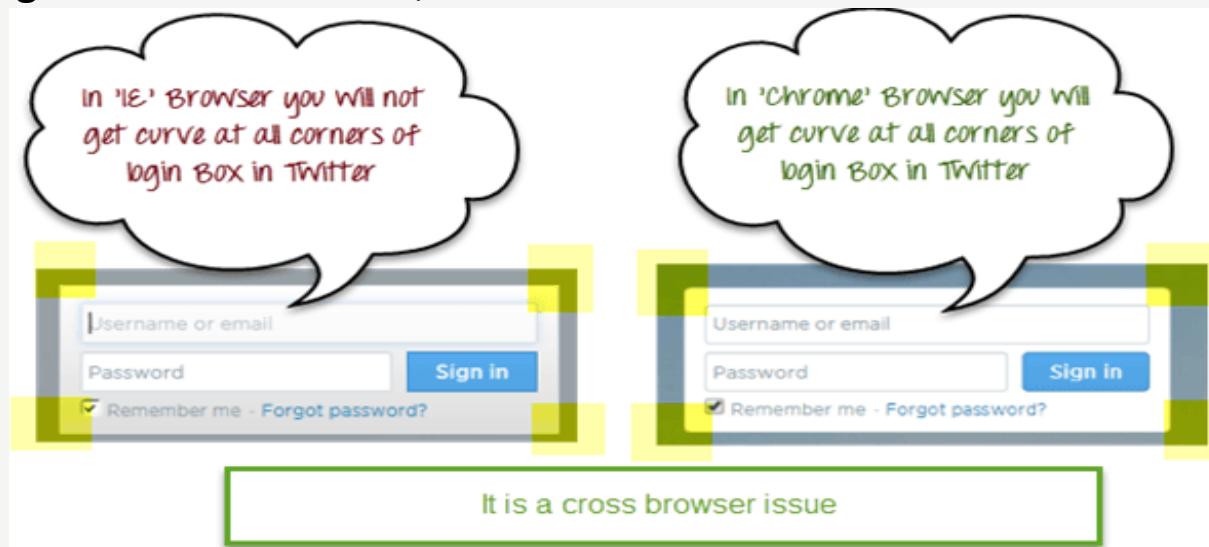


Working with Multiple Browsers(Cross browser Testing)

Why do we need Cross Browser Testing?

A web application can be opened in any browser by the end user. For example, some people prefer to open <http://twitter.com> in **Firefox browser**, while others can be using **Chrome browser** or **IE**.

In the diagram below you can observe that in **IE**, the login box of Twitter is not showing curve at all corners, but we are able to see it in **Chrome browser**.



Working with Multiple Browsers(Cross browser Testing)

So we need to ensure that the web application will work as expected in all popular browsers so that more people can access it and use it.

This motive can be fulfilled with Cross Browser Testing of the product.

Reason Cross Browser Issues

1. Font size mismatch in different browsers.
2. JavaScript implementation can be different.
3. CSS,HTML validation difference can be there.
4. Some browser still not supporting HTML5.
5. Page alignment and div size.
6. Image orientation.
7. Browser incompatibility with OS. Etc.

Working with Multiple Browsers(Cross browser Testing)

How to perform Cross Browser Testing

To execute test cases with different browsers in the same machine at the same time we can integrate Testng framework with Selenium WebDriver.

Your **testing.xml** will look like that,

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite name="TestSuite" thread-count="2" parallel="tests" >
  <test name="ChromeTest">
    <parameter name="browser" value="Chrome" />
    <classes>
      <class name="parallelTest.CrossBrowserScript">
        </class>
    </classes>
  </test>
  <test name="FirefoxTest">
    <parameter name="browser" value="Firefox" />
    <classes>
      <class name="parallelTest.CrossBrowserScript">
        </class>
    </classes>
  </test>
</suite>
```

Test will run parallel

1st Test name is 'ChromeTest'

Parameter is 'Chrome'

2st Test name is 'FirefoxTest'

Parameter is 'Firefox'

This is the TestNGxml for cross Browser Testing

Working with Multiple Browsers(Cross browser Testing)

This testing.xml will map with the Test Case which will look like that

Parameter will be pass from testNG.xml

```
@BeforeTest  
@Parameters("browser")  
public void setup(String browser) throws Exception{  
    if(browser.equalsIgnoreCase("firefox")){  
        driver = new FirefoxDriver();  
    }  
    else if(browser.equalsIgnoreCase("chrome")){  
        System.setProperty("webdriver.chrome.driver",".\\chromedriver.exe");  
        driver = new ChromeDriver();  
    }  
}
```

check parameter value and create WebDriver according it

Here because the testing.xml has two Test tags ('ChromeTest', 'FirefoxTest'), this test case will execute two times for 2 different browsers.

First Test 'ChromeTest' will pass the value of parameter 'browser' as 'chrome' so ChromeDriver will be executed. This test case will run on Chrome browser.

Second Test 'FirefoxTest' will pass the value of parameter 'browser' as 'Firefox' so FirefoxDriver will be executed. This test case will run on FireFox browser.

Working with Multiple Browsers(Cross browser Testing)

Example Link

https://github.com/TopsCode/Automation-Testing/blob/master/Module-3/3.2%20TestNG/3.2.10%20CrossBrowserTesting_MultipleBrowserDemo.java

Testng.xml file Link-

<https://github.com/TopsCode/Automation-Testing/blob/master/Module-3/3.2%20TestNG/3.2.10.1%20testng.xml>

NOTE: To run the test, Right click on the **testng.xml**, Select Run As, and Click TestNG

Module -4 [Automation Frameworks]

- What is Framework
- Types of Framework
- Data Driven Framework
- Modular Driven Framework
- Keyword driven Framework
- Hybrid Framework

Apache POI (Excel)

- Most commercial automated software tools on the market support some sort of data driven testing, which allows you to automatically run a test case multiple times with different input and validation values.
- As Selenium Webdriver is more an automated testing framework than a ready-to-use tool, you will have to put in some effort to support data driven testing in your automated tests.
- Use Microsoft Excel as the format for storing parameters.
- An additional advantage of using Excel is that you can easily outsource the test data administration to someone other than yourself, someone who might have better knowledge of the test cases that need to be run and the parameters required to execute them.

Overview of Testing framework

What is Testing Framework:

- A testing framework or more specifically a testing automation framework is an execution environment for automated tests. It is the overall system in which the tests will be automated.
- It is defined as the set of assumptions, concepts, and practices that constitute a work platform or support for automated testing.

Testing framework is responsible for

- Defining the format in which to express expectations.
- Creating a mechanism to hook into or drive the application under test
- Executing the tests
- Reporting results

Properties of a testing framework:

- It is application independent.
- It is easy to expand, maintain and perpetuate/keep going.

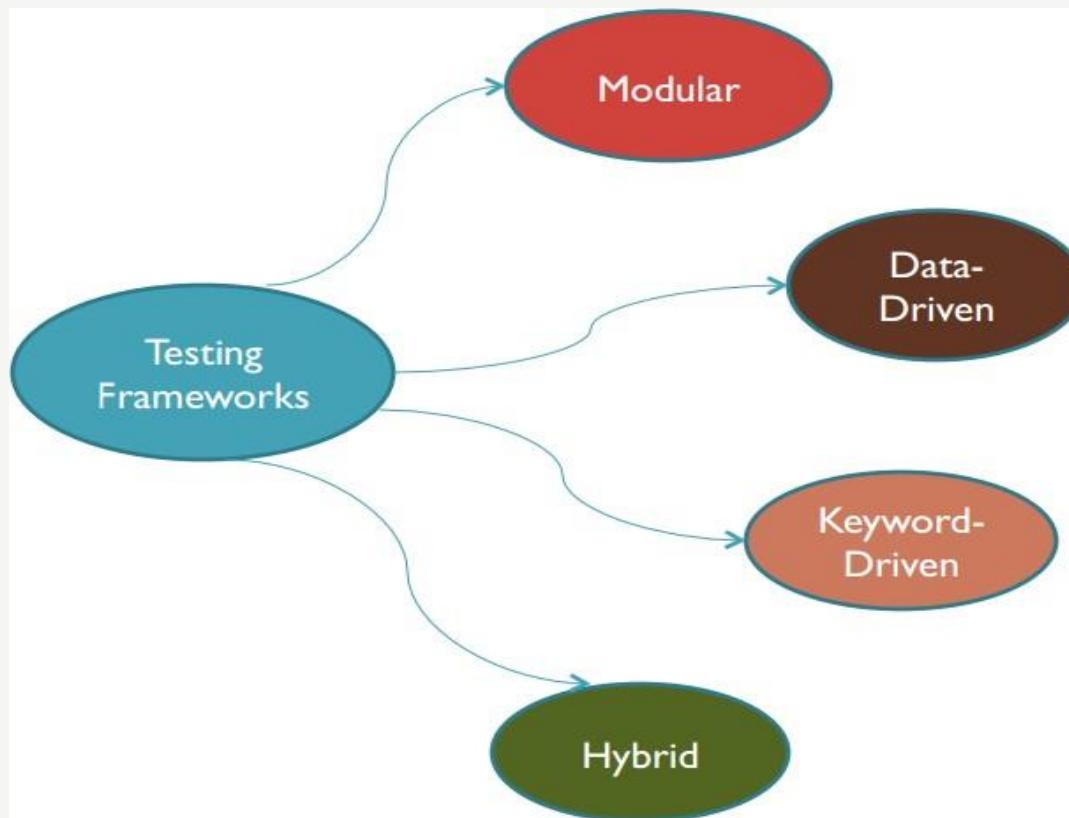
Why the testing framework is required:

- If we have a group of testers and suppose if each project implements a unique strategy then the time needed for the tester become productive in the new environment will take long.
- To handle this we cannot make changes to the automation environment for each new application that comes along.

Properties of a testing framework:

- For this purpose we use a testing framework that is application independent and has the capability to expand with the requirements of each application.
- Also an organized test framework helps in avoiding duplication of test cases automated across the application.
- In short Test frameworks helps teams organize their test suites and in turn help improve the efficiency of testing.

Types Of Testing Frameworks:



Modular Testing Framework

- The Modularity testing framework is built on the concept of abstraction.
- This involves the creation of independent scripts that represent the modules of the application under test. These modules in turn are used in a hierarchical fashion to build large test cases.
- Thus it builds an abstraction layer for a component to hide that component from the rest of the application. Thus the changes made to the other part of the application do not effect that component.

Modular Testing Framework

Example of Modular Testing Framework:

- To demonstrate the modular framework we use the calculator program.—Consider the basic functions of the calculator such as addition, subtraction, multiplication, division which are part of the Standard view.

We create scripts for these functions as follows:

Add:

Sub Main

```
Window Set Context, "Caption=Calculator", ""  
PushButton Click, "ObjectIndex=10" 'Press 5  
PushButton Click, "ObjectIndex=20" 'Press +  
PushButton Click, "ObjectIndex=14" 'Press 6  
PushButton Click, "ObjectIndex=21" 'Press =  
Result = LabelUP (CompareProperties, "Text=11.", "UP=Object Properties")  
'Compare Expected to Actual Results
```

End Sub

Modular Testing Framework

Advantages:

- —Modular division of scripts leads to easier maintenance and also the scalability of the automated test suites. —
- The functionality is available in easy to use test libraries so creating new driver scripts for different tests is easy and fast.

Disadvantages:

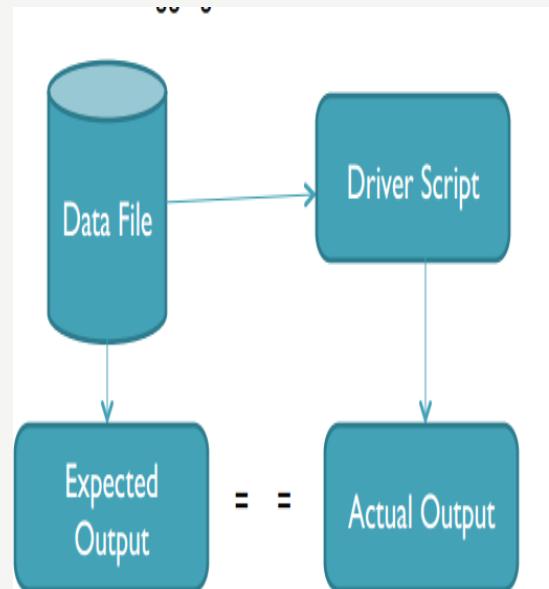
- The main problem with modular frameworks is that the test script have test data embedded in them. So when the test data needs to be updated we need to change the code of the script.
- This becomes a big problem when the test script is large. For this purpose, data- driven testing frameworks have been introduced.

Data Driven Testing

- A key benefit of automating functional testing is the ability to test large volumes of data on the system quickly.
- But you must be able to manipulate the data sets, perform calculations, and quickly create hundreds of test iterations and permutations with minimal effort.
- Test Automation Frameworks must have capability to integrate with spreadsheets and provide powerful calculation features.

Data-Driven Testing Framework

- Data driven testing is where the test input and the expected output results are stored in a separate data file (normally in a tabular format) so that a single driver script can execute all the test cases with multiple sets of data.
- The driver script contains navigation through the program, reading of the data files and logging of the test status information.



Data-Driven Testing Framework

Example of Data Driven Testing Framework

- To demonstrate the data driven testing framework we use the login page of the flight reservation.
- The first step involves creating the test data file. (testdata.csv). This data file contains the different types of input data which will be given to the driver script.

Test Case	Number1	Operator	Number2	Expected Result
Add	2	+	3	5
Subtract	3	-	2	1
Multiply	2	*	3	6
Divide	2	/	-2	-1

Data-Driven Testing Framework

In the next step we create a driver script and make references to the test data file.

```
data = open ( 'testdata.csv' ) . read ( )
lines = data . splitlines ( )
#excluding the header row
```

for line in lines:

 Read Number1

 Read Number2

 Read Operator

Calculate the result using the Operator on

Number 1 and Number2

Compare the result to the expected result. Reads the data from the data file computes the value and compares it with the expected result from the data file.

Data-Driven Testing Framework

Advantages:

- This framework reduces the number of overall test scripts needed to implement all the test cases.
- Less amount of code is required to generate all the test cases.

Disadvantages:

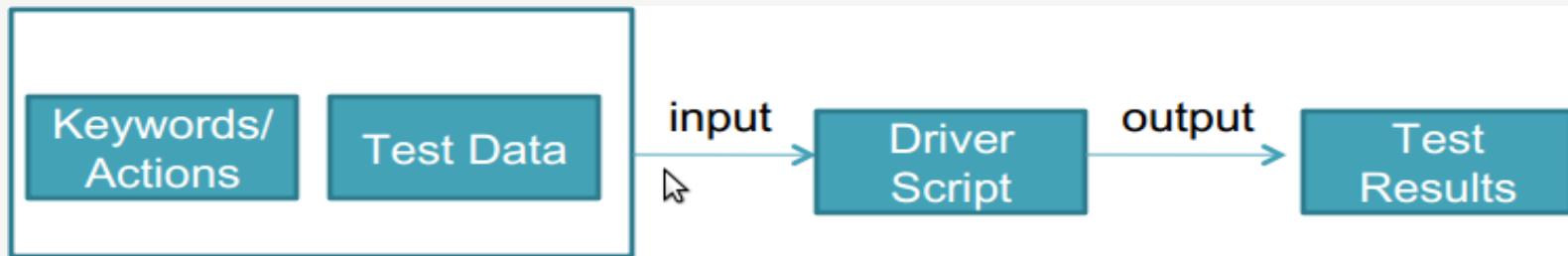
- The test cases created are similar and creating new kind of tests requires creating new driver scripts that understand different data. Thus the test data and driver scripts are strongly related that changing either requires changing the other.

•Refer this example Link-

<https://github.com/TopsCode/Automation-Testing/blob/master/Module-4/4.1%20DataDrivenExample/src/com/framework/DataDrivenDemo.java>

Keyword-Driven Testing Framework

- Keyword driven testing is an application independent framework utilizing data tables and self explanatory keywords to explain the actions to be performed on the application under test.
- Not only is the test data kept in the file but even the directives telling what to do which is in the test scripts is put in external input data file.
- These directives are called keywords. The keyword based testing is an extension to the data driven testing.



Keyword-Driven Testing Framework

Example of Keyword Driven Testing Framework:

- To demonstrate the keyword driven testing we take the actions performed by the mouse when making calculations.
- We create a table that maps the actions performed with the mouse on the window of the calculator application. In this table,
 - The windows column represents the application for which we are performing the mouse action.
 - The control column represents the control that we are clicking with the mouse.
 - The action column represents the action performed by the mouse.
 - The argument column contains the specific control value.

Keyword-Driven Testing Framework

Window	Control	Action	Arguments
Calculator	Menu		View, Standard
Calculator	Pushbutton	Click	2
Calculator	Pushbutton	Click	+
Calculator	Pushbutton	Click	3
Calculator	Pushbutton	Click	=
Calculator		Verify Result	5
Calculator		Clear	
Calculator	Pushbutton	Click	5
Calculator	Pushbutton	Click	-
Calculator	Pushbutton	Click	3
Calculator	Pushbutton	Click	=
Calculator		Verify Result	2

Keyword-Driven Testing Framework

After creating the table, we create a set of scripts for reading in the table, executing each step based on the keyword contained in the action field and logs any relevant information.

- The below pseudo code represents this test of scripts.
 - Main Script / Program
 - Connect to data tables.

Keyword-Driven Testing Framework

Read in row and parse out values.

- Pass values to appropriate functions.
- Close connection to data tables.

Advantages:

- Automation expertise is not required to maintain or create a new set of test cases.
- Keywords are reused across multiple test cases.

Disadvantages:

- The main problem is that this requires a more complicated framework than the data driven framework.
- With the keyword driven approach the test cases get longer and complex and this is due to the greater flexibility that this approach offers.

Keyword-Driven Testing Framework

•**Refer this example Link-**

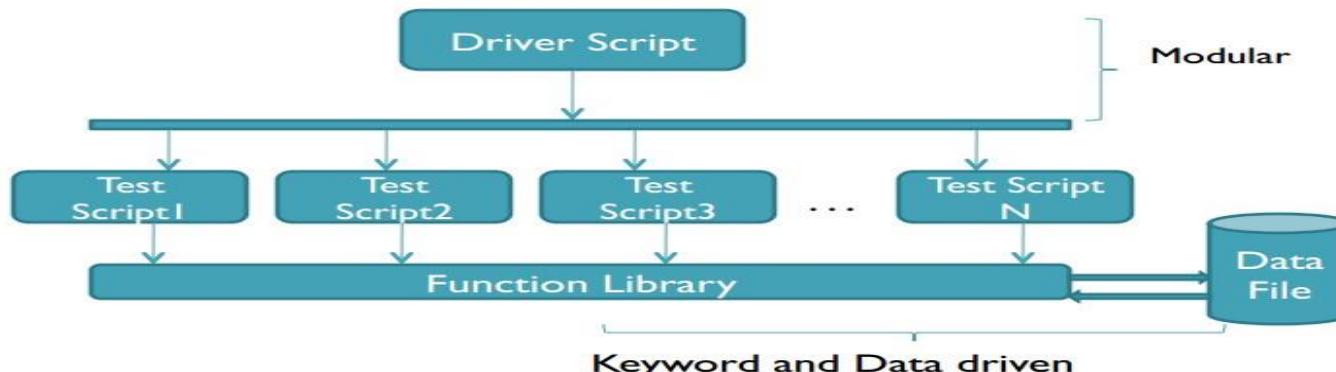
<https://github.com/TopsCode/Automation-Testing/blob/master/Module-4/4.2%20KeywordDrivenExample/test-output/index.html>

Hybrid Testing Framework

- Hybrid testing framework is the combination of modular, data-driven and keyword driven testing frameworks.
- This combination of frameworks helps the data driven scripts take advantage of the libraries which usually accompany the keyword driven testing.

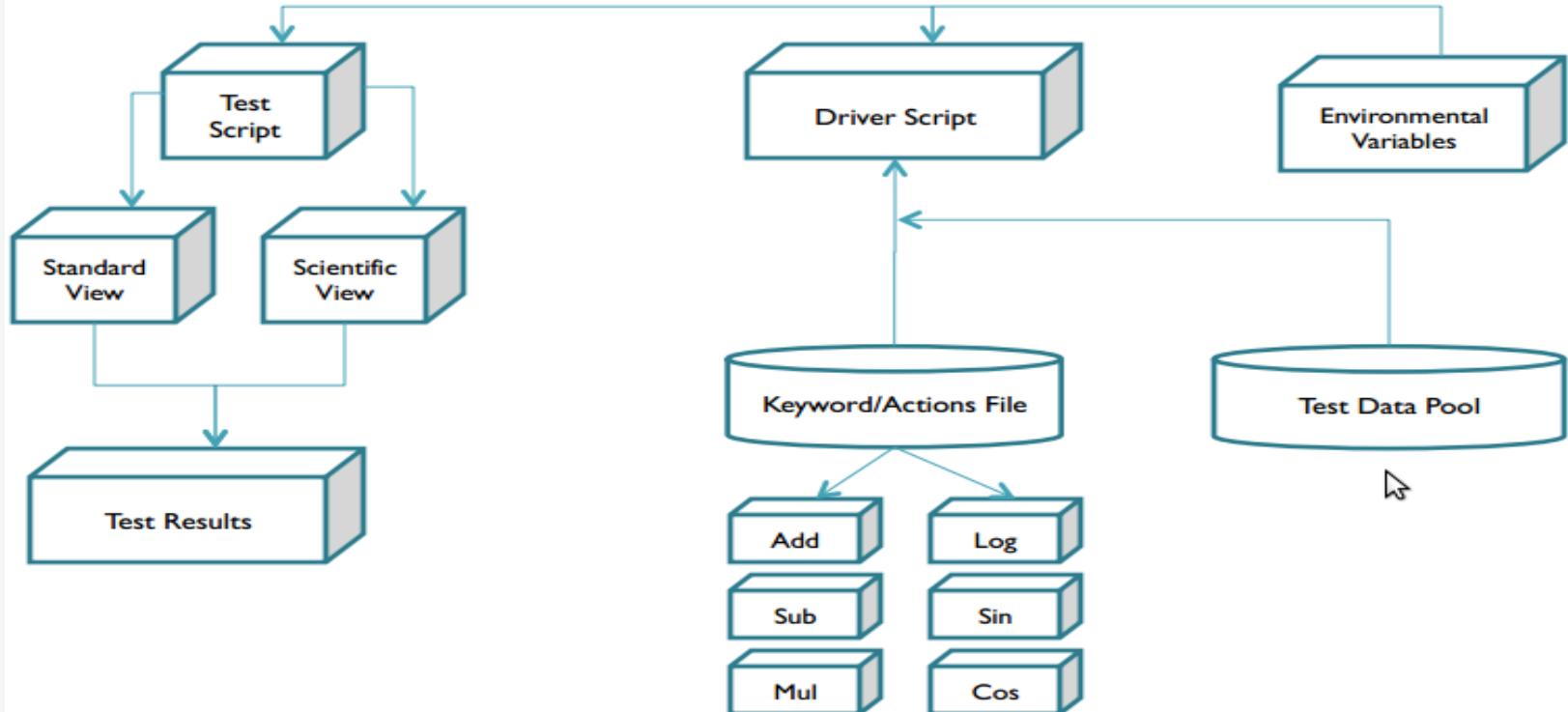
Hybrid Testing Framework

- Hybrid testing framework is the combination of modular, data-driven and keyword driven testing frameworks.
- This combination of frameworks helps the data driven scripts take advantage of the libraries which usually accompany the keyword driven testing.



Hybrid Testing Framework

The hybrid framework for the calculator can be shown as follows:



Comparison of Frameworks

Approach	Advantages	Disadvantages
Modular testing framework	Modular approach Reusable functions Hierarchical Structure	Test data within the scripts limits reusability, Test script is dependent on software.
Data driven testing framework	Improved Maintainability	Dependency on technical expertise, Test script is dependent on software.
Keyword driven testing framework	Ease of maintenance, Scalability, Less dependency of software.	Dependency on technical expertise, Requires large effort
Hybrid testing framework	Integrates the advantages of all the other frameworks.	Increased Complexity

Module – 5[Maven Project Management Tools]

- Introduction to Maven
- Installation of Maven
- Maven Structure
- Maven Dependency
- Maven Repository
- Maven Eclipse Integration

Maven Project Management Tool

Maven Introduction

- Maven is an automation and management tool developed by Apache Software Foundation.
- It was initially released on 13 July 2004.
- In Yiddish language the meaning of Maven is "accumulator of knowledge".
- It is written in Java Language and used to build and manage projects written in C#, Ruby, Scala, and other languages.
- It allows the developer to create projects using Project Object Model and plugins.
- It helps to build projects, dependency, and documentation.
- Its development process is very similar to ANT. However, it is much advanced than ANT.
- Maven is also able to build any number of projects into desired output such as jar, war, metadata.

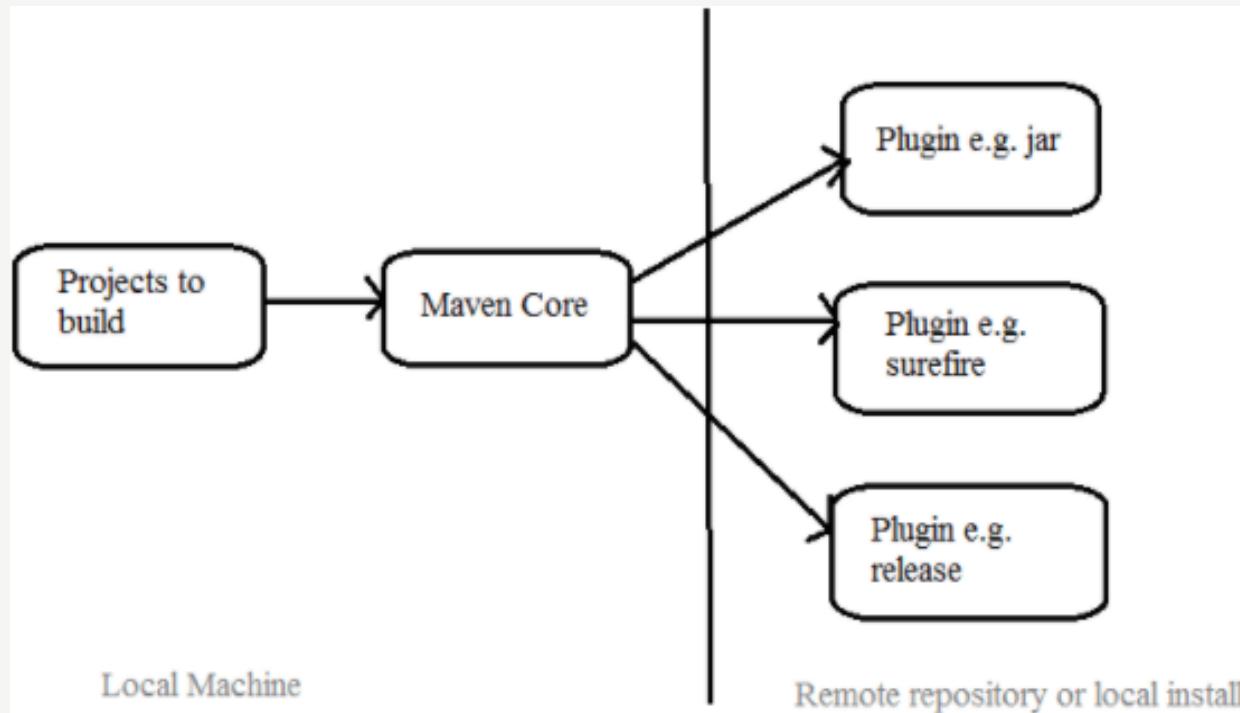
How can Maven benefit my development process?

- Maven helps the developer to create a java-based project more easily.
- Accessibility of new feature created or added in Maven can be easily added to a project in Maven configuration.
- It increases the performance of project and building process.
- The main feature of Maven is that it can download the project dependency libraries automatically.
- Below are the examples of some popular IDEs supporting development with Maven:
 - Eclipse
 - IntelliJ IDEA
 - JBuilder
 - NetBeans
 - MyEclipse

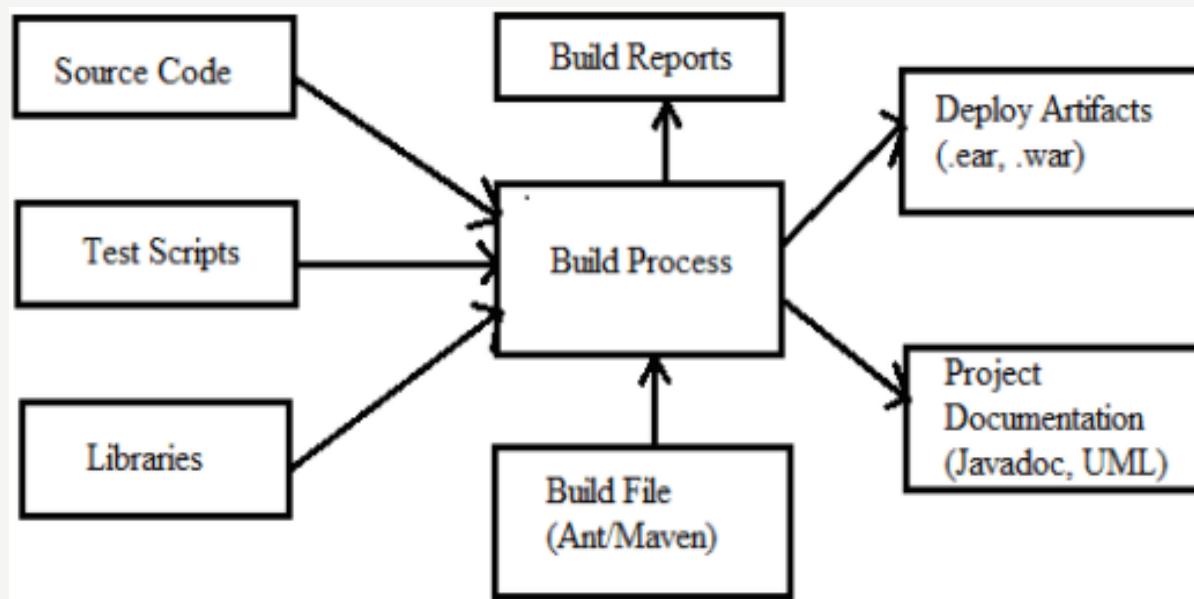
Processes which can manage using maven

- Builds
- Documentation
- Reporting
- Dependencies
- SCMs
- Releases
- Distribution
- mailing list

Maven Architecture



Maven Architecture



ANT v/s Maven

Ant	Maven
Ant doesn't have formal conventions , so we need to provide information of the project structure in build.xml file.	Maven has a convention to place source code, compiled code etc. So we don't need to provide information about the project structure in pom.xml file.
Ant is procedural , you need to provide information about what to do and when to do through code. You need to provide order.	Maven is declarative , everything you define in the pom.xml file.
It is less preferred than Maven.	It is more preferred than Ant.

ANT v/s Maven

Ant	Maven
There is no life cycle in Ant.	There is life cycle in Maven.
It is a tool box .	It is a framework .
It is mainly a build tool .	It is mainly a project management tool .
The ant scripts are not reusable .	The maven plugins are reusable .

Installation of Maven

You can download and install maven on windows, linux and MAC OS platforms.

Here, we are going to learn how to install maven on windows OS.

To install maven on windows, you need to perform following steps:

1. Download maven and extract it
2. Add JAVA_HOME and MAVEN_HOME in environment variable
3. Add maven path in environment variable
4. Verify Maven

Maven - POM

- POM stands for Project Object Model.
- It is fundamental unit of work in Maven.
- It is an XML file that resides in the base directory of the project as pom.xml.
- The POM contains information about the project and various configuration detail used by Maven to build the project(s).
- POM also contains the goals and plugins.
- While executing a task or goal, Maven looks for the POM in the current directory. It reads the POM, gets the needed configuration information, and then executes the goal.

Maven - POM

- Some of the configuration that can be specified in the POM are following –
 - project dependencies
 - plugins
 - goals
 - build profiles
 - project version
 - developers
 - mailing list
- Before creating a POM, we should first decide the project **group** (groupId), its **name** (artifactId) and its version as these attributes help in uniquely identifying the project in repository.

Maven – POM Example

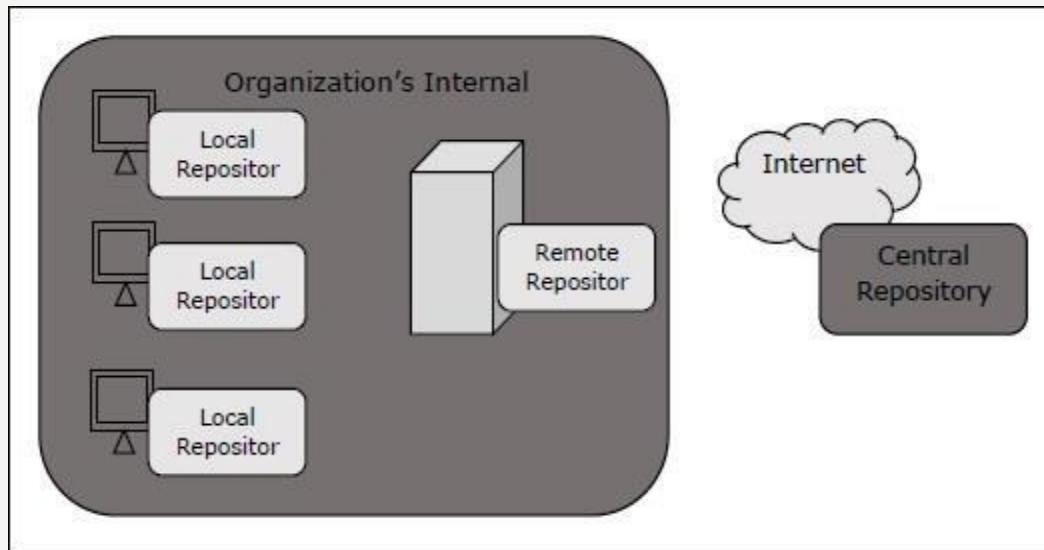
Sr.No.	Node & Description
1	Project root This is project root tag. You need to specify the basic schema settings such as apache schema and w3.org specification.
2	Model version Model version should be 4.0.0.
3	groupId This is an Id of project's group. This is generally unique amongst an organization or a project. For example, a banking group com.company.bank has all bank related projects.

Maven – POM Example

Sr.No.	Node & Description
4	artifactId This is an Id of the project. This is generally name of the project. For example, consumer-banking. Along with the groupId, the artifactId defines the artifact's location within the repository.
5	version This is the version of the project. Along with the groupId, It is used within an artifact's repository to separate versions from each other. For example – com.company.bank:consumer-banking:1.0 com.company.bank:consumer-banking:1.1.

Maven Repositories

- Repository is a directory where all the project jars, library jar, plugins or any other project specific artifacts are stored and can be used by Maven easily.
- Maven repository are of three types.-
 - local
 - central
 - remote



How to use Maven

- To configure the Maven, you need to use Project Object Model, which is stored in a pom.xml-file.
- POM includes all the configuration setting related to Maven.
- Plugins can be configured and edit in the <plugins> tag of a pom.xml file. And developer can use any plugin without much detail of each plugin.
- When user start working on Maven, it provides default setting of configuration, so the user does not need to add every configuration in pom.xml

Maven Dependencies

```
<!-- https://mvnrepository.com/artifact/javax.servlet/jsp-api -->
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>jsp-api</artifactId>
    <version>2.0</version>
    <scope>provided</scope>
</dependency>
</dependencies>
```

Maven Dependencies

```
<build>
    <finalName>MavenWebApp</finalName>
    <plugins>
        <plugin>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.5.1</version>
            <configuration>
                <source>1.8</source>
                <target>1.8</target>
            </configuration>
        </plugin>
    </plugins>
</build>
</project>
```

Steps/process involved in building the project

- Add / Write the code for application creation and process that into source code repository
- Edit configuration / pom.XML / plugin details
- Build the application
- Save the build process output as WAR or EAR file to a local location or server
- Get the file from local location or server and deploy the file to the production site or client site Updated the application document with date and updated version number of the application
- create and generate a report as per the application or requirement.

Refer this example Link-

<https://github.com/TopsCode/Automation-Testing/blob/master/Module-5/5.1%20MavenExample/src/test/java/com/mavendemo/MavenClassDemo.java>

Thank You

Appium Automation Testing

Modules

- [Appium Introduction](#)
- [Appium Set Up On Window](#)
- [Project Set Up](#)
- [Native App Automation](#)

Module - 1 [Appium Introduction]

Agenda

- Appium Introduction
- Appium Architecture and how works
- Types of Mobile App

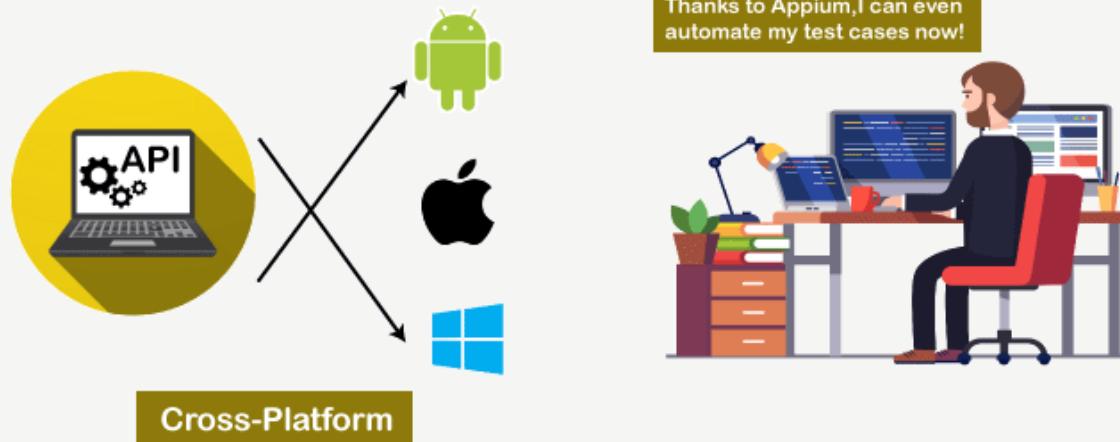
Appium Introduction

What is Appium?

- **Appium** is an open source automation tool for running scripts and testing native applications, mobile-web applications and hybrid applications on Android or iOS using a webdriver.
- It is a cross-platform mobile automation tool, which means that it allows the same test to be run on multiple platforms.
- Multiple devices can be easily tested by Appium in parallel.
- In today's development area, the demand for mobile applications is high.
- Currently, people are converting their websites into mobile apps.
- Therefore, it is very important to know about mobile software automation testing technology and also stay connected with new technology.
- **Appium** is a mobile application testing tool that is currently trending in **Mobile Automation Testing Technology**.
- Appium is used for automated testing of **native**, **hybrid**, and **web** applications.

What is Appium?

- It supports automation test on the simulators (iOS) and emulators (Android) as well as physical devices (Android and iOS both).
- Previously, this tool mainly focused on IOS and Android applications that were limited to mobile application testing only.
- Few updates back, Appium declared that it would now support desktop application testing for windows as well.



What is Appium?

- Appium is very much similar to the **Selenium Webdriver** testing tool. So, if you already know Selenium Webdriver, Appium becomes very easy to learn.
- Appium has **NO dependency** on mobile device OS because it has a framework that converts the Selenium Webdriver commands to UIAutomator and UIAutomation commands for Android and iOS respectively, that depends on the device type rather than the OS type.
- It supports several languages such as Java, PHP, Objective C, C#, Python, JavaScript with node.js, and Ruby, and many more that have Selenium client libraries.
- Selenium is the backend of Appium that provides control over the functionality of Selenium for testing needs.

Features of Appium

- Appium does not require application source code or library.
- Appium provides a strong and active community.
- Appium has multi-platform support i.e., it can run the same test cases on multiple platforms.
- Appium allows the parallel execution of test scripts.
- In Appium, a small change does not require re-installation of the application.
- Appium supports various languages like C#, Python, Java, Ruby, PHP, JavaScript with node.js, and many others that have Selenium client library.

Advantage of Appium

- Appium is an open-source tool, which means it is freely available. It is easy to install.
- It allows the automated testing of hybrid, native, and web applications.
- Unlike other testing tools, you do not need to include any additional agents in your app to make Appium compatible with automation.
- It tests the same app, which is going to upload in App Store.
- An additional feature added to Appium. Now it would support desktop application testing for windows as well along with mobile application testing.
- Appium is a cross-platform, freely available mobile testing tool, which allows us the cross-platform mobile testing.
- This means you can test on multiple platforms (single API for both Android and IOS platforms).

Disadvantage of Appium

- Along with some features and advantages, Appium has some drawbacks too, which are as follows-Lack of detailed reports.
- Since the tests depend on the remote web driver, so it is a bit slow.
- It is not a limitation, but an overhead that Appium uses UIAutomator for Android that only supports Android SDK, API 16, or higher. However, Appium supports older APIs, but not directly. It uses another open-source library Selendroid to support older APIs.
- In iOS, only one instance (iOS Script) can run on one Mac OS device, which means one test can be executed at a time per Mac.
- If you want to run your tests on multiple iOS devices at the same time, you need to arrange the same number of Mac machines.
- But it would be expensive to arrange various Mac machines.
- Solution: This problem can be resolved if you will run your script in the mobile cloud of Sauce Lab.
- Currently, it allows scripts to run on multiple iOS simulators at the same time.

Appium Architecture

- Appium is an HTTP server that is written in node.js. It starts a "test case" on the device that gives rise to a server and listens for proxied commands from the main Appium server.
- Tester writes the Test scripts to execute on device or Emulator. Several webdriver sessions for different platforms like Android and IOS are created and handled by the Appium.
- Test Scripts written by the tester executes on the Emulator or device by sending them as requests to the Appium server.
- Each vendor, such as IOS or Android, has a different method and mechanism to execute test cases on the device.
- So, the test case executes after listening commands from the Appium server. Appium uses JSON wire protocols to send command requests to Appium server.

Appium Architecture

Appium Architecture

Selenium

Selenium act as a client and send command (http request) to Appium server via JSON wire Protocol.
 JSON Wire Protocol
 Contains all the capabilities and configuration set in code

Appium

Appium Server then creates a automation session for the client and also checks the desired capabilities of client and then Appium server sends the server request to UIAutomator (Android) OR UIAutomation (IOS)

UIAutomator

This frameworks will then communicate with bootstrap.jar which is running in Emulator/Real device for performing client operations

Selenium (Client lib and all the jars)

JSON Wire Protocol

Appium (Server) (Written in Node.Js)

Appium Server Request

Appium Server Request

UIAutomator (Android)

UIAutomation (IOS)

Android Mobile

UIAutomator request to bootstrap.jar

UIAutomation request to bootstrap.js

IOS Mobile

Bootstrap.jar works as TCP server. And then it executes the command on the devices.

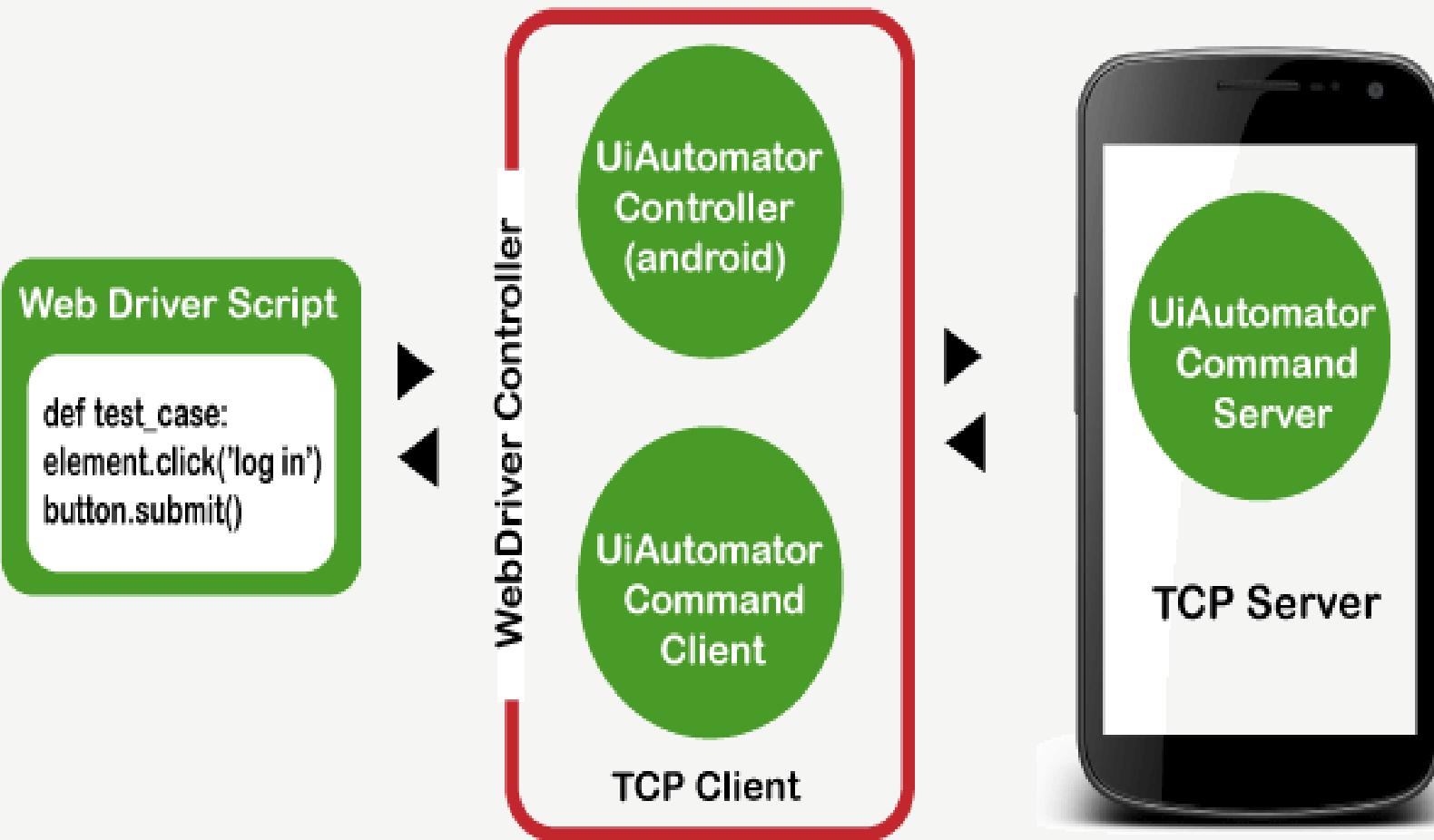
How Appium Works?

- When we install the Appium, a server is also installed with it on our machine that exposes the REST API.
- It receives command and connection requests from the client and executes that command on devices like iOS or Android.
- It replies with the HTTP responses.
- To execute requests, it uses a mobile test automation framework to run the user interface of the app. For Example –
- Apple instruments used for iOS
- Selendroid used for Android API 15 or less
- UIAutomator used for Android API 16 or higher

Appium in Android

- On Android, Appium proxies the command to a **UIAutomator** script running on the device.
- UIAutomator is a native UI automation framework of Android that allows you to run **Junit** test cases directly into the device using command line.
- Although it uses Java programming language, but Appium allows to run it from any WebDriver supported language.
- Android uses **bootstrap.jar**, which works as a TCP server. It is used to send the test commands to perform the actions on Android device using UIAutomator.
- In the below figure, see the Appium architecture in respect to Android automation -

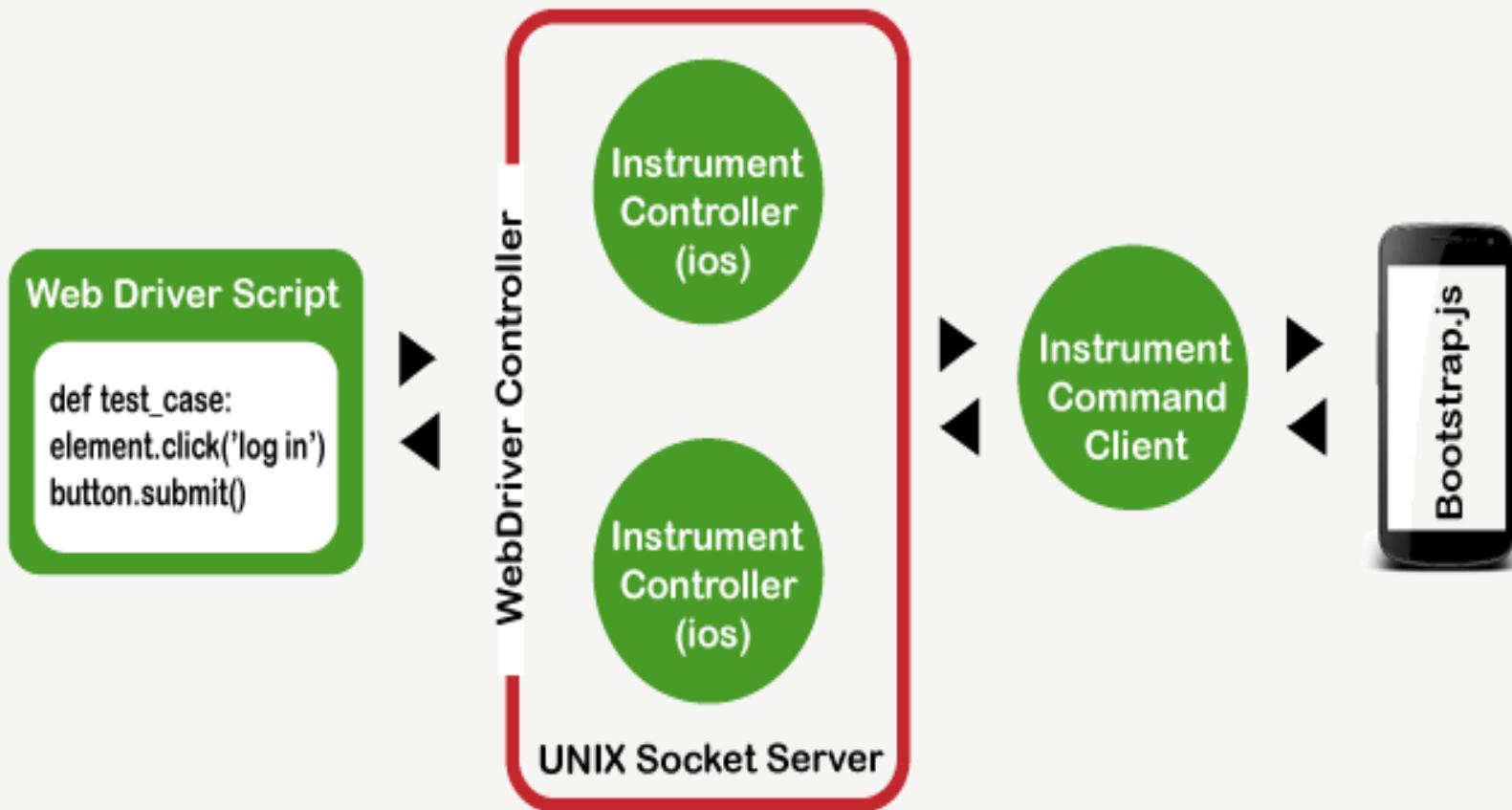
Appium in Android



Appium in IOS

- As Android uses UIAutomator, iOS uses UIAutomation. Similar to the Android, Appium proxies the command to a **UIAutomation** test case running on the Mac instruments environment.
- Apple provides this application "**instrument**" that performs various activities like building, profiling, and controlling iOS apps. On the other hand, it also has an automation component where you can write commands in JavaScript.
- It uses UIAutomation API to interact with Application UI. Appium uses same libraries to automate iOS Apps.
- In the below figure, see the Appium architecture in respect to iOS automation -

Appium in IOS



Types of Mobile App

- As we discussed earlier, Appium has the ability to deal with all sorts of applications, i.e., native, hybrid, and web.
- Let's understand them in details -
 - 1) Native
 - 2) Web
 - 3) Hybrid

Native Application

- Native applications are software programs that are developed by keeping a certain platform in mind.
- These applications are developed using a specific software development kit.
- Native apps are developed for use on a specific device and can be installed from the App Store, such as Google Play Store or Apple's App Store.
- They can work offline and can also use the device notification system.
- Some native application examples are - Pinterest, Skype, Snapchat, etc.

Web Application

- Web applications are not real applications, they are websites that run on browsers.
- These applications are developed using HTML, CSS, and JavaScript at a very low price.
- Unlike Android and iOS apps, they do not require a Software Development Kit (SDK) for developers to work with. Web applications are not developed for a particular platform.
- Since the web applications run on web browsers, they don't require any installation.
- Some web application examples are - Flipkart, Ali Express, twitter, etc.

Hybrid Application

- Hybrid application is a combination of native and web applications.
- Like native applications, these applications can be downloaded from the App Store and also can take advantage of device features, but actually they are web applications inside.
- They are developed using web development languages - HTML, CSS, and JavaScript like the hybrid app, which allows them to run on any platform.
- Some hybrid application examples are - OLA, Instagram, Basecamp, etc.

Module - 2 [Appium Setup on Windows]

Java JDK Setup

- 1) Install the Java Development Kit Software from:

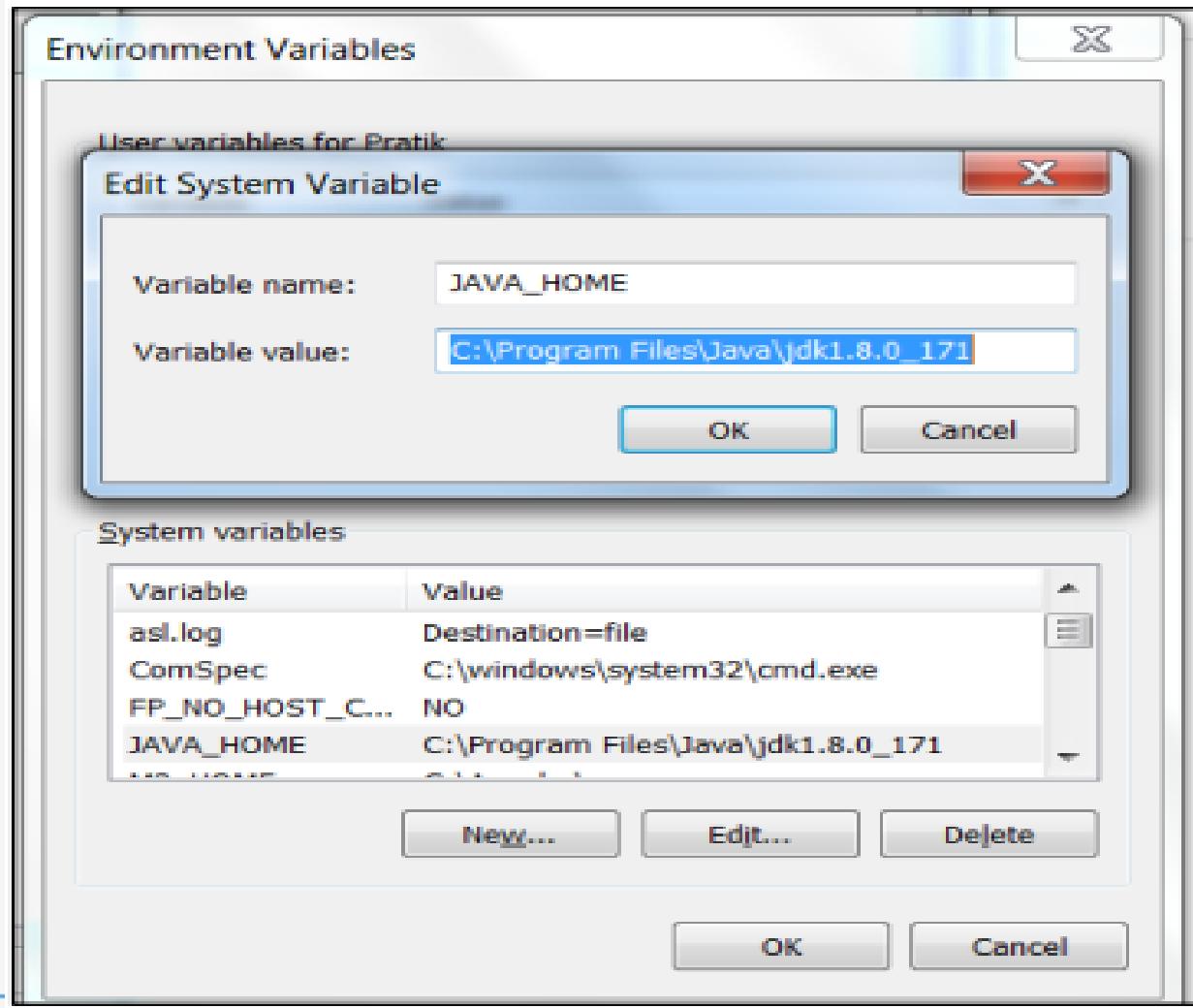
<https://www.oracle.com/in/java/technologies/downloads/>

- 2) Select the appropriate JDK software and click Download
- 3) Set JAVA_HOME:

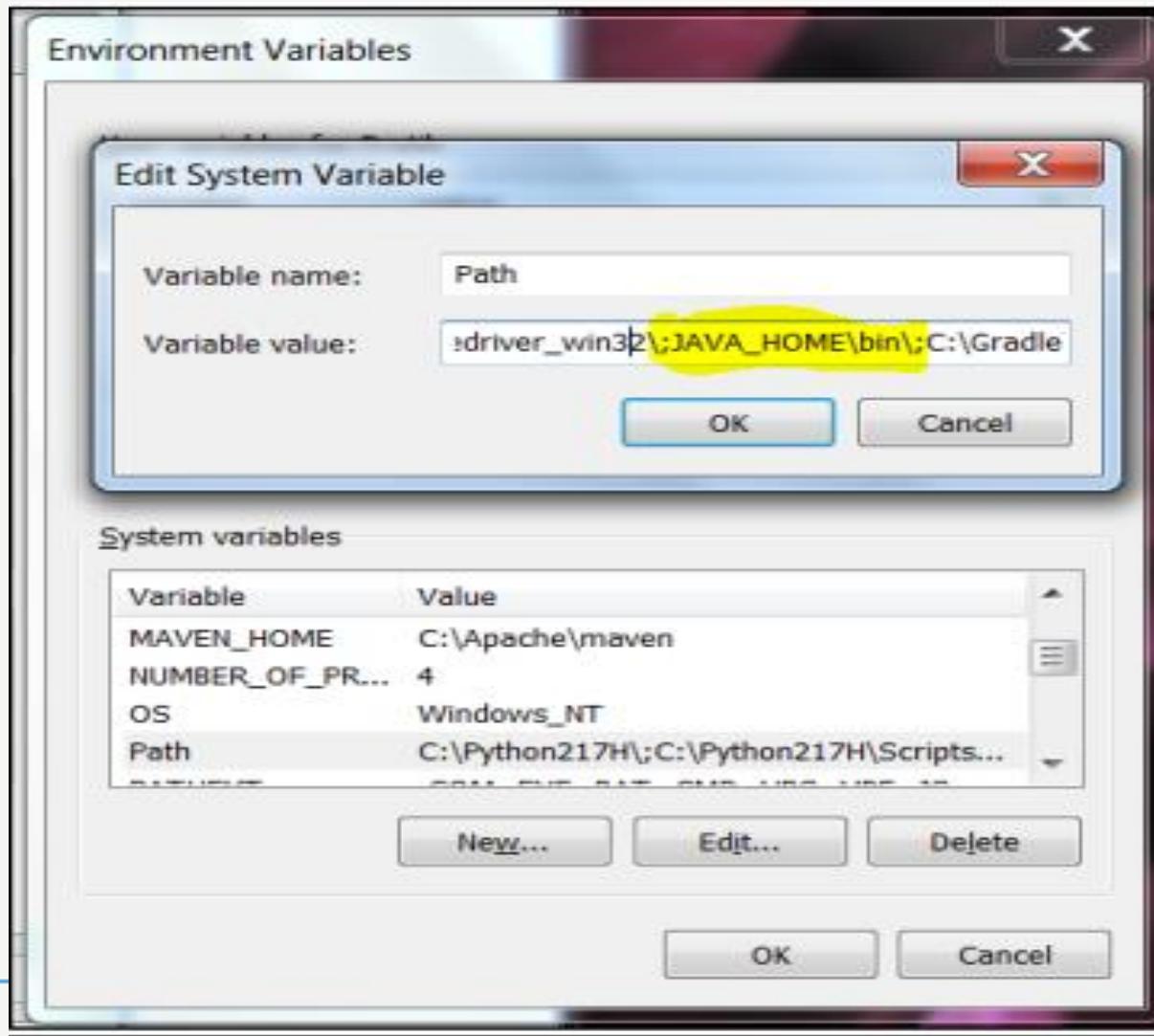
- Right click My Computer and select Properties.
- On the Advanced tab, select Environment Variables, and then edit JAVA_HOME to point to where the JDK software is located, like:

C:\Program Files\Java\jdk1.8.0_11u.

Java JDK Setup



Java JDK Setup

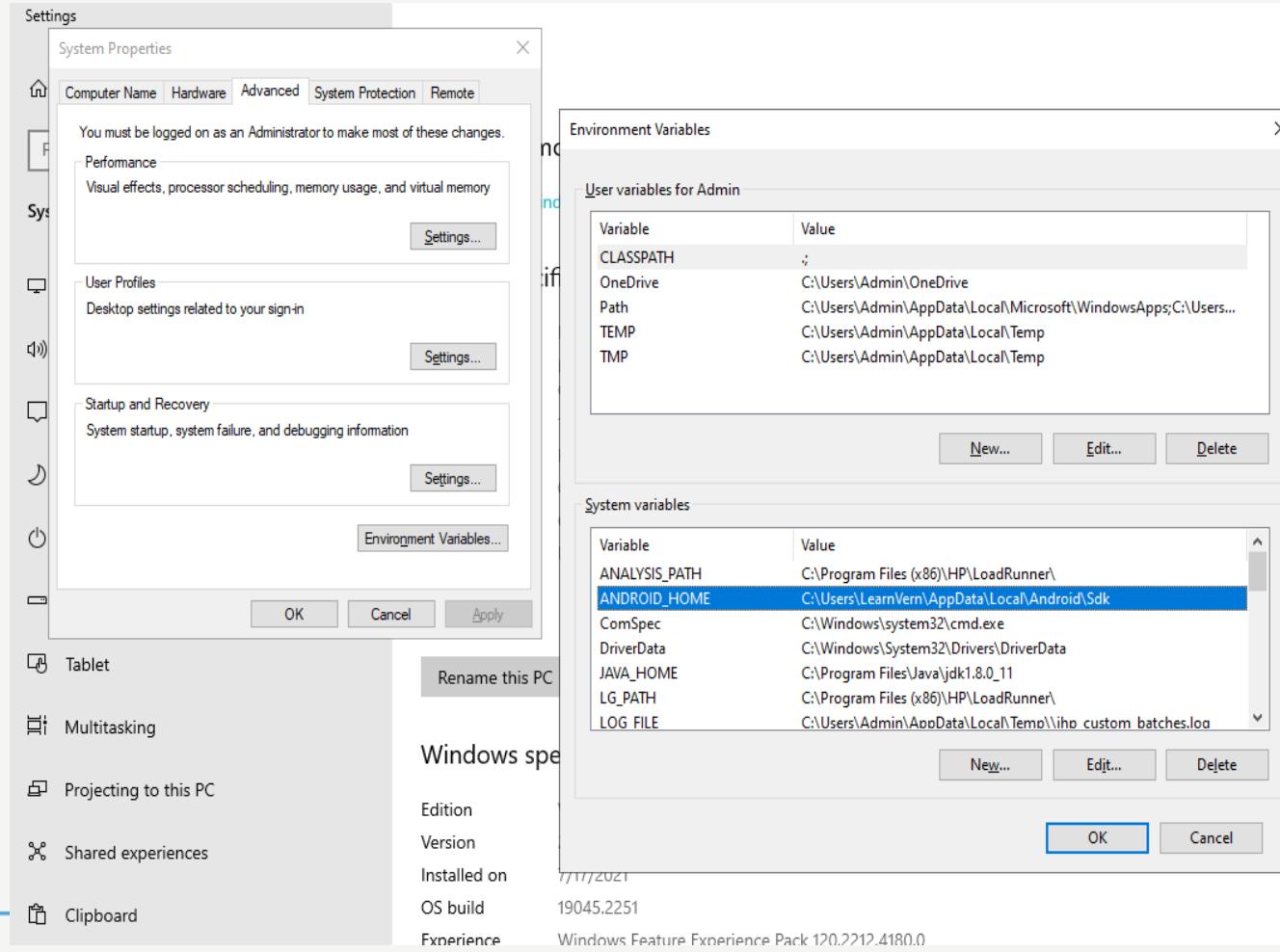


Android Studio Setup

- 1). Install Android Studio and the SDK:
- Go to: <https://developer.android.com/studio/index.html>
- Download and Install Android Studio
- Open Android Studio and then download the needed Android SDK files from Tools > Android > SDK Manager

Android Studio Setup

- 2). Set ANDROID_HOME :



This page has a few new settings
Some settings from Control Panel
have moved here, and you can copy
your PC info so it's easier to share.

Related settings

[BitLocker settings](#)

[Device Manager](#)

[Remote desktop](#)

[System protection](#)

[Advanced system settings](#)

[Rename this PC \(advanced\)](#)

Help from the web

[Finding out how many cores my processor has](#)

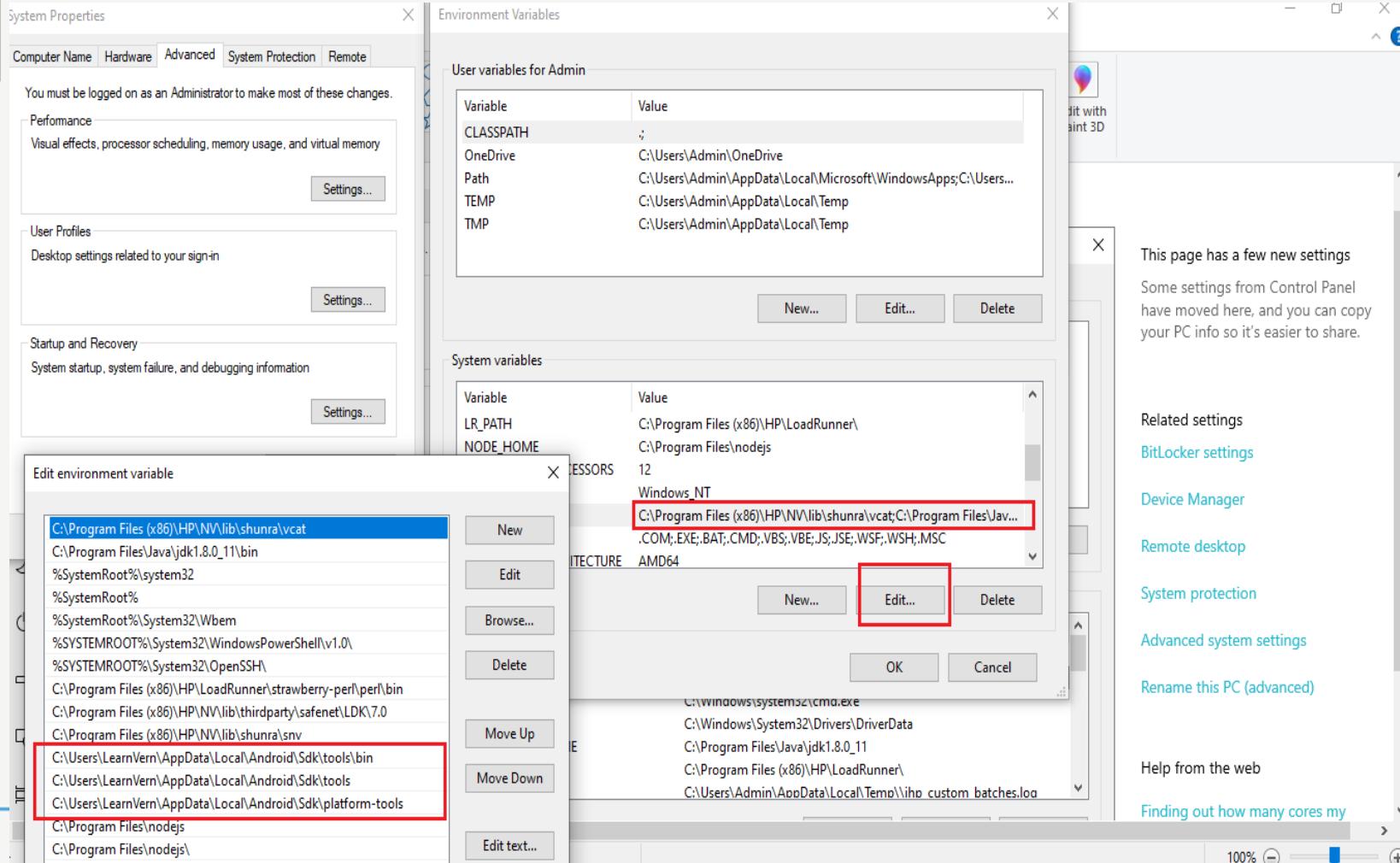
[Checking multiple Languages support](#)

 [Get help](#)

 [Give feedback](#)

Android Studio Setup

3). Other Path Set In system variable (path as a edit mode)



This page has a few new settings

Some settings from Control Panel have moved here, and you can copy your PC info so it's easier to share.

Related settings

[BitLocker settings](#)

[Device Manager](#)

[Remote desktop](#)

[System protection](#)

[Advanced system settings](#)

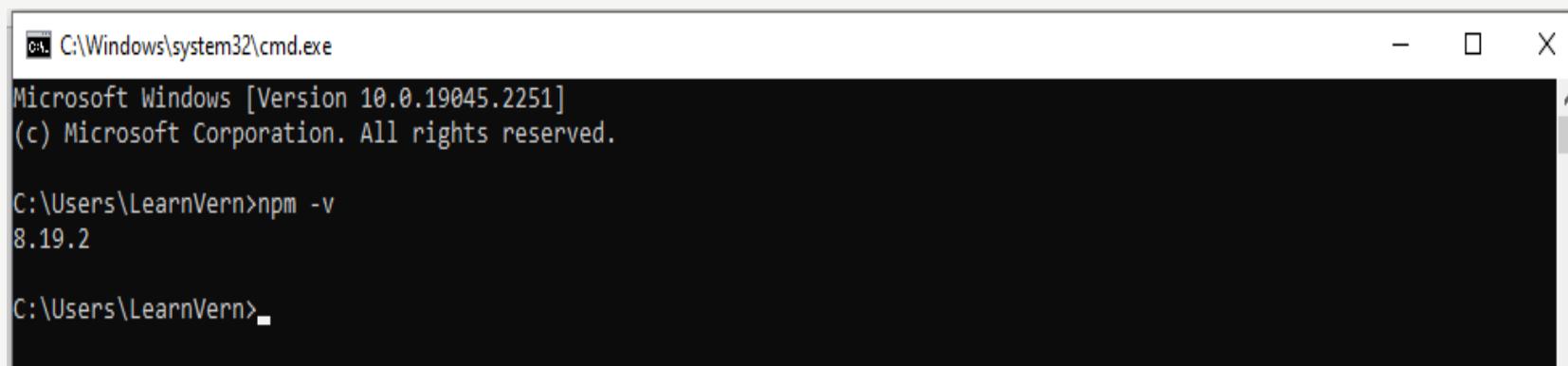
[Rename this PC \(advanced\)](#)

[Help from the web](#)

[Finding out how many cores my](#)

Installation of Node JS Setup

- 1). Install Node.js from: <https://nodejs.org/en/download/>
- 2). You can verify installation by entering \$ npm -v command at the command prompt and it will display the version.



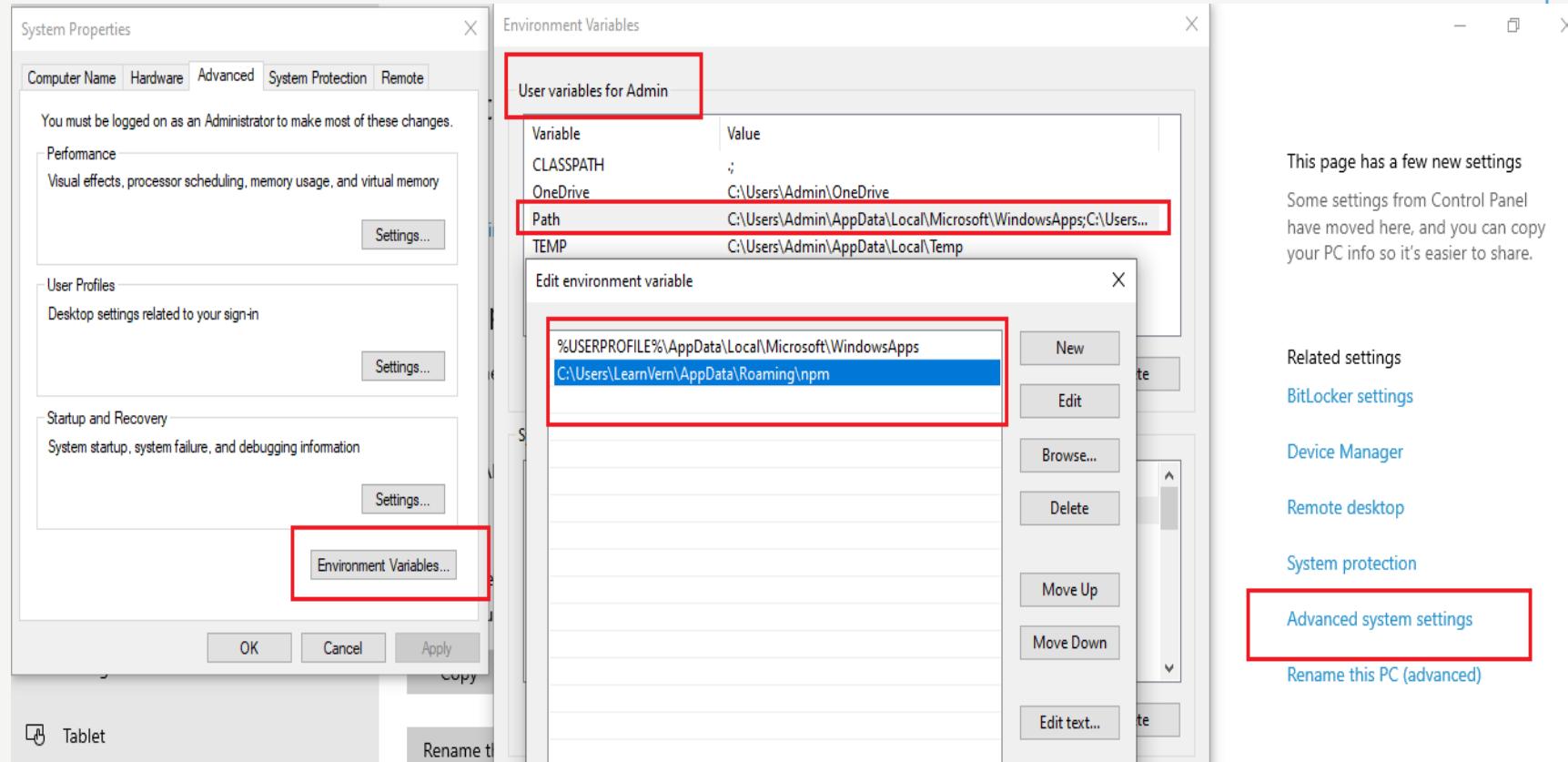
```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19045.2251]
(c) Microsoft Corporation. All rights reserved.

C:\Users\LearnVern>npm -v
8.19.2

C:\Users\LearnVern>
```

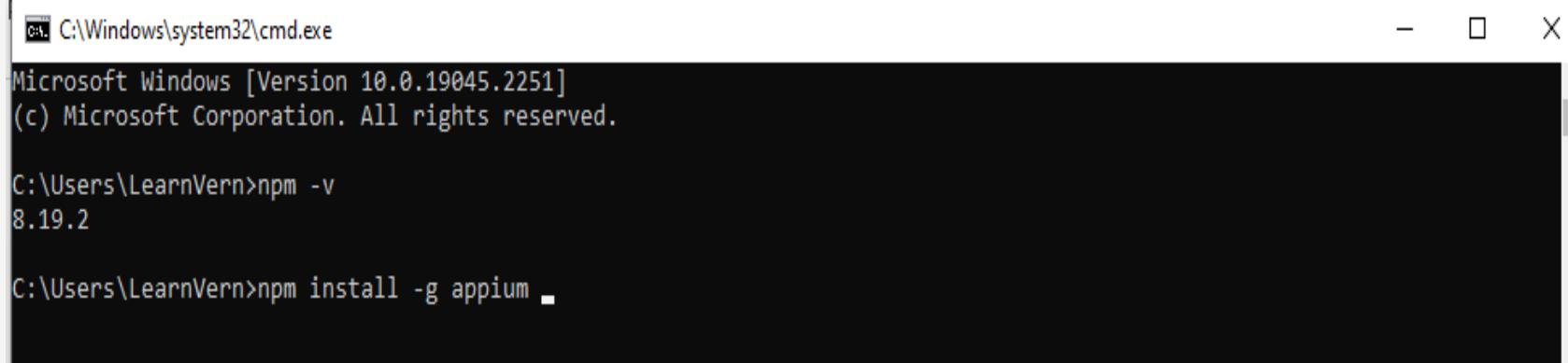
Installation of Node JS Setup

- 3). to setup path of node with user variable path



Installation of Node JS Setup

- 5). install appium with the help of node npm

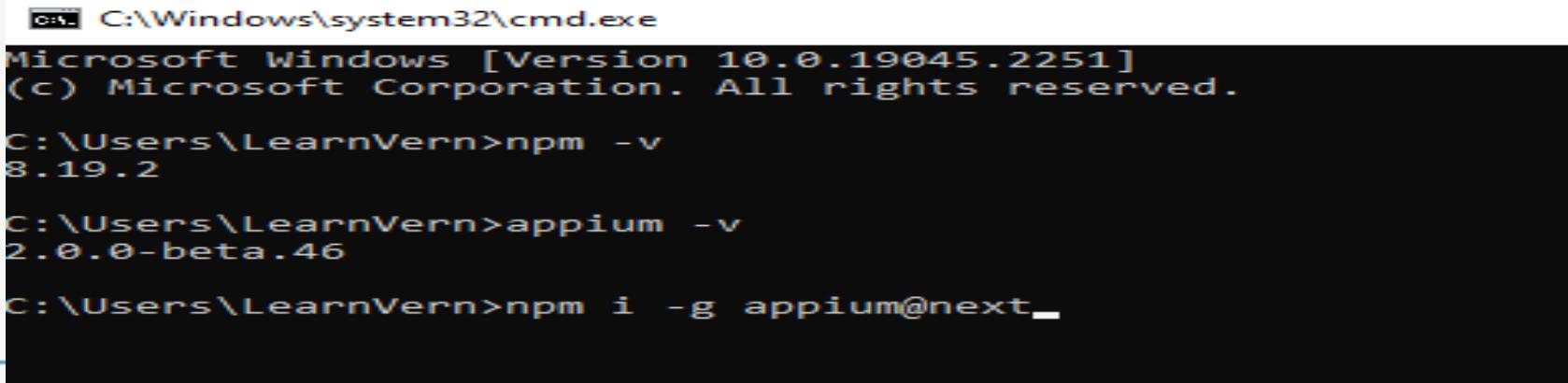


```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19045.2251]
(c) Microsoft Corporation. All rights reserved.

C:\Users\LearnVern>npm -v
8.19.2

C:\Users\LearnVern>npm install -g appium
```

- 6). install Appium Latest Update



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19045.2251]
(c) Microsoft Corporation. All rights reserved.

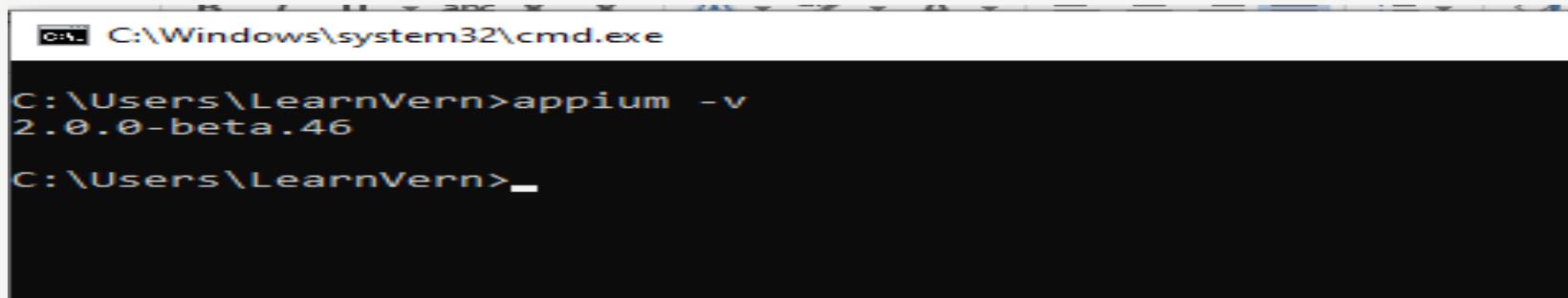
C:\Users\LearnVern>npm -v
8.19.2

C:\Users\LearnVern>appium -v
2.0.0-beta.46

C:\Users\LearnVern>npm i -g appium@next
```

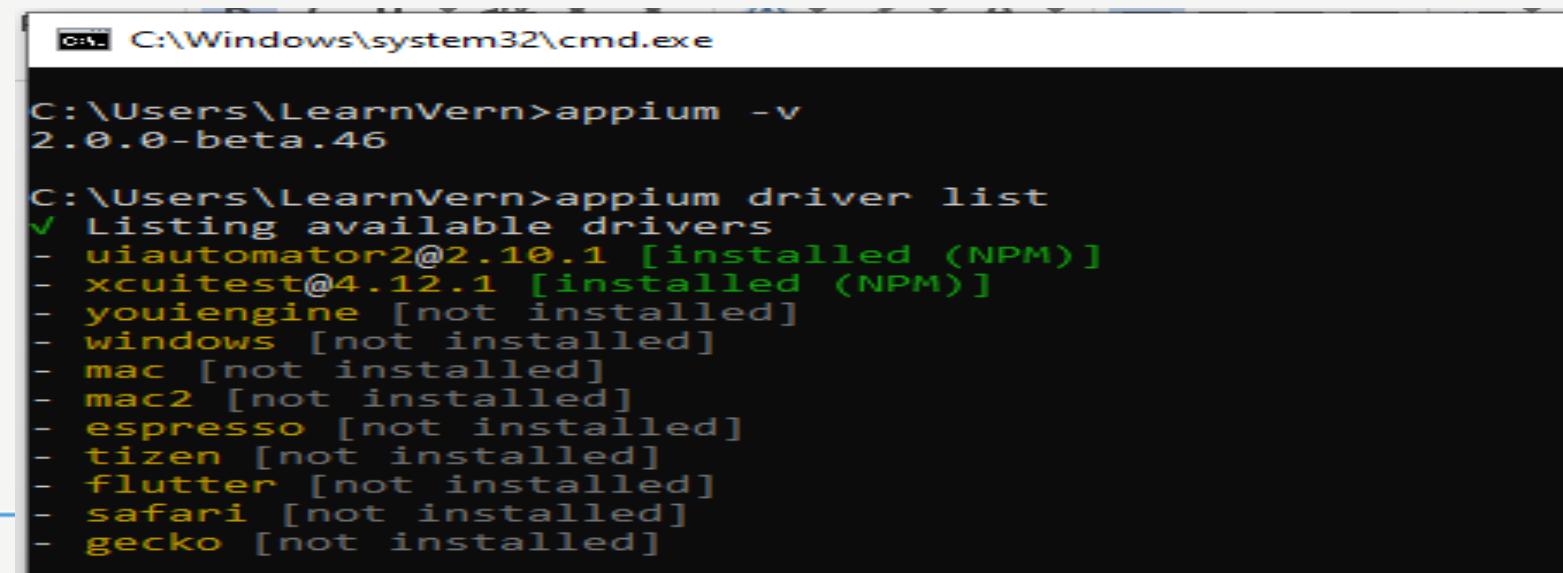
Installation of Node JS Setup

- 7). check appium latest updated version



```
C:\Windows\system32\cmd.exe
C:\Users\LearnVern>appium -v
2.0.0-beta.46
C:\Users\LearnVern>
```

- 8). appium driver list and then check must be install UIAutomator2 and xcuitest



```
C:\Windows\system32\cmd.exe
C:\Users\LearnVern>appium -v
2.0.0-beta.46
C:\Users\LearnVern>appium driver list
✓ Listing available drivers
- uiautomator2@2.10.1 [installed (NPM)]
- xcuitest@4.12.1 [installed (NPM)]
- youiengine [not installed]
- windows [not installed]
- mac [not installed]
- mac2 [not installed]
- espresso [not installed]
- tizen [not installed]
- flutter [not installed]
- safari [not installed]
- gecko [not installed]
```

Installation of Node JS Setup

- If both driver is not installed in our system so used this command to install both driver

```
C:\Windows\system32\cmd.exe

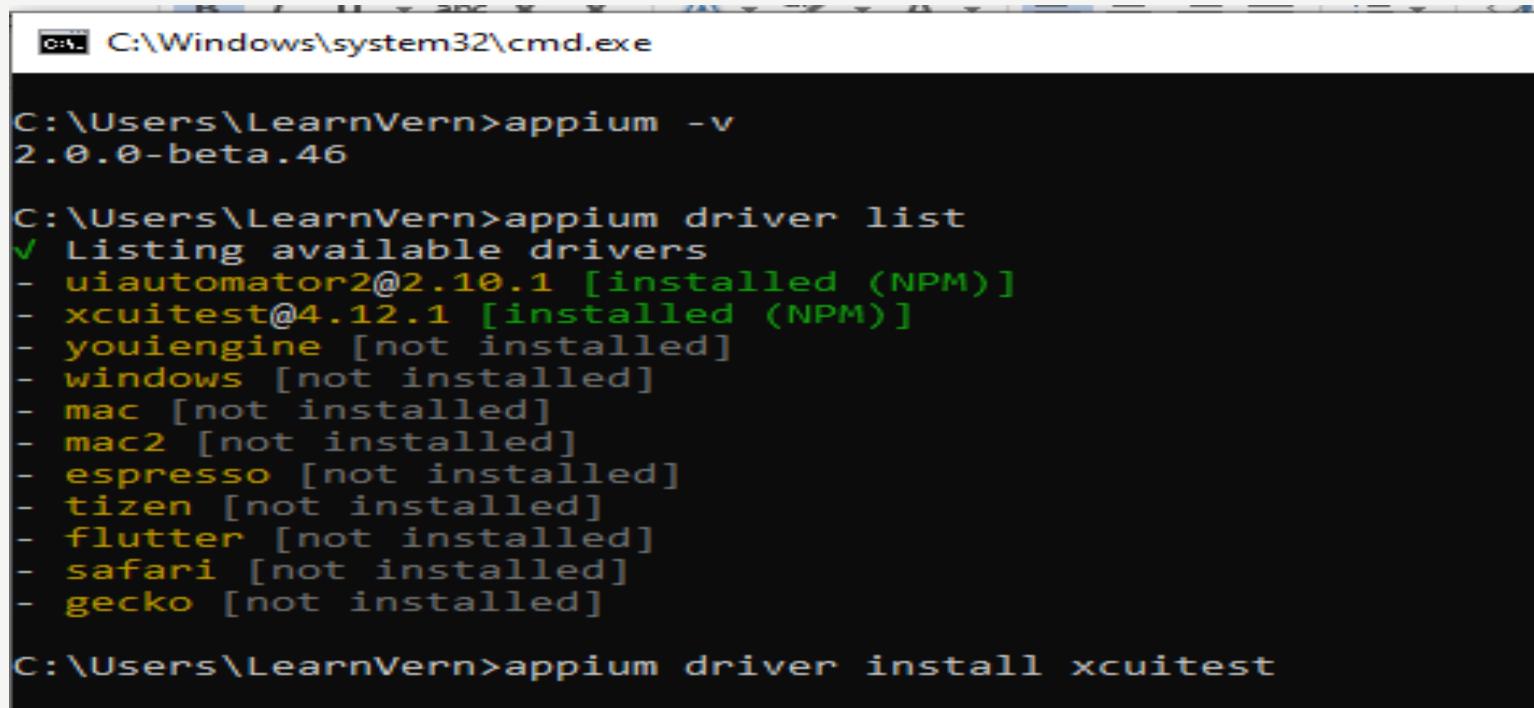
C:\Users\LearnVern>appium -v
2.0.0-beta.46

C:\Users\LearnVern>appium driver list
✓ Listing available drivers
- uiautomator2@2.10.1 [installed (NPM)]
- xcuitest@4.12.1 [installed (NPM)]
- youiengine [not installed]
- windows [not installed]
- mac [not installed]
- mac2 [not installed]
- espresso [not installed]
- tizen [not installed]
- flutter [not installed]
- safari [not installed]
- gecko [not installed]

C:\Users\LearnVern>appium driver install uiautomator2
```

- Above command to install uiautomator2 driver

Installation of Node JS Setup



```
C:\Windows\system32\cmd.exe
C:\Users\LearnVern>appium -v
2.0.0-beta.46

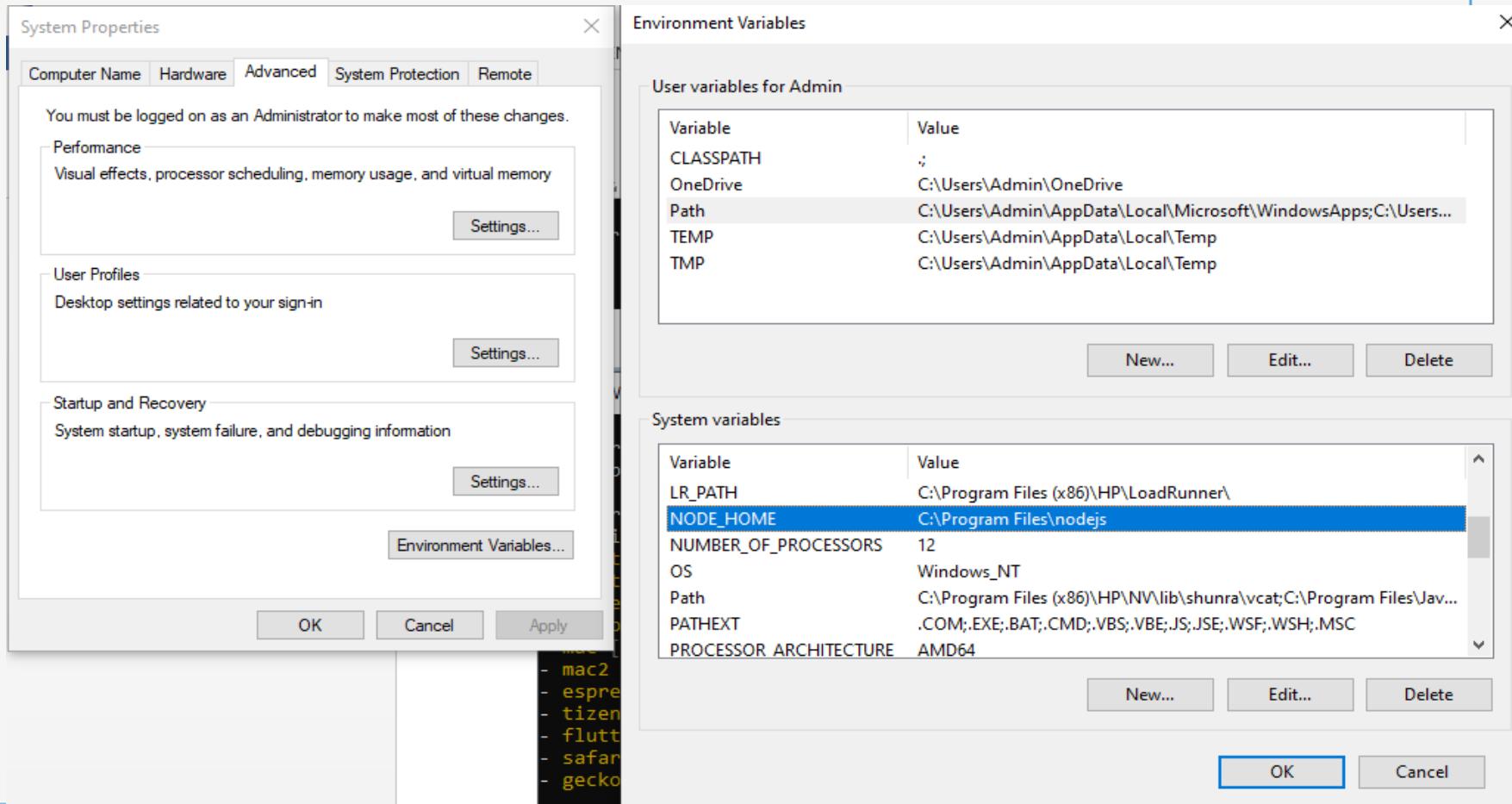
C:\Users\LearnVern>appium driver list
✓ Listing available drivers
- uiautomator2@2.10.1 [installed (NPM)]
- xcuitest@4.12.1 [installed (NPM)]
- youiengine [not installed]
- windows [not installed]
- mac [not installed]
- mac2 [not installed]
- espresso [not installed]
- tizen [not installed]
- flutter [not installed]
- safari [not installed]
- gecko [not installed]

C:\Users\LearnVern>appium driver install xcuitest
```

- Above command to install xcuitest driver

Installation of Node JS Setup

- 10). Node_Home path set in system variable path



Appium Doctor

- To check the Appium installation and dependencies, you can install **appium-doctor** from [here](#).
- **Appium-doctor** is an application tool to **verify Appium installation**.
- It shows all the missing things that you need to do.
- So, this will be very useful to run appium-doctor whenever you get any issue.
- It will install through **npm**.

```
C:\Users\ajain153>npm install appium-doctor -g
npm WARN deprecated authorize-ios@1.2.1: Moved into appium
npm WARN deprecated core-js@1.2.7: core-js@<2.6.5 is no longer
ctual version of core-js@2.
C:\Users\ajain153\AppData\Roaming\npm\appium-doctor -> C:\Users
appium-doctor.js
+ appium-doctor@1.10.0
added 326 packages in 51.248s
```

503 × 136

Limitation of Appium

- Microsoft Windows does not support running Appium Inspector.
- Appium does not allow the testing for Android versions lower than 4.2.
- Appium provides limited support for testing Hybrid applications.
- E.g., Switching action of application is not possible to test i.e., web app to native app and vice versa.

Competitors of Appium

- There are several tools available for automated testing mobile applications, such as Robotium, Appium, Experitest, Selendroid, Kobiton, and Testdroid, etc.
- They all are tough contestants for Appium.
- But Selendroid and Robotium are one of the top competitors of Appium.
- Let us know some differences and see how they differ from each other.

Appium vs Robotium

- **Appium** is a cross-platform tool that supports both iOS and Android. Whereas **Robotium** only supports Android.
- **Appium** supports various languages while **Robotium** only supports Java programming language.
- **Appium** does not require application source code/library, whereas **Robotium** tool requires application source code or library.
- **Appium** can be used to test native, web, and hybrid mobile applications, whereas **Robotium** can only test native and hybrid applications.
- **Appium** supports many frameworks like Selenium. But **Robotium** is not compatible with Selenium at all.
- In **Appium**, you don't have to reinstall the application for a small change. But **Robotium** code leads to complete rebuild for a small change.

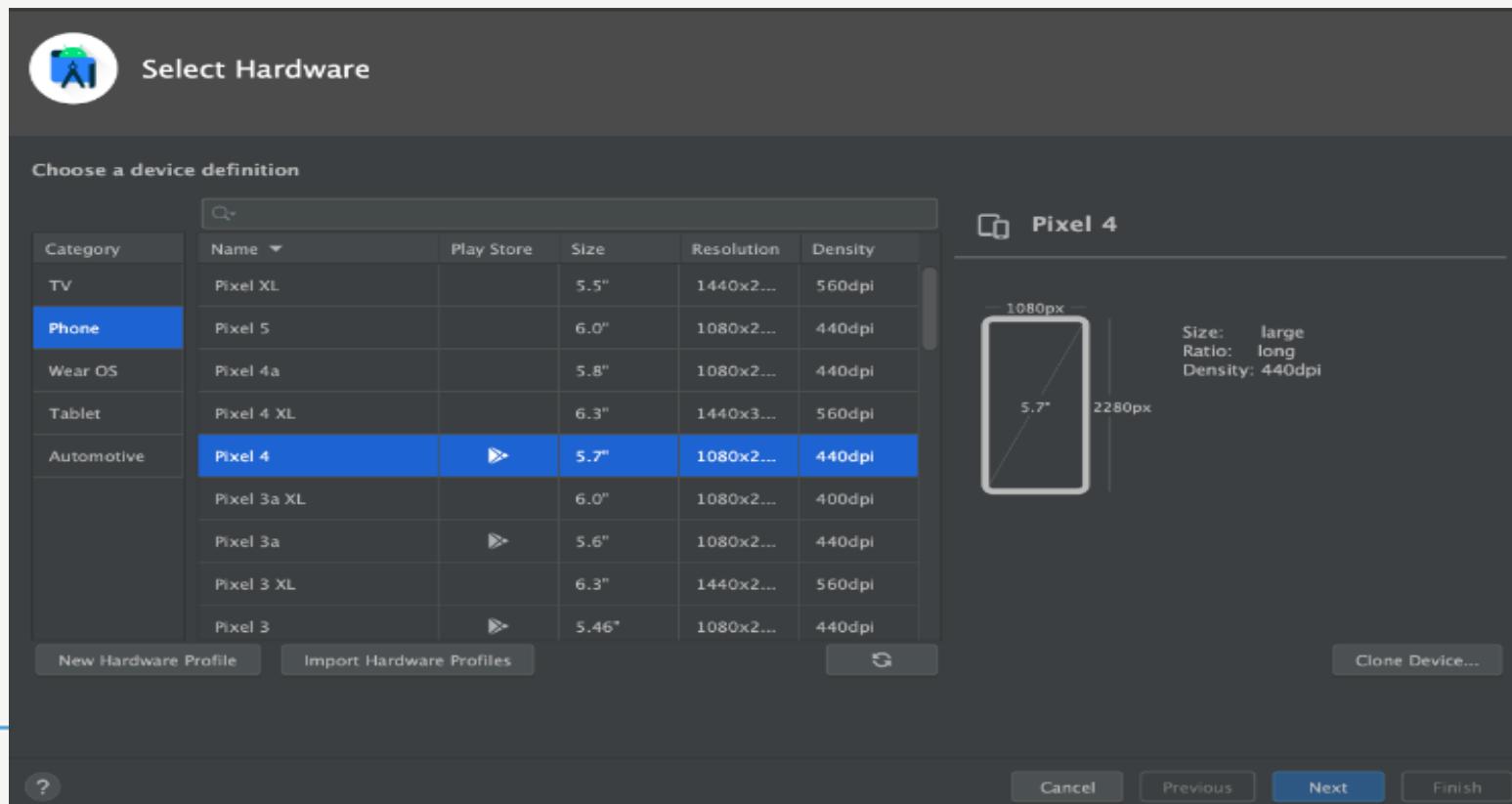
Appium vs Selendroid

- **Appium** is an open-source automation tool that supports both iOS and Android, while **Selendroid** is a test automation framework that only supports Android.
- In **Appium**, a small change does not require reinstallation of the application. But **Selendroid** requires reinstallation of the application.
- **Appium** has a strong and active community, whereas **Selendroid** does not have a strong community like Appium.
- **Appium** supports many frameworks and languages. On the other hand, **Selendroid** is compatible with Jenkins and Selenium.
- **Appium** does not require application source code/library, while **Selendroid** requires application source code or library.
- **Appium** supports all Android APIs with a limitation. Appium uses UIAutomator for tests running on API ≥ 17 , while for older APIs, it runs tests using Selendroid.

Emulator Setup: Create Virtual Android Device

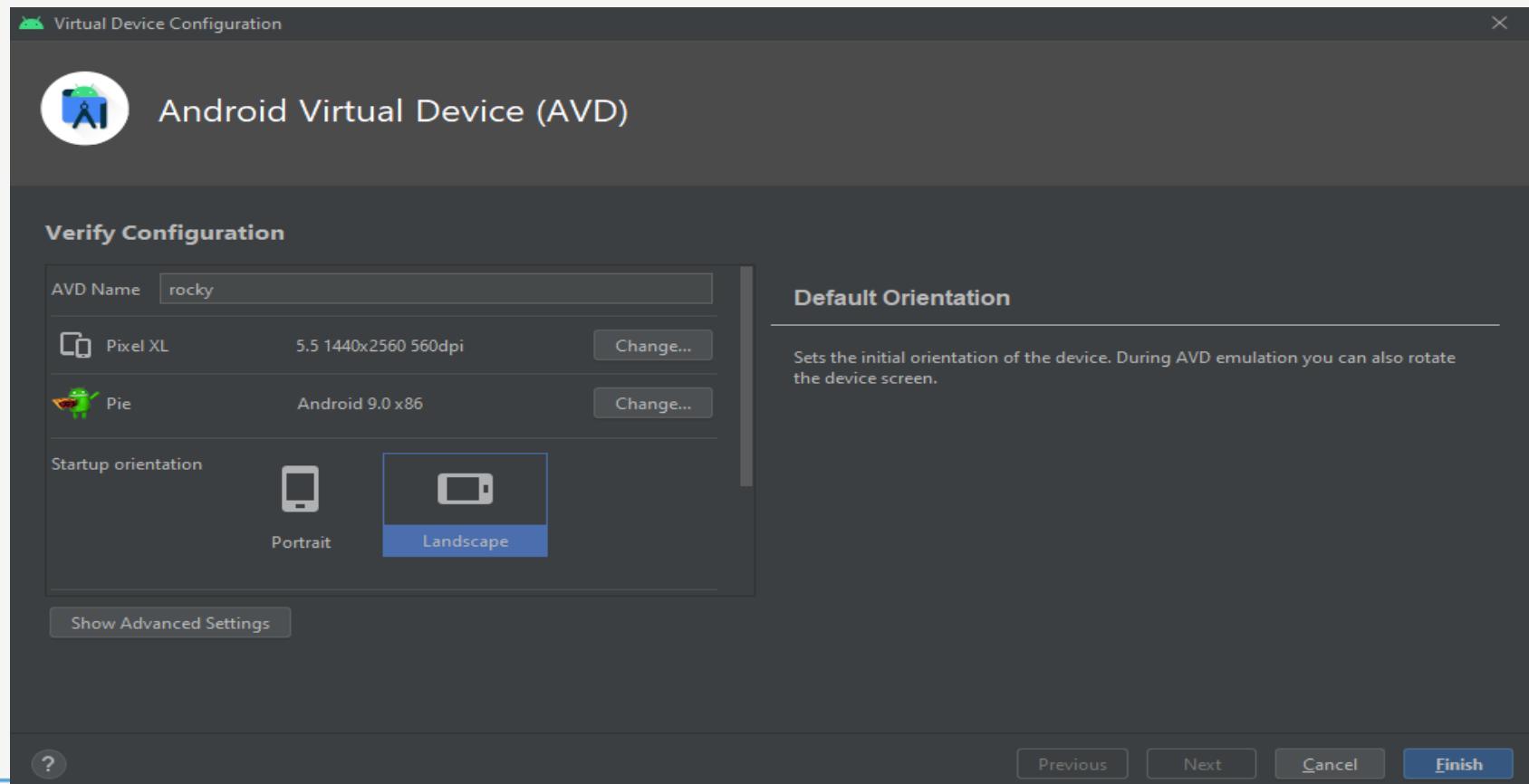
- 1) Open the Device Manager.
- 2) Click **Create Device**

The Select Hardware window appears.



Emulator Setup: Create Virtual Android Device

- 3) create avd with name.



Emulator Setup: Create Virtual Android Device

- 4) Select a hardware profile, and then click Next.

Select a System Image

Recommended x86 Images Other Images

Release Name	API Level	ABI	Target
Tiramisu	33	x86_64	Android Tiramisu
Sv2	32	x86_64	Android 12L (Go)
S	31	x86_64	Android 12.0 (Go)
R	30	x86	Android 11.0 (Go)
Q	29	x86	Android 10.0 (Go)
Pie	28	x86	Android 9.0 (Go)
Oreo	27	x86	Android 8.1 (Go)
Oreo	26	x86	Android 8.0 (Go)
Nougat	25	x86	Android 7.1.1 (Go)
Nougat	24	x86	Android 7.0 (Go)

Pie



API Level
28

Android
9.0

Google Inc.

System Image
x86

We recommend these images because they run the fastest support Google APIs.

Questions on API level?
See the [API level distribution chart](#)

OK Cancel

Emulator Setup: Create Virtual Android Device

- 5) Select the system image for a particular API level, and then click Next.
- 6) Change AVD properties as needed, and then click **Finish**
 - Click Show Advanced Settings to show more settings, such as the skin.
 - The new AVD appears in the Virtual tab of the Device Manager and the target drop-down menu.

Emulator Setup: Create Driver Session

```
UiAutomator2Options options = new UiAutomator2Options();
    options.setDeviceName("rocky");

    options.setApp("C:\\\\Users\\\\LearnVern\\\\Desktop\\\\p\\\\MyFirstTest\\\\src\\\\test
\\\\java\\\\resources\\\\ApiDemos-debug.apk");

    AndroidDriver driver = new AndroidDriver(new
URL("http://127.0.0.1:4723"),options);

    Thread.sleep(10000);
```

Real Device Setup: Enable USB Debugging

- On Android 4.1 and lower, the Developer options screen is available by default.
- On Android 4.2 and higher, you must enable this screen.
- To enable developer options, tap the Build Number option 7 times.
- You can find this option in one of the following locations, depending on your Android version:
 - Android 9 (API level 28) and higher: Settings > About Phone > Build Number
 - Android 8.0.0 (API level 26) and Android 8.1.0 (API level 26): Settings > System > About Phone > Build Number
 - Android 7.1 (API level 25) and lower: Settings > About Phone > Build Number
- Before you can use the debugger and other tools, you need to enable USB debugging,
- which allows Android Studio and other SDK tools to recognize your device when connected via USB. To enable USB debugging, toggle the USB debugging option in the Developer Options menu.

Real Device Setup: Enable USB Debugging

- You can find this option in one of the following locations, depending on your Android version:
- Android 9 (API level 28) and higher: Settings > System > Advanced > Developer Options > USB debugging
- Android 8.0.0 (API level 26) and Android 8.1.0 (API level 26): Settings > System > Developer Options > USB debugging
- Android 7.1 (API level 25) and lower: Settings > Developer Options > USB debugging

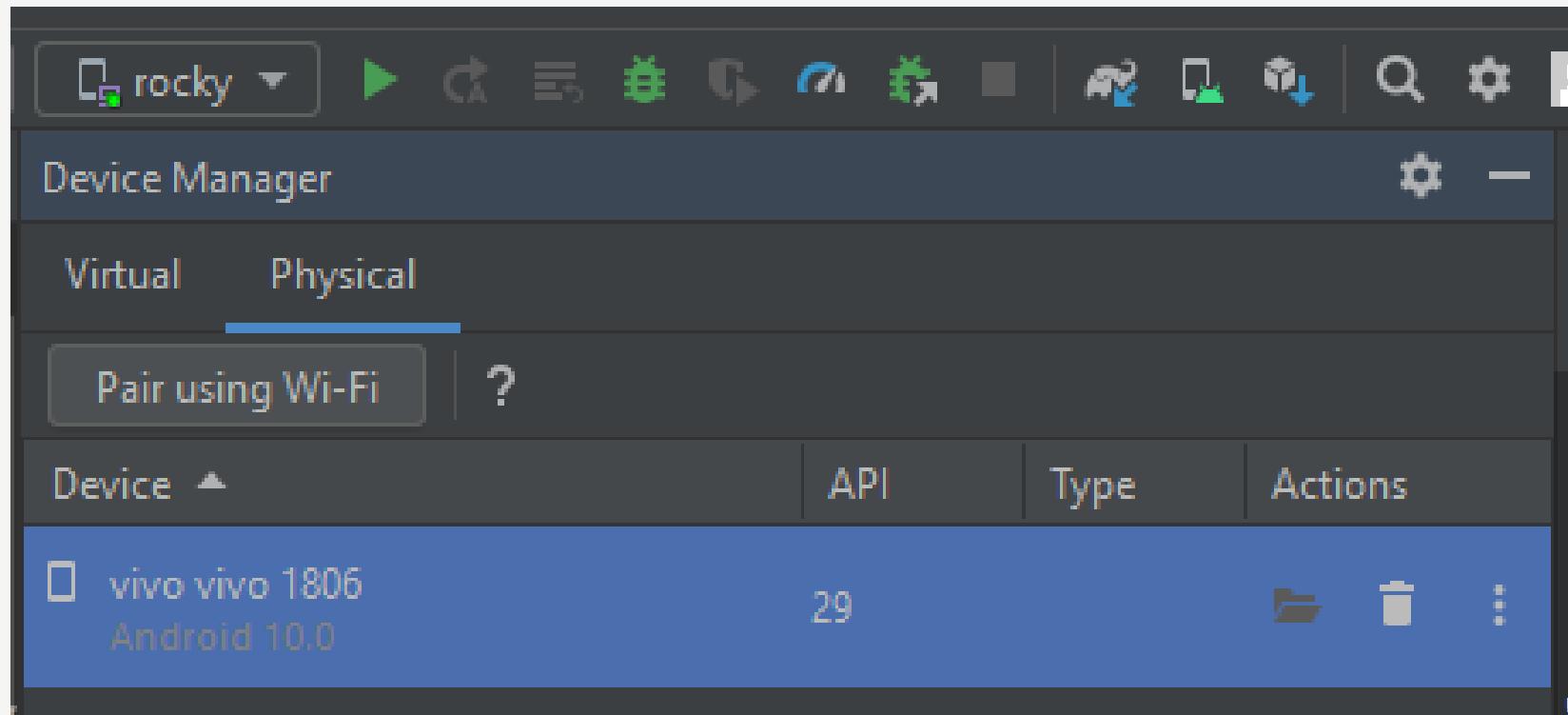
Real Device Setup: Create Driver Session

- Connect with mobile phone with cable
- Open you developer option on
- Then visible your device with physical tab

Virtual		Physical	
		Pair using Wi-Fi	
		?	
Device	▲	API	Type
Actions			
vivo vivo 1806		29	  
Android 10.0			

Real Device Setup: Create Driver Session

- To run the one time with any project so successfully run in few minute



Real Device Setup: Create Driver Session

```
DesiredCapabilities cap =new DesiredCapabilities();
    cap.setCapability("deviceName", "vivo 1806");
    cap.setCapability("udid", "QOKBUKS4PFQSV8X4");
    cap.setCapability("platformName", "Android");
    cap.setCapability("platformVersion", "10");
    cap.setCapability("appPackage", "com.android.bbkcalculator");
    cap.setCapability("appActivity",
"com.android.bbkcalculator.Calculator");
    cap.setCapability("automationName", "UiAutomator2");
    URL url=new URL("http://127.0.0.1:4723/");
driver=new AppiumDriver(url,cap);
```

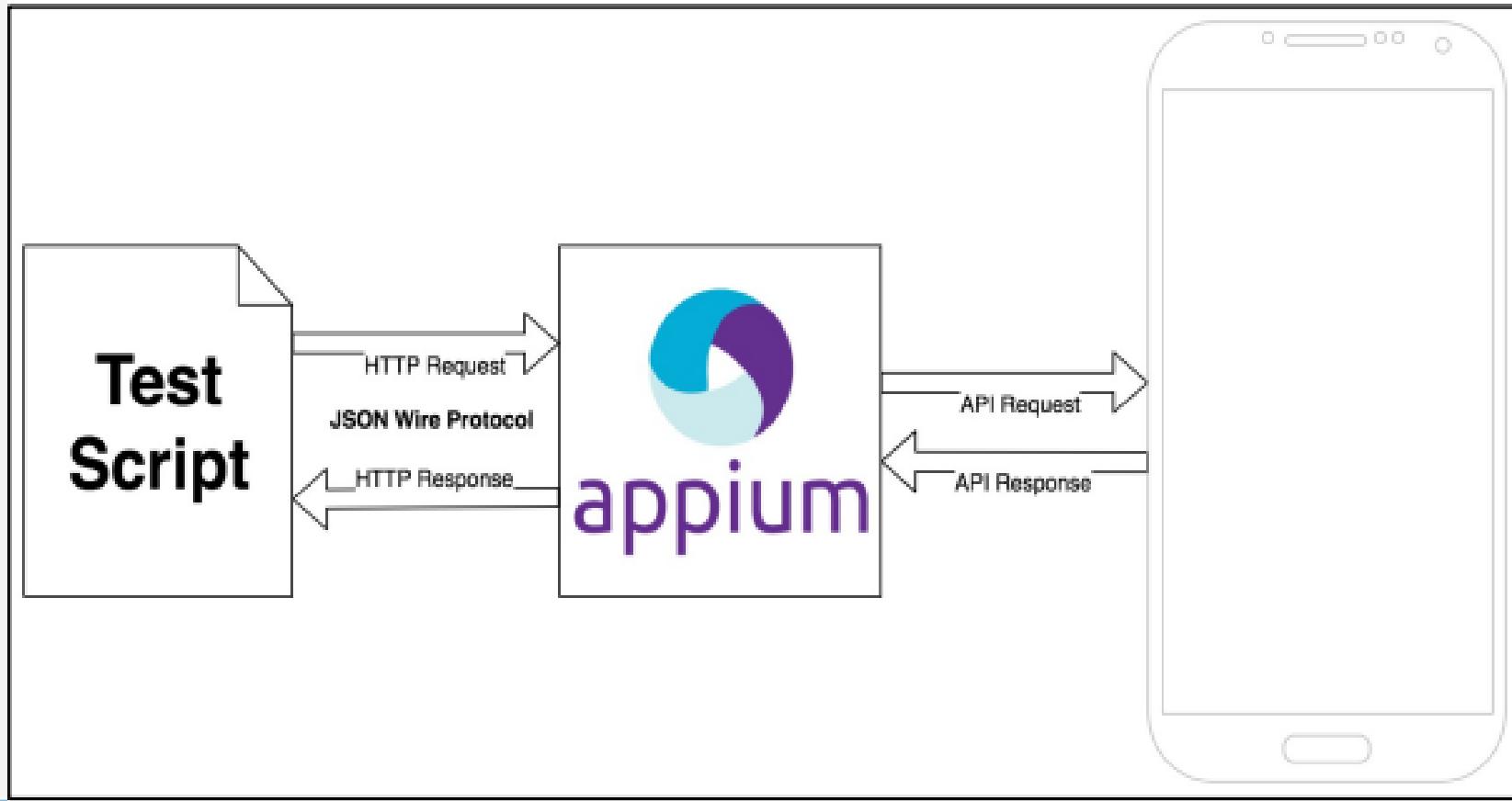
Project Set Up

What are Desired Capabilities?

- Desired Capabilities' help us to modify the behavior of server while Automation.
- In Appium, it is a type of hashmap or key-value pair, used to send a command to APPIUM server.
- In APPIUM, all the client commands are running in the context of a session.
- For example, a client sent POST/session request containing JSON object to APPIUM server.
- Hence, to send any desired request or to maintain any desired session with the server, a set of Key and value pair is used. This is known as '**Desired Capabilities.**'
- Desired Capabilities help us to configure the Appium server and provide the criteria which we wish to use for running our automation script.
- For example, we can request the environment (emulator or real-device), which version of the operating system to run the test on, and more.
- Desired Capabilities are key/value pairs encoded in JSON format and are sent to the Appium Server by the Appium client when a new automation session is requested

Project Set Up

What are Desired Capabilities?



Project Set Up

What are Desired Capabilities?

- Android:

```
{  
  "appium:app":  
    "C:\\\\Users\\\\LearnVern\\\\Desktop\\\\p\\\\MyFirstTest\\\\src\\\\test\\\\java\\\\resources\\\\ApiDe  
    mos-debug.apk",  
  "appium:deviceName": "rocky",  
  "platformName": "android",  
  "appium:automationName": "UIAutomator2"  
}
```

Project Set Up

What are Desired Capabilities?

```
DesiredCapabilities cap =new DesiredCapabilities();
    cap.setCapability("deviceName", "vivo 1806");
    cap.setCapability("udid", "QOKBUKS4PFQSV8X4");
    cap.setCapability("platformName", "Android");
    cap.setCapability("platformVersion", "10");
    cap.setCapability("appPackage", "com.android.bbkcalculator");
    cap.setCapability("appActivity",
"com.android.bbkcalculator.Calculator");
    cap.setCapability("automationName", "UiAutomator2");
    URL url=new URL("http://127.0.0.1:4723/");
driver=new AppiumDriver(url,cap);
```

Project Set Up

Important Role of Desired Capability-

- DesiredCapabilities' help the user to control the session request with the server.
- For example- if we want iOS session then we might set Capability as PlatformName = iOS. Or if we want Android session then we might set Capability as PlatformName = Android.
- DesiredCapabilities' are used to set up the Webdriver instance eg: FirefoxDriver, ChromeDriver, InternetExplorerDriver etc.
- DesiredCapability is very useful for Selenium Grid. Eg: It is used to access different test cases on a different browser and different operating system. Based on mentioned DesiredCapability Grid, hub will point to the corresponding node. Here, these nodes are defined using 'set' property method

Project Set Up

Important Role of Desired Capability-

```
DesiredCapabilities obj = new DesiredCapabilities();
obj.setBrowserName("firefox");
obj.setVersion("18.0.1");
obj.setPlatform(org.openqa.selenium.Platform.WINDOWS);
```

A desired capability is a library defined package. Prior to use 'DesiredCapabilities', it should be imported from below mentioned library
Org.openqa.selenium.remote.DesiredCapabilities

Project Set Up

Below table depicts some commonly used Android capabilities and its value to use-

Capabilities	Description	Values/Uses
appPackage	Call desired Java package in android that user want to run	Value= com.example.myapp/ Obj.setCapability("appPackage", "com.whatsapp");
appActivity	Application Activity that user wants to launch from the package.	Value= MainActivity, .Settings Obj.setCapability("appActivity", "com.whatsapp.Main");
appWaitPackage	Package from which application needs to wait for	Value=com.example.android.myapp
appWaitActivity	Any Android activity that user need wait time	Value= SplashActivity capabilities.setCapability("appWaitActivity", "com.example.game.SplashActivity")

Project Set Up

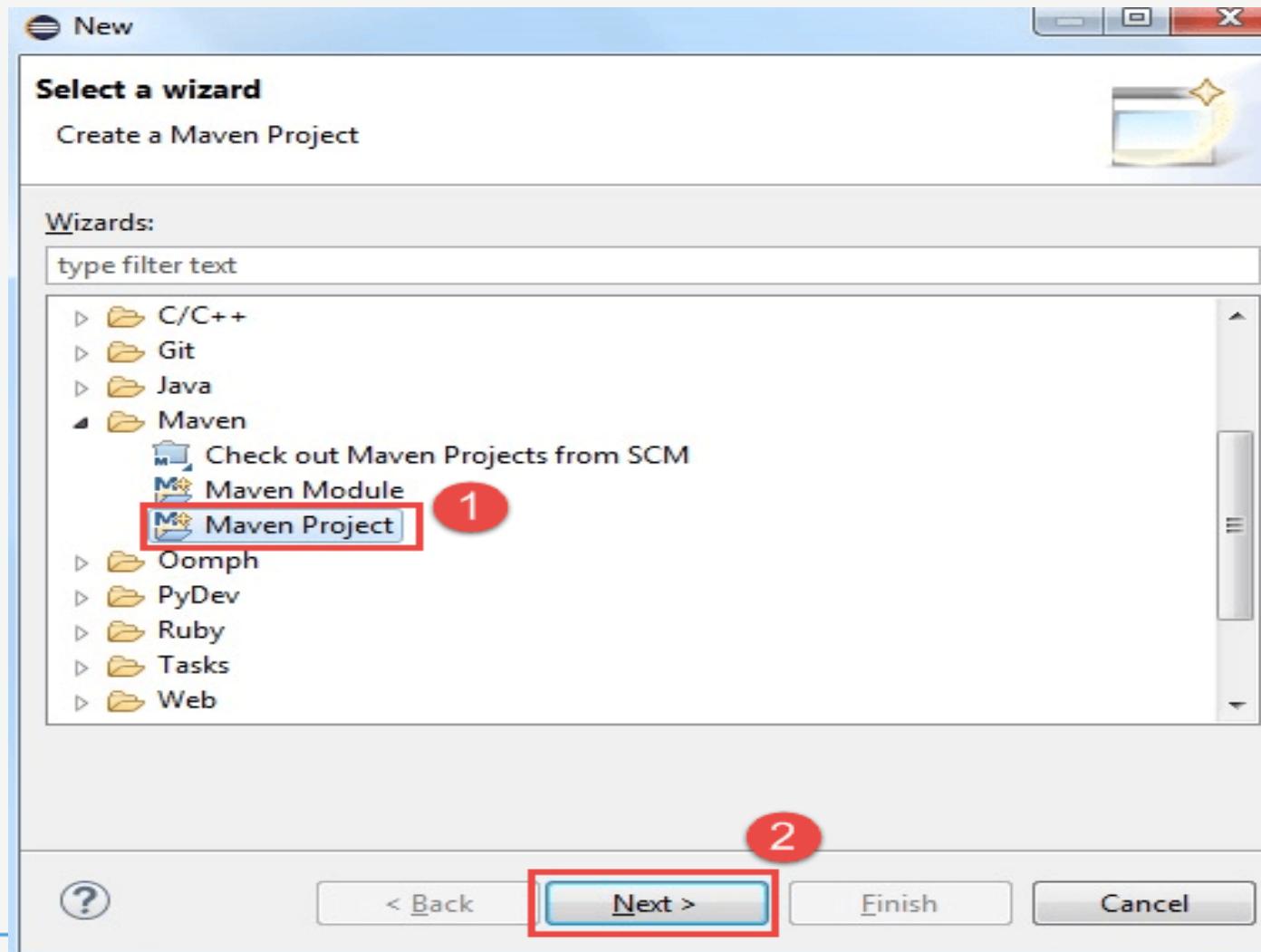
Below table depicts some commonly used iOS capabilities and its value to use-

Capabilities	Description	Values
LaunchTimeout	Total time (in ms) to wait for instrumentation.	2000
UDID	To identify unique device number for connected physical device	166aestu4

Create java project using maven

- After configuring Appium Java Maven plug-in Eclipse.
- It will be ready to test any android .apk application with Appium and Maven as shown in the Appium Maven project example below.
- **Step 1)** In this step,
- Go to NEW >> select Maven project
- Click on 'next' button

Create java project using maven



Create java project using maven

- Step 2) Then in ‘New Maven Project’ window, enter ‘Appium Test’ in Group Id and Artifact Id column. In this step, you have to enter.

Group Id

Artifact Id

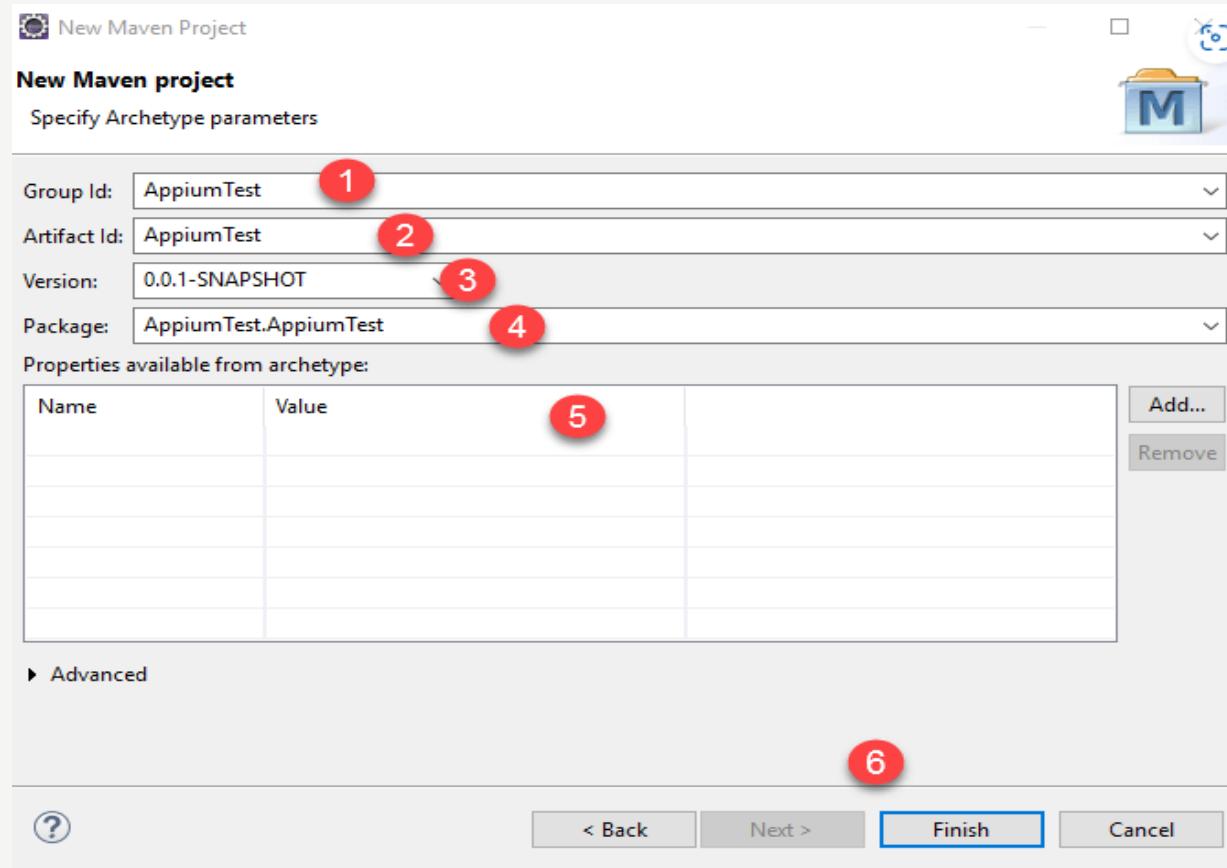
Version

Packaging

Name and Description

Finish

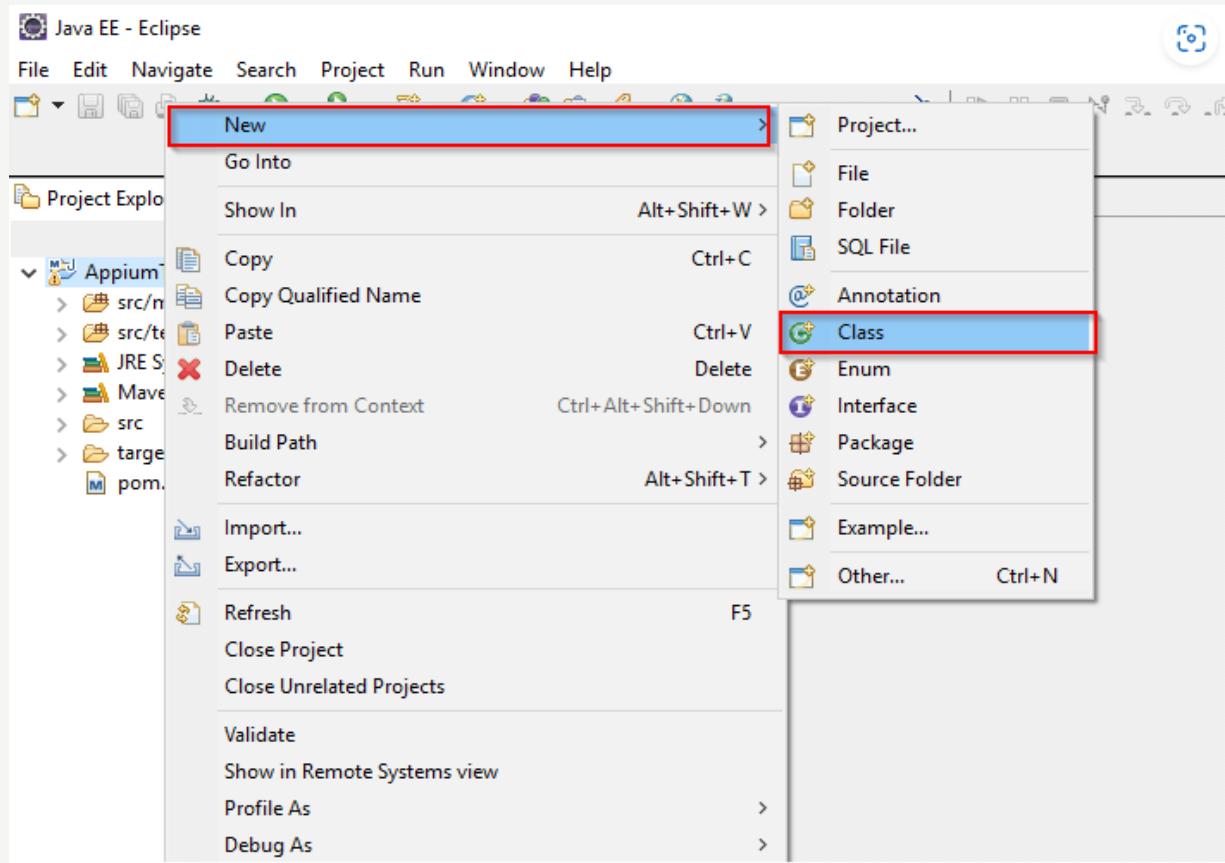
Create java project using maven



Clicking on Finish button. It will open a new class on the defined Group Id (AppiumTest) name.

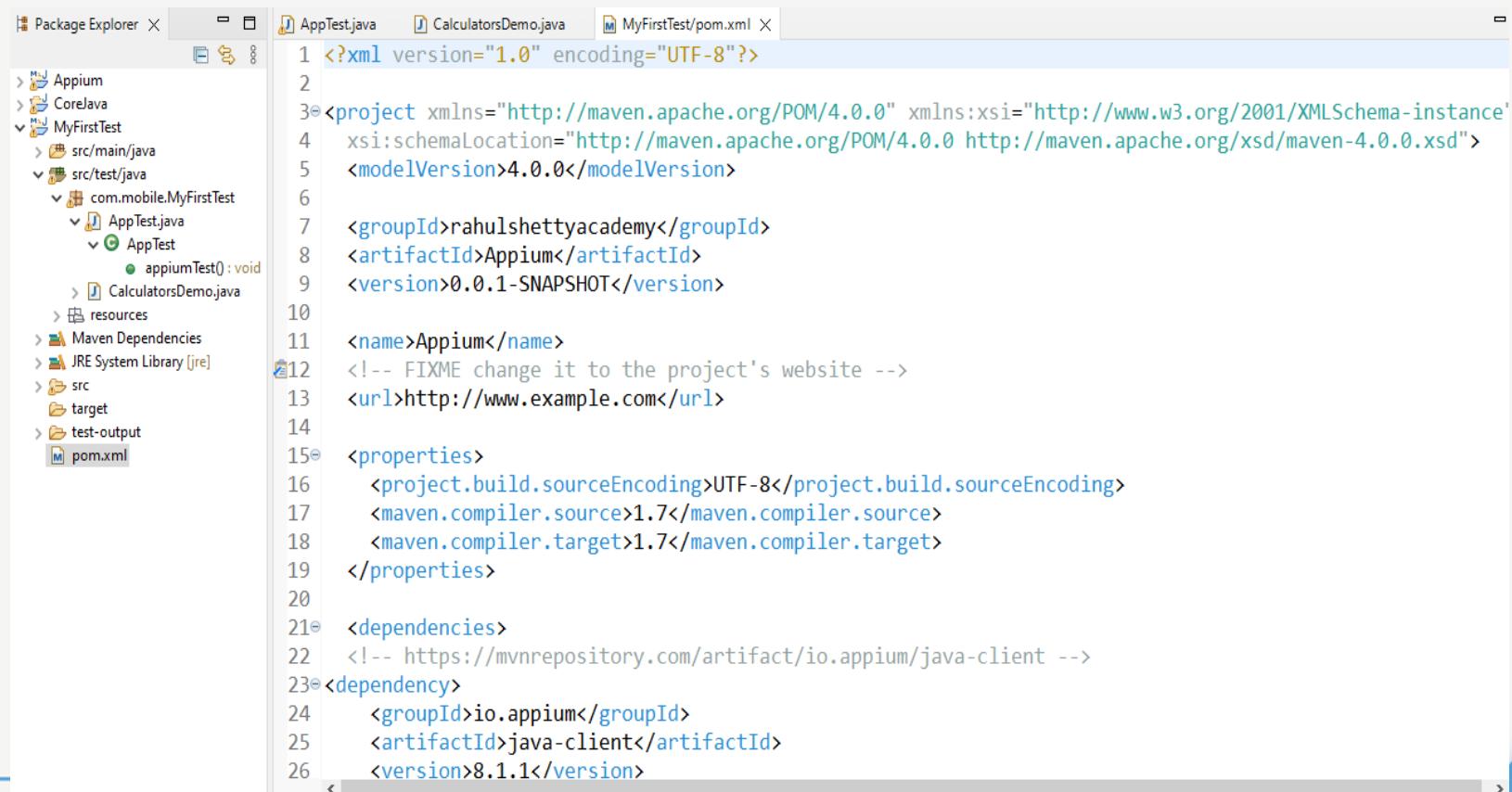
Create java project using maven

- **Step 3)** To start with Appium script. Right click on ‘src/main/java’ from left side explorer window. Then select New >> class. Write the Appium code inside the selected class.



Create java project using maven

- **Step 4)** In the same project, click over pom.xml from left explorer menu. All dependencies will be visible by default in ‘pom.xml’ tab. Refer to the Image below-



The screenshot shows the Eclipse IDE interface with the Package Explorer view on the left and the Editor view on the right.

Package Explorer View:

- Project name: Appium
- Source folders:
 - src/main/java
 - src/test/java
 - com.mobile.MyFirstTest
 - AppTest.java
 - AppTest
 - appiumTest() : void
 - CalulatorsDemo.java
 - resources- Maven Dependencies
- JRE System Library [jre]
- src
- target
- test-output
- pom.xml

Editor View (pom.xml tab):

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>rahulshettyacademy</groupId>
  <artifactId>Appium</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>Appium</name>
  <!-- FIXME change it to the project's website -->
  <url>http://www.example.com</url>
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.7</maven.compiler.source>
    <maven.compiler.target>1.7</maven.compiler.target>
  </properties>
  <dependencies>
    <!-- https://mvnrepository.com/artifact/io.appium/java-client -->
    <dependency>
      <groupId>io.appium</groupId>
      <artifactId>java-client</artifactId>
      <version>8.1.1</version>
    </dependency>
  </dependencies>
</project>
```

Create java project using maven

- If in the case of default pom.xml does not exist then just add all Maven Appium dependencies. (extracted from Maven central repository website}
- <http://search.maven.org/#search|gav|1|g%3A%22io.appium%22%20AND%20a%3A%22java-client%22>

```
<properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.7</maven.compiler.source>
    <maven.compiler.target>1.7</maven.compiler.target>
</properties>

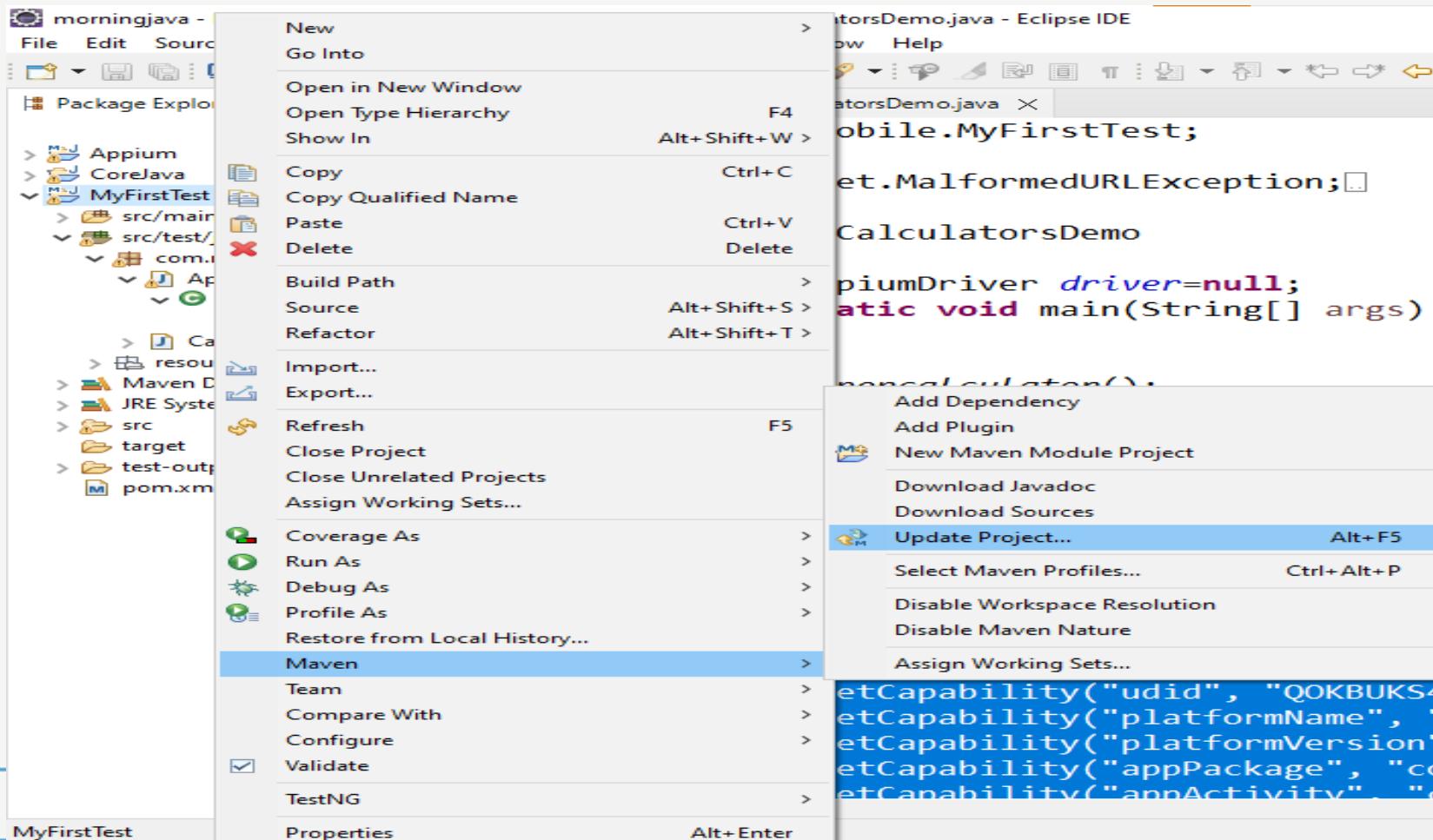
<dependencies>
    <!-- https://mvnrepository.com/artifact/io.appium/java-client -->
<dependency>
    <groupId>io.appium</groupId>
    <artifactId>java-client</artifactId>
    <version>8.1.1</version>
</dependency>
    <!-- https://mvnrepository.com/artifact/org.testng/testng -->
<dependency>
    <groupId>org.testng</groupId>
    <artifactId>testng</artifactId>
    <version>7.6.0</version>
</dependency>
    <!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-support -->
<dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-support</artifactId>
    <version>4.2.0</version>
</dependency>
```

Create java project using maven

```
<build>
  <pluginManagement><!-- lock down plugins versions to avoid using Maven defaults (may be moved to parent pom) -->
    <plugins>
      <!-- clean lifecycle, see https://maven.apache.org/ref/current/maven-core/lifecycles.html#clean_Lifecycle -->
      <plugin>
        <artifactId>maven-clean-plugin</artifactId>
        <version>3.1.0</version>
      </plugin>
      <!-- default lifecycle, jar packaging: see https://maven.apache.org/ref/current/maven-core/default-bindings.html#Plugin_bindings_for_jar_packaging -->
      <plugin>
        <artifactId>maven-resources-plugin</artifactId>
        <version>3.0.2</version>
      </plugin>
      <plugin>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.8.0</version>
      </plugin>
      <plugin>
        <artifactId>maven-surefire-plugin</artifactId>
        <version>2.22.1</version>
      </plugin>
      <plugin>
        <artifactId>maven-jar-plugin</artifactId>
        <version>3.0.2</version>
      </plugin>
      <plugin>
        <artifactId>maven-install-plugin</artifactId>
        <version>2.5.2</version>
      </plugin>
      <plugin>
        <artifactId>maven-deploy-plugin</artifactId>
        <version>2.8.2</version>
      </plugin>
      <!-- site lifecycle, see https://maven.apache.org/ref/current/maven-core/lifecycles.html#site_Lifecycle -->
      <plugin>
        <artifactId>maven-site-plugin</artifactId>
        <version>3.7.1</version>
      </plugin>
      <plugin>
        <artifactId>maven-project-info-reports-plugin</artifactId>
        <version>3.0.0</version>
      </plugin>
    </plugins>
  </pluginManagement>
  <plugins>
    <!-- clean lifecycle, see https://maven.apache.org/ref/current/maven-core/lifecycles.html#clean_Lifecycle -->
    <plugin>
      <artifactId>maven-clean-plugin</artifactId>
      <version>3.1.0</version>
    </plugin>
    <!-- default lifecycle, jar packaging: see https://maven.apache.org/ref/current/maven-core/default-bindings.html#Plugin_bindings_for_jar_packaging -->
    <plugin>
      <artifactId>maven-resources-plugin</artifactId>
      <version>3.0.2</version>
    </plugin>
    <plugin>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.8.0</version>
    </plugin>
    <plugin>
      <artifactId>maven-surefire-plugin</artifactId>
      <version>2.22.1</version>
    </plugin>
    <plugin>
      <artifactId>maven-jar-plugin</artifactId>
      <version>3.0.2</version>
    </plugin>
    <plugin>
      <artifactId>maven-install-plugin</artifactId>
      <version>2.5.2</version>
    </plugin>
    <plugin>
      <artifactId>maven-deploy-plugin</artifactId>
      <version>2.8.2</version>
    </plugin>
    <!-- site lifecycle, see https://maven.apache.org/ref/current/maven-core/lifecycles.html#site_Lifecycle -->
    <plugin>
      <artifactId>maven-site-plugin</artifactId>
      <version>3.7.1</version>
    </plugin>
    <plugin>
      <artifactId>maven-project-info-reports-plugin</artifactId>
      <version>3.0.0</version>
    </plugin>
  </plugins>
</build>
```

Create java project using maven

- Step 6) to update all dependency and plugin for that project



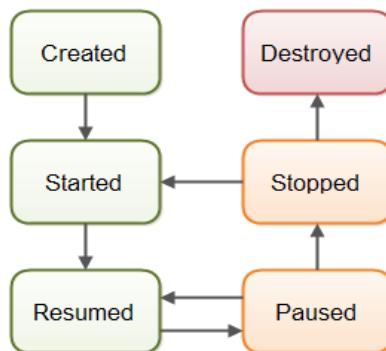
Android: How to get appPackage and appActivity?

WHAT IS ANDROID APP PACKAGE?

- An Android app package is a .apk file by which Android operating system distributes and installs the Android application and its middleware.
- APK file or app package is analogous to software packages of other platforms like .exe file, APPX file, etc

WHAT IS ANDROID APP ACTIVITY?

- Android app activity is the screen of the Android app. It's a kind of a window, like Microsoft windows OS for Laptop or desktop device.
- It has the **Main screen** and from here we can jump to any other screens for performing different **activities**



HOW ARE ANDROID APP PACKAGE AND ANDROID APP ACTIVITY IMPORTANT FOR APPUIUM TESTING?

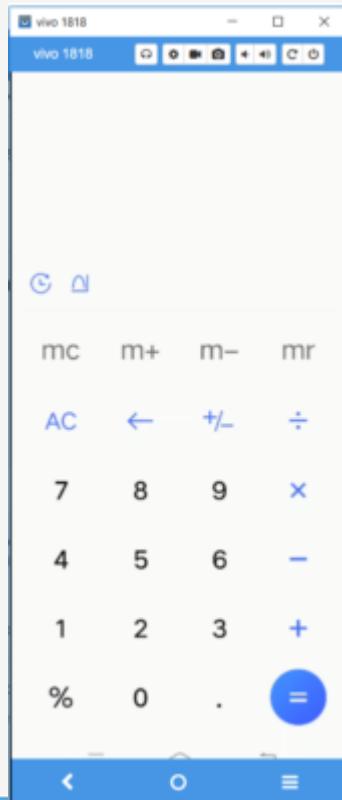
- Android App package is the destination of your apk file or installed file in your Android device, whereas Android App Activity is the destination path of the screen which will be launched by Appium server.
- We give the name of that screen through Android App Activity at which we are going to perform the testing activities. After setting all the capabilities,
- Appium first launches the app through the app package, then it navigates to the desired screen through Android app activity path.

HOW TO GET ANDROID APP PACKAGE AND ANDROID APP ACTIVITY DETAILS OF ANY ANDROID APP?

- There are various ways to get these details; but, here I am going to tell you the easiest and efficient ways to collect app package and app activity details. Let's start with command line technique.

STEP #1: OPEN TESTABLE APP IN YOUR DEVICE

Suppose I need App package and app activity details of calculator main page.



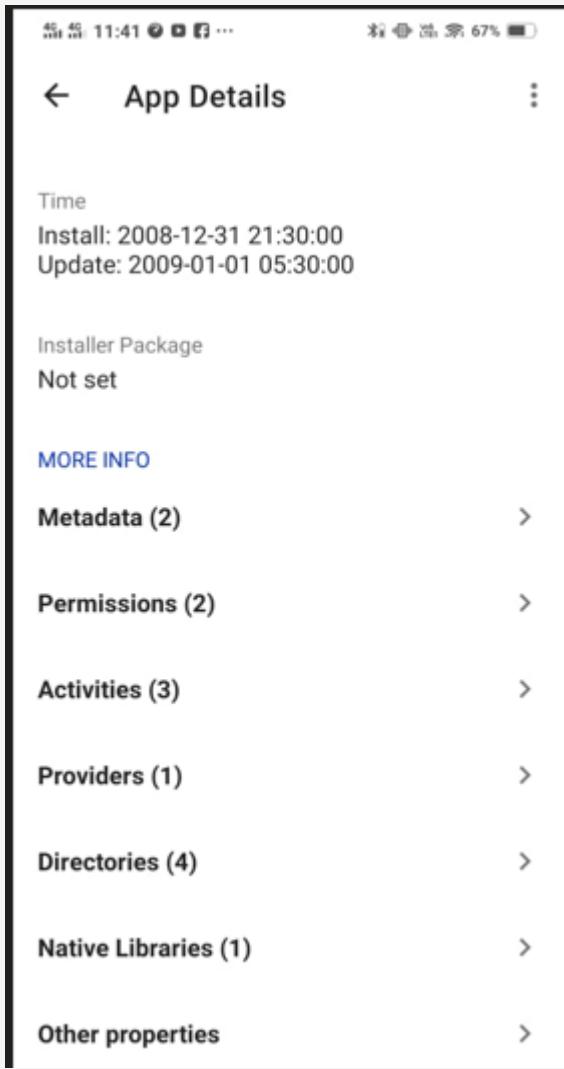
Step 2 download and install APKinfo application



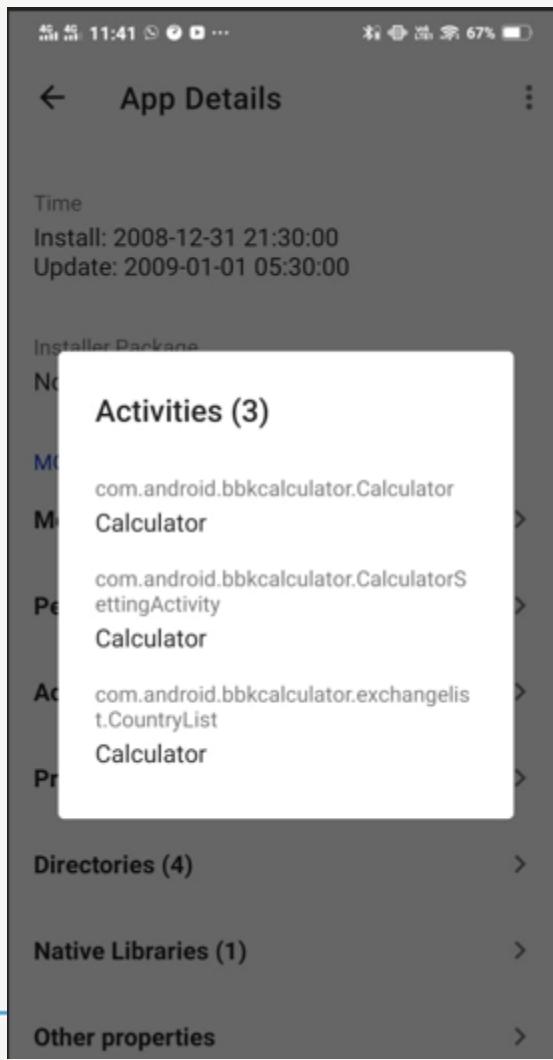
Step 3) to select the calculator app



Step 4) go to activity option then open the activity option



Step 5) to select appactivity name



Native app automation

● Appium Inspector Walk-through

- Appium Inspector is a GUI inspector for mobile apps and more, powered by a (separately installed) Appium server,
 - This gives the users the ability to automate mobile scripts with a flexible UI.
 - You can connect to a local server or to a cloud.
-
- You have two ways of connecting to SeeTest cloud:
 - Connect to your Seetest Cloud (Appium server tab)
 - Connect to your SeeTest Cloud instance (Experitest tab is available in the Appium Inspector version 2021.12.2)

Connect to your See test Cloud (Appium server tab)

- Open the Appium Inspector
- Click on Appium server tab
- Add the remote host and the port (no need to add the protocol with the host)
- Add "/wd/hub" to the remote path
- Check the SSL check box for secure cloud
- The accessKey needs to be added as part of the capabilities "experitest:accessKey"

Connect to your See test Cloud (Appium server tab)

Appium Server  experitest Select Cloud Providers

Remote Host	uscloud.experitest.com	Remote Port	443
Remote Path	/wd/hub	<input checked="" type="checkbox"/> SSL	
> Advanced Settings			

Desired Capabilities Saved Capability Sets 7 Attach to Session...

appium:automationName	text	XCUITest	
platformName	text	iOS	
appium:deviceName	text	auto	
appium:udid	text	f6b55ce8gfdgdfh4ujh3ce7b0	
appium:appiumVersion	text	1.20.2	
experitest:accessKey	text	eyJ4cC51Iewrtyhgfcgfhkhk	

Automatically add necessary Appium vendor prefixes on start

JSON Representation

```
{
  "appium:automationName": "XCUITest",
  "platformName": "iOS",
  "appium:deviceName": "auto",
  "appium:udid": "f6b55ce8gfdgdfh4ujh3ce7b02e4c2c6813dcf5ab2dc816f8af0",
  "appium:appiumVersion": "1.20.2",
  "experitest:accessKey": "eyJ4cC51Iewrtyhgfcgfhkhk"
}
```

[Desired Capabilities Documentation](#)

Connect to your See test Cloud (Appium server tab)

- You can specify which appium version to use by changing the project version from the cloud or by adding "**appium:appiumVersion**" capability with a version that exists in the cloud.

appium:appiumVersion

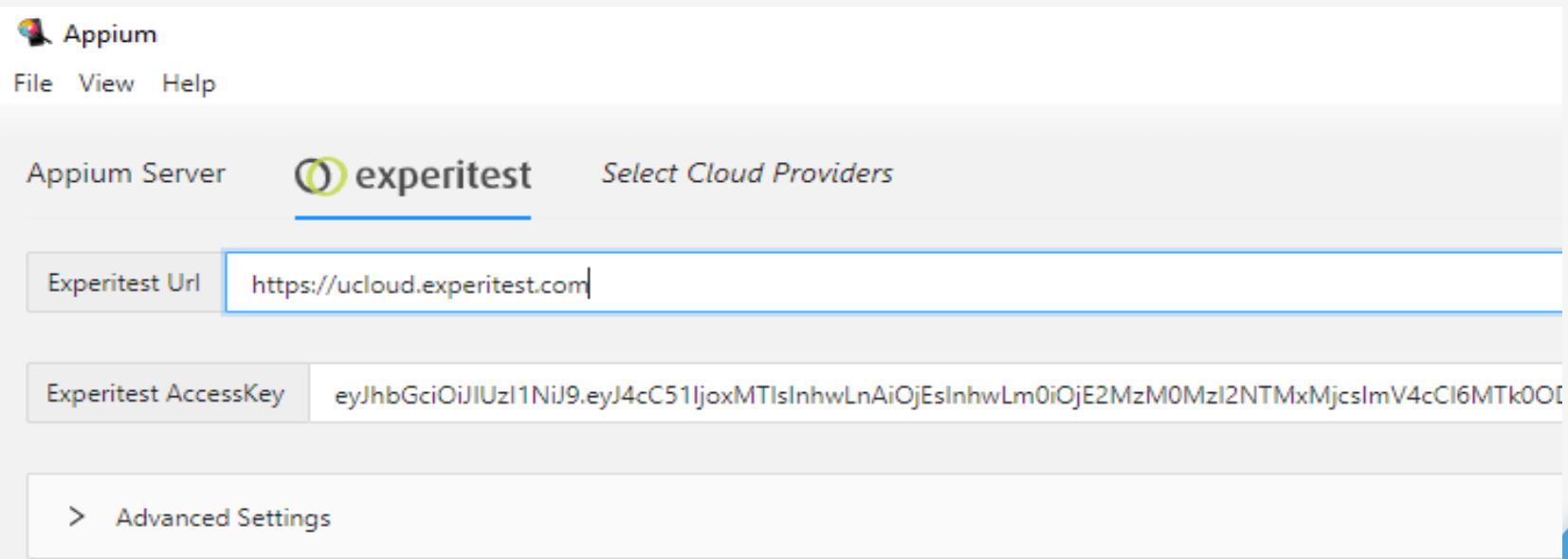
text

v

1.20.1

Connect to your SeeTest Cloud instance (Experitest tab is available in the Appium Inspector version 2021.12.2)

- Open the Appium Inspector
 - Click on Select cloud providers
 - Select Experitest
 - Populate the cloud URL and the AccessKey



Populate the Desired Capabilities

- We need to populate the capabilities in order to point the Appium instance to a device we want to work with.
- Appium inspector is expected to work by default with Appium 2.0 which is based on the W3C client, and its capabilities (W3C capabilities)
- so for it to work we need to add the "appium:" prefix to all of appium capabilities we want to use (appium capabilities)

Populate the Desired Capabilities

iOS Basic Capabilities:

automationName - XCUITest

deviceName - Name of the target device, but can be generic such as "Samsung"

platformName - iOS

udid - Device Serial Number

Desired Capabilities Saved Capability Sets 7 Attach to Session...

appium:automationName	text	XCUITest
platformName	text	iOS
appium:deviceName	text	iPhone
appium:udid	text	f6b55ce8ce7b02e4c2c6813d

Automatically add necessary Appium vendor prefixes on start

Populate the Desired Capabilities

iOS Browser Capabilities:

You need to add to the basic capabilities:

browserName - safari

Desired Capabilities Saved Capability Sets 7 Attach to Session...

appium:automationName	text	XCUITest	
platformName	text	iOS	
appium:deviceName	text	iPhone	
appium:udid	text	f6b55ce8ce7b02e4c2c6813d	
browserName	text	safari	

Automatically add necessary Appium vendor prefixes on start

Populate the Desired Capabilities

iOS Application Capabilities:

- You need to add to the basic capabilities:
to Install & Launch an Application, the following
capabilities are required:
 - **app** - cloud:<Name of Bundle Identifier>
 - **bundleId** - Bundle Identifier of the Application
- To simply Launch an Application, the following
capabilities are required:
- **bundleId** - Bundle Identifier of the Application

Populate the Desired Capabilities

iOS Application Capabilities :

appium:automationName	text	XCUITest
platformName	text	iOS
appium:udid	text	f6b55ce8ce7b02e4c2c6813d
appium:app	text	cloud:com.experitest.ExperiB
appium:bundleId	text	com.experitest.ExperiBank

- Automatically add necessary Appium vendor prefixes on start

Populate the Desired Capabilities

Android Basic Capabilities

automationName - UIAutomator2

deviceName - Name of the target device, but can be generic such as "Samsung"

platformName - Android

udid - Device Serial Number

appium:automationName	text	UIAutomator2
platformName	text	Android
appium:deviceName	text	auto
appium:udid	text	R522WBY1DRY5RENA0QCE8

Automatically add necessary Appium vendor prefixes on start

Populate the Desired Capabilities

Android Browser Capabilities:

You need to add to the basic capabilities:

browserName - Chrome

appium:automationName	text	UIAutomator2
platformName	text	Android
appium:deviceName	text	auto
appium:udid	text	RF8N735NR5H82N2SR67ZY1
browserName	text	Chrome

- Automatically add necessary Appium vendor prefixes on start

Populate the Desired Capabilities

Android Application Capabilities:

You need to add to the basic capabilities:

to Install & Launch an Application, the following capabilities are required:

app - cloud:<Name of AppPackage/appActivity>

appPackage - Package Name of the Application

appActivity - Activity Name of the Application

To simply Launch an Application, the following capabilities are required:

appPackage - Package Name of the Application

appActivity - Activity Name of the Application

Populate the Desired Capabilities

Android Application Capabilities:

appium:automationName	text	UIAutomator2
platformName	text	Android
appium:deviceName	text	auto
appium:udid	text	RF8N3JF5IE46VS735NZYN
appium:app	text	cloud:com.experitest.ExperiB
appium:appPackage	text	com.experitest.ExperiBank
appium:appActivity	text	.LoginActivity
<input checked="" type="checkbox"/> Automatically add necessary Appium vendor prefixes on start		

Populate the Desired Capabilities

IMPORTANT NOTES:

- You can use the checkbox, to add the Appim prefix automatically
 Automatically add necessary Appium vendor prefixes on start
- You can also use the web application for Appium inspector, but for it to work you need to add to the server conf file the property allow-cors (you can read about it here)

Attach to an existing session

- It allows you to attach to an existing session by providing just the session-id (as shown in the following screenshot). This comes in handy when you already have an Appium session, and you are in the middle of a running test. Attaching to an existing Appium session is possible because the inspector is just an Appium client:

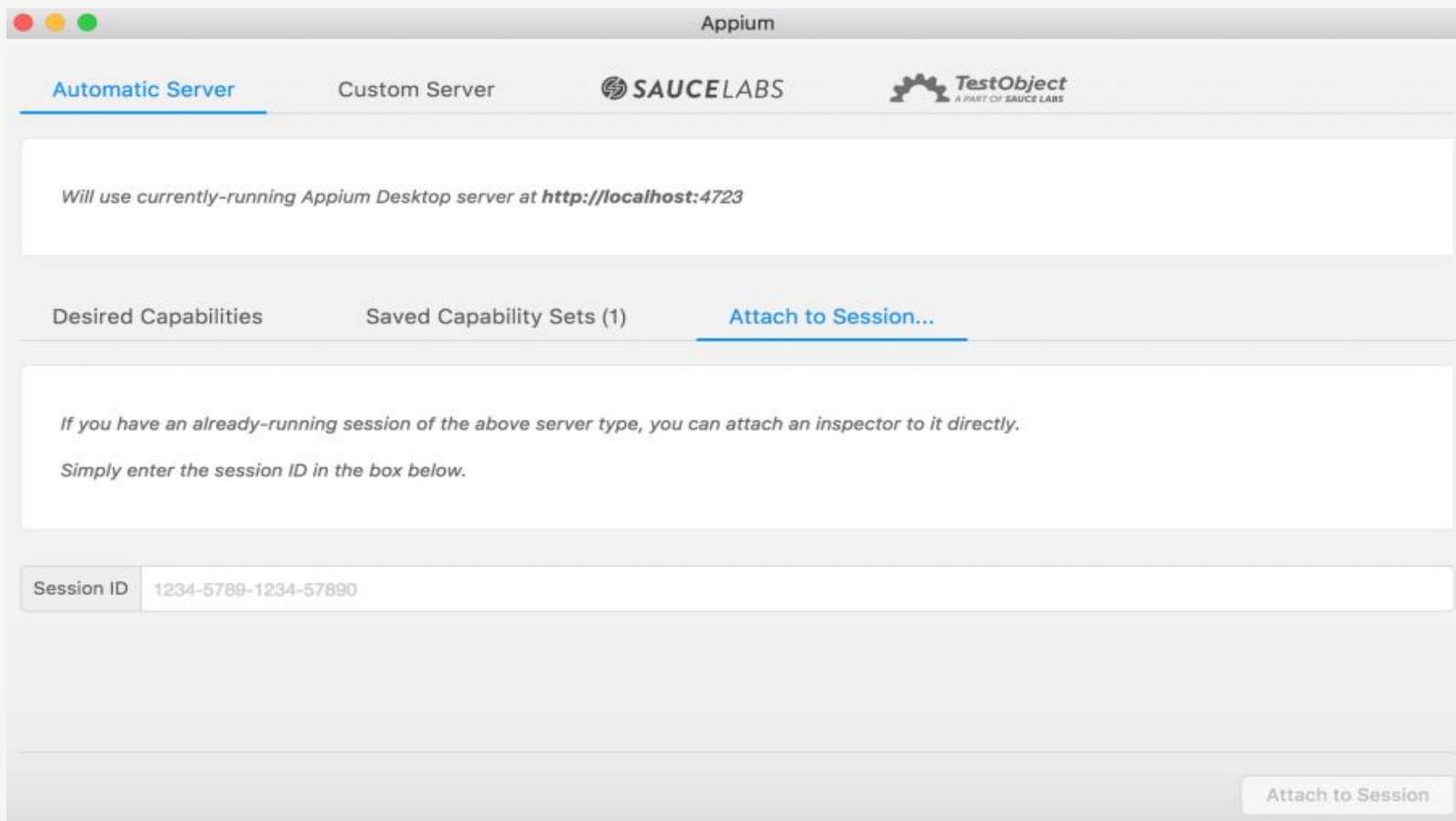
Populate the Desired Capabilities

Attach to an existing session

- It allows you to attach to an existing session by providing just the session-id (as shown in the following screenshot).
- This comes in handy when you already have an Appium session, and you are in the middle of a running test.
- Attaching to an existing Appium session is possible because the inspector is just an Appium client:

Populate the Desired Capabilities

Attach to an existing session



The screenshot shows the Appium desktop application window. At the top, there are three colored window control buttons (red, yellow, green) on the left and the word "Appium" on the right. Below the title bar, there are two tabs: "Automatic Server" (which is underlined in blue) and "Custom Server". To the right of these tabs are logos for "SAUCE LABS" and "TestObject A PART OF SAUCE LABS".

In the main content area, a message states: "Will use currently-running Appium Desktop server at <http://localhost:4723>".

Below this message, there are three buttons: "Desired Capabilities", "Saved Capability Sets (1)", and "Attach to Session...". The "Attach to Session..." button is underlined in blue, indicating it is the active tab.

Under the "Attach to Session..." tab, there is a note: "If you have an already-running session of the above server type, you can attach an inspector to it directly. Simply enter the session ID in the box below." Below this note is a text input field containing the session ID "1234-5789-1234-57890".

At the bottom right of the window, there is a "Attach to Session" button.

Android: XML and Element Attributes

The six most common views are:

TextView displays a formatted text label

<https://developer.android.com/reference/android/widget/TextView#xml-attributes>

ImageView displays an image resource

<https://developer.android.com/reference/android/widget/ImageView>

Button can be clicked to perform an action

<https://developer.android.com/reference/android/widget/Button>

ImageButton displays a clickable image

<https://developer.android.com/reference/android/widget/ImageButton>

EditText is an editable text field for user input

<https://developer.android.com/reference/android/widget/EditText>

ListView is a scrollable list of items containing other views

<https://developer.android.com/reference/android/widget/ListView>

Android: XML and Element Attributes

View Identifiers

- Any view can have an identifier attached that uniquely names that view for later access. You can assign a view an id within the XML layout:

```
<Button android:id="@+id/my_button" />
```

- This id can then be accessed within the Java code for the corresponding activity (in onCreate of Activity for example):
- Java: **Button myButton = findViewById(R.id.my_button);**
- Another important note is that any view with an id specified will automatically retain its state on a configuration change (i.e phone orientation switch).

Android: XML and Element Attributes

View Height and Width

- Every view has height and width properties controlling the size of the view. Height and width have to be defined in the XML for every view with:
- `<TextView android:layout_width="165dp"
 android:layout_height="wrap_content" />`
- This can take the form of wrap_content (adjust height and width to the content size), match_parent (adjust height and width to the full size of the parent container), and a dimensions value such as 120dp.
- This can be changed at runtime in a number of ways:

Android: XML and Element Attributes

Views Margin and Padding

Margins and padding values for views allows us to position and space elements in a layout.

Layout Margin defines the amount of space around the outside of a view
Padding defines the amount of space around the contents or children of a view.

An example of setting margins and padding:

```
<LinearLayout>
    <TextView android:layout_margin="5dp" android:padding="5dp">
        <Button layout_marginBottom="5dp">
    </LinearLayout>
```

Android: XML and Element Attributes

View Gravity

Gravity can be used to define the direction of the contents of a view.

gravity determines the direction that the contents of a view will align (like CSS text-align).

layout_gravity determines the direction of the view within it's parent (like CSS float).

An example of applying gravity:

```
<TextView  
    android:gravity="left"  
    android:layout_gravity="right"  
    android:layout_width="165dp" android:layout_height="wrap_content"  
    android:textSize="12sp" android:text="@string/hello_world" />
```

Android: XML and Element Attributes

View Attributes

Every view has many different attributes which can be applied to manage various properties. Certain properties are shared across many views such as android:layout_width. Other properties are based on a view's function such as android:textColor.

Attribute	Description	Example Value
android:background	Background for the view	#ffffff
android:onClick	Method to invoke when clicked	onButtonClicked
android:visibility	Controls how view appears	invisible
android:hint	Hint text to display when empty	@string/hint
android:text	Text to display in view	@string/foo
android:textColor	Color of the text	#000000
android:textSize	Size of the text	21sp
android:textStyle	Style of the text formatting	bold

XML Attribute for All Elements

android:accessibilityHeading	Whether or not this view is a heading for accessibility purposes.
android:accessibilityLiveRegion	Indicates to accessibility services whether the user should be notified when this view changes.
android:accessibilityPaneTitle	The title this view should present to accessibility as a pane title.
android:accessibilityTraversalAfter	Sets the id of a view after which this one is visited in accessibility traversal.
android:accessibilityTraversalBefore	Sets the id of a view before which this one is visited in accessibility traversal.
android:allowClickWhenDisabled	Whether or not allow clicks on disabled view.
android:alpha	alpha property of the view, as a value between 0 (completely transparent) and 1 (completely opaque).
android:autoHandwritingEnabled	Whether or not the auto handwriting initiation is enabled in this View.
android:autofillHints	Describes the content of a view so that a autofill service can fill in the appropriate data.
android:autofilledHighlight	Drawable to be drawn over the view to mark it as autofilled May be a reference to another resource, in the form "@[+][package:]type/name" or a theme attribute in the form "?[package:]type/name".

XML Attribute for All Elements

<u>android:background</u>	A drawable to use as the background.
<u>android:backgroundTint</u>	Tint to apply to the background.
<u>android:backgroundTintMode</u>	Blending mode used to apply the background tint.
<u>android:clickable</u>	Defines whether this view reacts to click events.
<u>android:clipToOutline</u>	Whether the View's Outline should be used to clip the contents of the View.
<u>android:contentDescription</u>	Defines text that briefly describes content of the view.
<u>android:contextClickable</u>	Defines whether this view reacts to context click events.
<u>android:defaultFocusHighlightEnabled</u>	Whether this View should use a default focus highlight when it gets focused but doesn't have <u>R.attr.state_focused</u> defined in its background.
<u>android:drawingCacheQuality</u>	Defines the quality of translucent drawing caches.
<u>android:duplicateParentState</u>	When this attribute is set to true, the view gets its drawable state (focused, pressed, etc.) from its direct parent rather than from itself.
<u>android:elevation</u>	base z depth of the view.
<u>android:fadeScrollbars</u>	Defines whether to fade out scrollbars when they are not in use.
<u>android:fadingEdgeLength</u>	Defines the length of the fading edges.
<u>android:filterTouchesWhenObscured</u>	Specifies whether to filter touches when the view's window is obscured by another visible window.

XML Attribute for All Elements

android:fitsSystemWindows	Boolean internal attribute to adjust view layout based on system windows such as the status bar.
android:focusable	Controls whether a view can take focus.
android:focusableInTouchMode	Boolean that controls whether a view can take focus while in touch mode.
android:focusedByDefault	Whether this view is a default-focus view.
android:forceHasOverlappingRendering	Whether this view has elements that may overlap when drawn.
android:foreground	Defines the drawable to draw over the content.
android:foregroundGravity	Defines the gravity to apply to the foreground drawable.
android:foregroundTint	Tint to apply to the foreground.
android:foregroundTintMode	Blending mode used to apply the foreground tint.
android:hapticFeedbackEnabled	Boolean that controls whether a view should have haptic feedback enabled for events such as long presses.
android:id	Supply an identifier name for this view, to later retrieve it with View.findViewById() or Activity.findViewById() .
android:importantForAccessibility	Describes whether or not this view is important for accessibility.
android:importantForAutofill	Hints the Android System whether the view node associated with this View should be included in a view structure used for autofill purposes.
android:importantForContentCapture	Hints the Android System whether the view node associated with this View should be use for content capture purposes.
android:isScrollContainer	Set this if the view will serve as a scrolling container, meaning that it can be resized to shrink its overall window so that there will be space for an input method.
android:keepScreenOn	Controls whether the view's window should keep the screen on while visible.

XML Attribute for All Elements

android:keyboardNavigationCluster	Whether this view is a root of a keyboard navigation cluster.
android:layerType	Specifies the type of layer backing this view.
android:layoutDirection	Defines the direction of layout drawing.
android:longClickable	Defines whether this view reacts to long click events.
android:minHeight	Defines the minimum height of the view.
android:minWidth	Defines the minimum width of the view.
android:nextClusterForward	Defines the next keyboard navigation cluster.
android:nextFocusDown	Defines the next view to give focus to when the next focus is View.FOCUS_DOWN If the reference refers to a view that does not exist or is part of a hierarchy that is invisible, a RuntimeException will result when the reference is accessed.
android:nextFocusForward	Defines the next view to give focus to when the next focus is View.FOCUS_FORWARD If the reference refers to a view that does not exist or is part of a hierarchy that is invisible, a RuntimeException will result when the reference is accessed.
android:nextFocusLeft	Defines the next view to give focus to when the next focus is View.FOCUS_LEFT .
android:nextFocusRight	Defines the next view to give focus to when the next focus is View.FOCUS_RIGHT If the reference refers to a view that does not exist or is part of a hierarchy that is invisible, a RuntimeException will result when the reference is accessed.
android:nextFocusUp	Defines the next view to give focus to when the next focus is View.FOCUS_UP If the reference refers to a view that does not exist or is part of a hierarchy that is invisible, a RuntimeException will result when the reference is accessed.
android:onClick	Name of the method in this View's context to invoke when the view is clicked.
android:outlineAmbientShadowColor	Sets the color of the ambient shadow that is drawn when the view has a positive Z or elevation value.

XML Attribute for All Elements

android:outlineSpotShadowColor	Sets the color of the spot shadow that is drawn when the view has a positive Z or elevation value.
android:padding	Sets the padding, in pixels, of all four edges.
android:paddingBottom	Sets the padding, in pixels, of the bottom edge; see R.attr.padding .
android:paddingEnd	Sets the padding, in pixels, of the end edge; see R.attr.padding .
android:paddingHorizontal	Sets the padding, in pixels, of the left and right edges; see R.attr.padding .
android:paddingLeft	Sets the padding, in pixels, of the left edge; see R.attr.padding .
android:paddingRight	Sets the padding, in pixels, of the right edge; see R.attr.padding .
android:paddingStart	Sets the padding, in pixels, of the start edge; see R.attr.padding .
android:paddingTop	Sets the padding, in pixels, of the top edge; see R.attr.padding .
android:paddingVertical	Sets the padding, in pixels, of the top and bottom edges; see R.attr.padding .
android:preferKeepClear	Sets a preference to keep the bounds of this view clear from floating windows above this view's window.
android:requiresFadingEdge	Defines which edges should be faded on scrolling.
android:rotation	rotation of the view, in degrees.
android:rotationX	rotation of the view around the x axis, in degrees.
android:rotationY	rotation of the view around the y axis, in degrees.
android:saveEnabled	If false, no state will be saved for this view when it is being frozen.
android:scaleX	scale of the view in the x direction.
android:scaleY	scale of the view in the y direction.
android:screenReaderFocusable	Whether this view should be treated as a focusable unit by screen reader accessibility tools.

XML Attribute for All Elements

android:scrollIndicators	Defines which scroll indicators should be displayed when the view can be scrolled.
android:scrollX	The initial horizontal scroll offset, in pixels.
android:scrollY	The initial vertical scroll offset, in pixels.
android:scrollbarAlwaysDrawHorizontalTrack	Defines whether the horizontal scrollbar track should always be drawn.
android:scrollbarAlwaysDrawVerticalTrack	Defines whether the vertical scrollbar track should always be drawn.
android:scrollbarDefaultDelayBeforeFade	Defines the delay in milliseconds that a scrollbar waits before fade out.
android:scrollbarFadeDuration	Defines the delay in milliseconds that a scrollbar takes to fade out.
android:scrollbarSize	Sets the width of vertical scrollbars and height of horizontal scrollbars.
android:scrollbarStyle	Controls the scrollbar style and position.
android:scrollbarThumbHorizontal	Defines the horizontal scrollbar thumb drawable.
android:scrollbarThumbVertical	Defines the vertical scrollbar thumb drawable.
android:scrollbarTrackHorizontal	Defines the horizontal scrollbar track drawable.
android:scrollbarTrackVertical	Defines the vertical scrollbar track drawable.
android:scrollbars	Defines which scrollbars should be displayed on scrolling or not.
android:soundEffectsEnabled	Boolean that controls whether a view should have sound effects enabled for events such as clicking and touching.
android:stateListAnimator	Sets the state-based animator for the View.
android:tag	Supply a tag for this view containing a String, to be retrieved later with View.getTag() or searched for with View.findViewWithTag() .
android:textAlignment	Defines the alignment of the text.
android:textDirection	Defines the direction of the text.

XML Attribute for All Elements

<u>android:theme</u>	Specifies a theme override for a view.
<u>android:tooltipText</u>	Defines text displayed in a small popup window on hover or long press.
<u>android:transformPivotX</u>	x location of the pivot point around which the view will rotate and scale.
<u>android:transformPivotY</u>	y location of the pivot point around which the view will rotate and scale.
<u>android:transitionName</u>	Names a View such that it can be identified for Transitions.
<u>android:translationX</u>	translation in x of the view.
<u>android:translationY</u>	translation in y of the view.
<u>android:translationZ</u>	translation in z of the view.
<u>android:visibility</u>	Controls the initial visibility of the view.

Android: Finding Elements using different Locator Strategies

- Understanding how to properly use Locators is key to building your automation scripts.
- After all, if you're unable to “find” the UI element, you cannot control it (such as clicking a button).
- In Mobile (or Web) Automation Testing automating any scenario follows these 2 steps:
 - 1) Find the UI element locators (uniquely).
 - 2) Perform an action on that element

Android: Finding Elements using different Locator Strategies

- **What is an Element Locator?**

- An Element Locator is nothing but an address that identifies a UI Element on a Mobile App (or Website).
- As there are many UI elements present on a single mobile application screen there can be a chance that same (generic)address can refer to more than one element.
- This means that we need to find a unique address for the element.
- As you will see, sometimes this is easy, and other times you have to do some further exploration to uniquely identify your UI element.

Android: Finding Elements using different Locator Strategies

● What is an Element Locator?

- The way in which you uniquely identify the element is called a locator strategy. Appium makes many different strategies available.
- As you may expect, there are many different locator strategies available to you, including
 - 1) Accessibility ID
 - 2) Class name
 - 3) ID
 - 4) Name
 - 5) XPath
 - 6) Image (Recently Introduced)
 - 7) Android UiAutomator (UiAutomator2 only)
 - 8) Android View Tag (Espresso only)

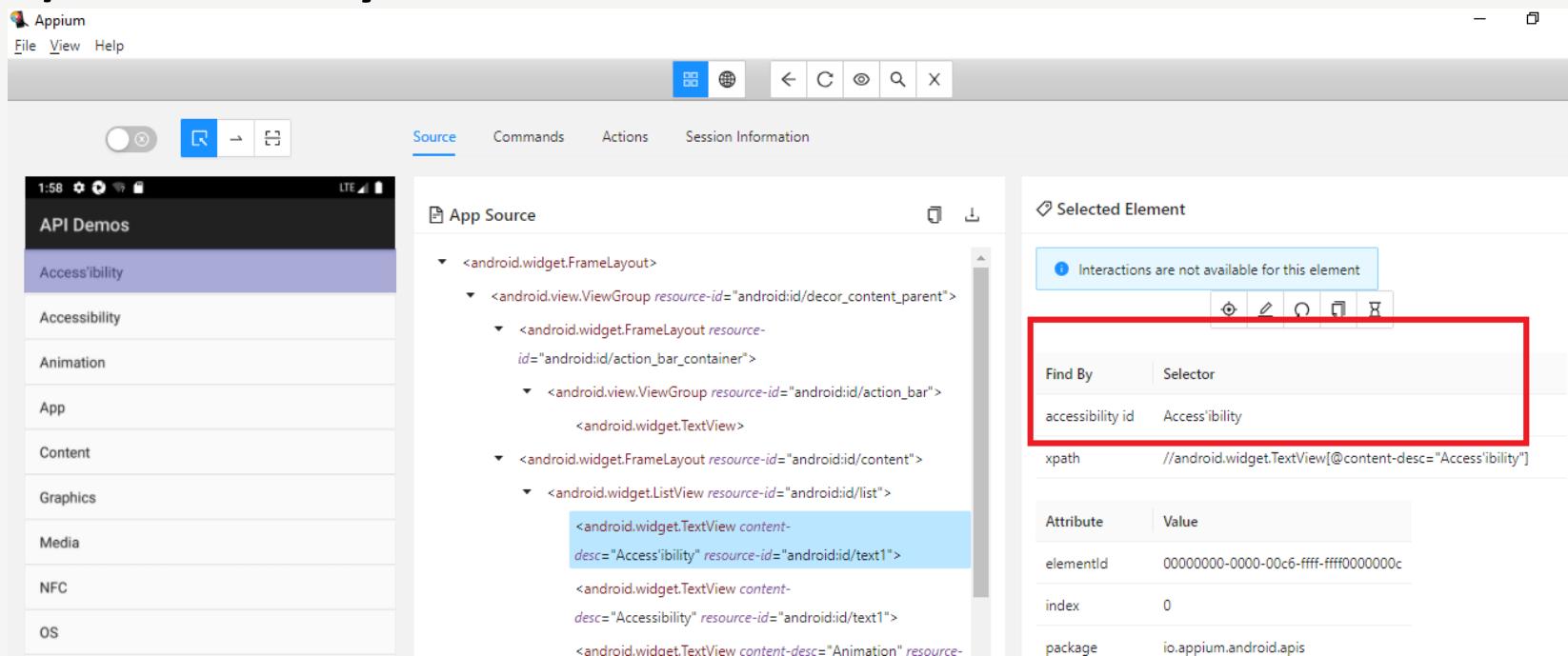
Android: Finding Elements using different Locator Strategies

1) Accessibility ID

- This is the best preferred locator strategy in Appium. Always use this one if you can.
- It is a Cross-platform locator strategy as this works in a similar way on iOS and Android which makes your code more reusable.
- **iOS:** If the accessibility id property(attribute) value is set at development time (by the app developers) then you can inspect it using the Appium Inspector(Android & iOS) or UiAutomatorViewer (Android).
- When Accessibility Id property value is not defined by developer, it is by default equals to the Name of that UI Element.
- **Android:** Accessibility Id property is equals to content-desc property(attribute) on Android.

Android: Finding Elements using different Locator Strategies

1) Accessibility ID



The screenshot shows the Appium Inspector interface. On the left, there's a mobile device screenshot of the "API Demos" app with the "Accessibility" item selected in a purple bar. The main area shows the XML structure of the app's UI. On the right, the "Selected Element" panel displays the selected element's details. A red box highlights the "Find By" dropdown in the "Selector" section, which is currently set to "accessibility id". Other options like "xpath", "attribute", and "elementId" are also visible.

```
driver.findElementByAccessibilityId("accessibility").click();
```

[Click here for program](#)

Android: Finding Elements using different Locator Strategies

2) Class Name

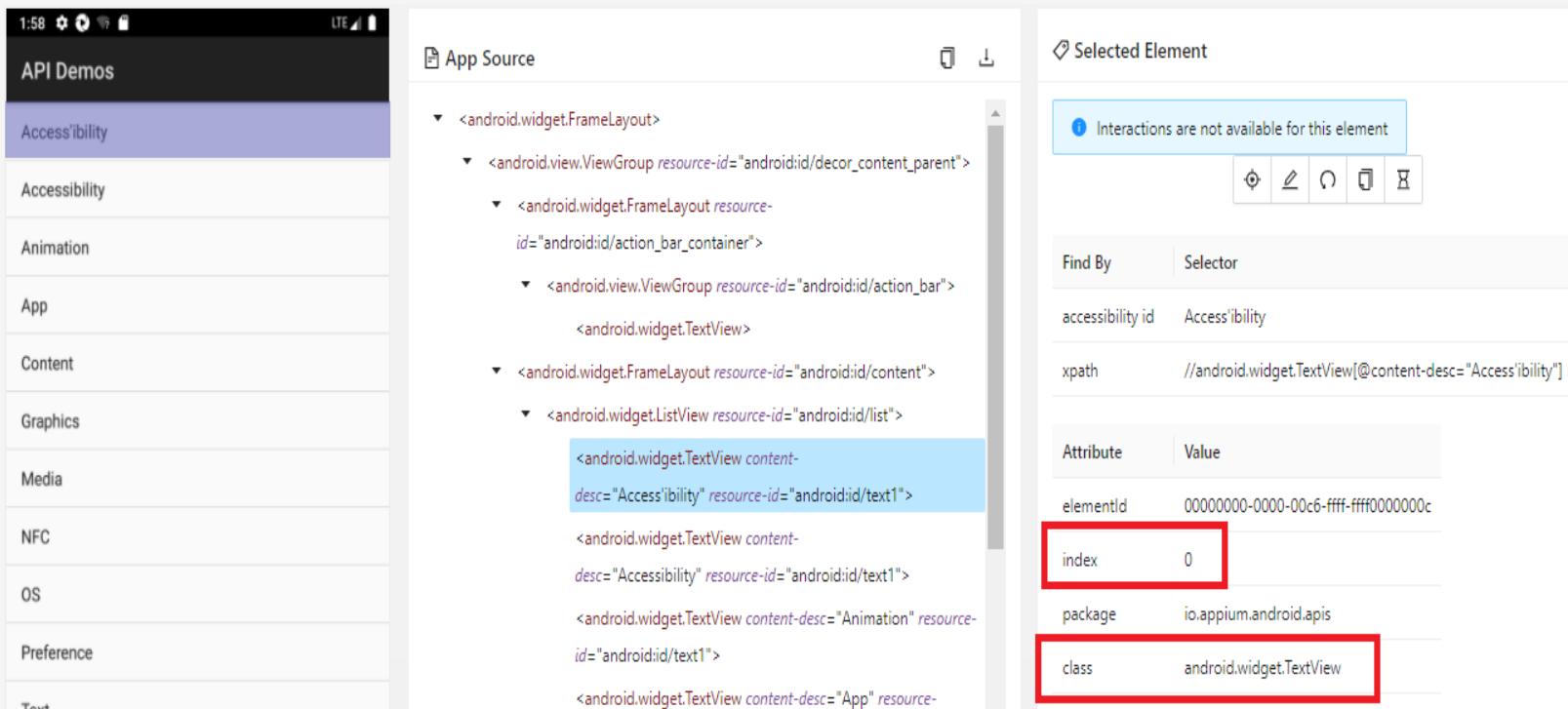
- Finding an element using Class Name is generic and it does not guarantee to find the unique element because many elements have the same class name.
- **iOS:** In iOS the class name is the fully qualified name of UIAutomation class, and it starts with “UIA” keyword such as UIAButton, UIARadioButton and UIATextField for old versions of iPhone Apps, and on recent versions made on Swift programming language you can find the “XCUITest” keyword.
- **Android:** In Android, the class name is the fully qualified name of the UIAutomator class and these are examples of it: android.widget.TextView , android.widget.Button, android.widget.ImageButton, android.widget.CheckBox etc.

Java :

```
WebElement element = driver.findElement(By.className("android.widget.TextView"));  
// OR  
WebElement element = driver.findElementByClassName("android.widget.TextView");
```

Android: Finding Elements using different Locator Strategies

2) Class Name



The screenshot shows the Appium Inspector interface. On the left, there's a mobile application preview titled "API Demos" with a navigation menu. The "Accessibility" item is highlighted in purple. The main pane displays the XML structure of the application's UI components. A specific `<android.widget.TextView>` element is selected, highlighted with a blue background. In the "Selected Element" panel on the right, the class of the element is listed as "android.widget.TextView". The "Find By" section contains an XPath query: `//android.widget.TextView[@content-desc='Accessibility']`. Below it, there are sections for "Attribute" and "Value" with entries for "elementId" and "index". The "index" field is highlighted with a red border.

Attribute	Value
elementId	0000000-0000-00c6-ffff-0000000c
index	0
package	io.appium.android.apis
class	android.widget.TextView

[Click Here for Program](#)

Android: Finding Elements using different Locator Strategies

3) ID :

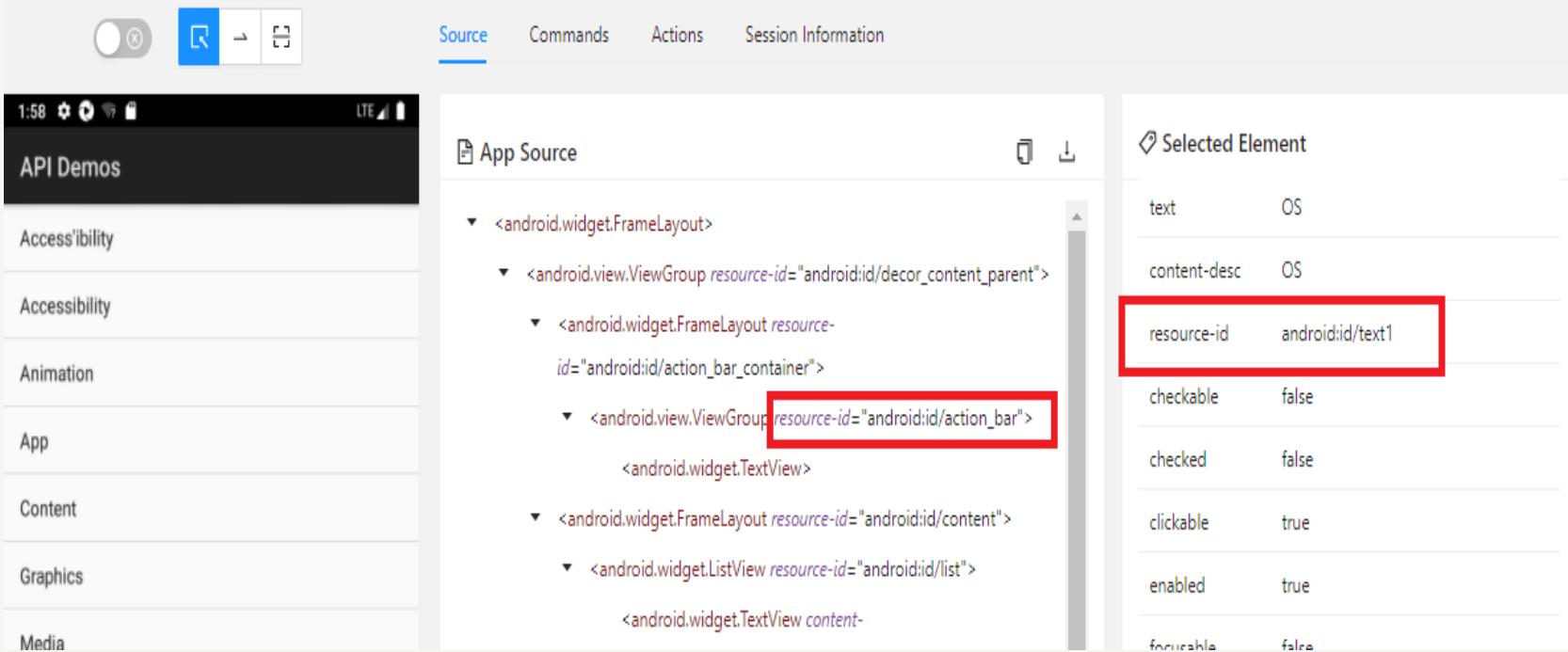
- Finding an element using Class Name is
- In Mobile Application Automation id is are in form of Native context, it is not similar to Selenium WebDriver's CSS id.
- id are also cross-platform locator strategy similar like accessibility id.
- **iOS:** It will find elements by name and label attribute but before that Appium will try to search for a accessibility id that will match with the given id string
- **Android:** In Android, it's resource-id attribute. It contains common :id/ string format.
- You can use either that full string (ex. io.selendroid.testapp:id/startUserRegistration) or only (startUserRegistration). So in the below code both options are valid

Java:

```
driver.findElement(By.id("android:id/text1")).click();
```

Android: Finding Elements using different Locator Strategies

3) ID :



The screenshot shows the Android Device Monitor interface. The top navigation bar includes tabs for Source, Commands, Actions, and Session Information. The Source tab is active, displaying the XML structure of the 'API Demos' application. A red box highlights the line of code for the action bar's resource ID: <android.widget.ViewGroup resource-id="android:id/action_bar">. To the right, a 'Selected Element' table provides details about this specific element:

text	OS
content-desc	OS
resource-id	android:id/text1
checkable	false
checked	false
clickable	true
enabled	true
focuseable	false

[Click here for program](#)

Android: Finding Elements using different Locator Strategies

4) Name

- **iOS & Android:** It's the Name of the element on both platforms. This isn't used as often as accessibility id and id strategies are mostly used

Java :

```
driver.findElement(By.name("android:name/text1")).click();
```

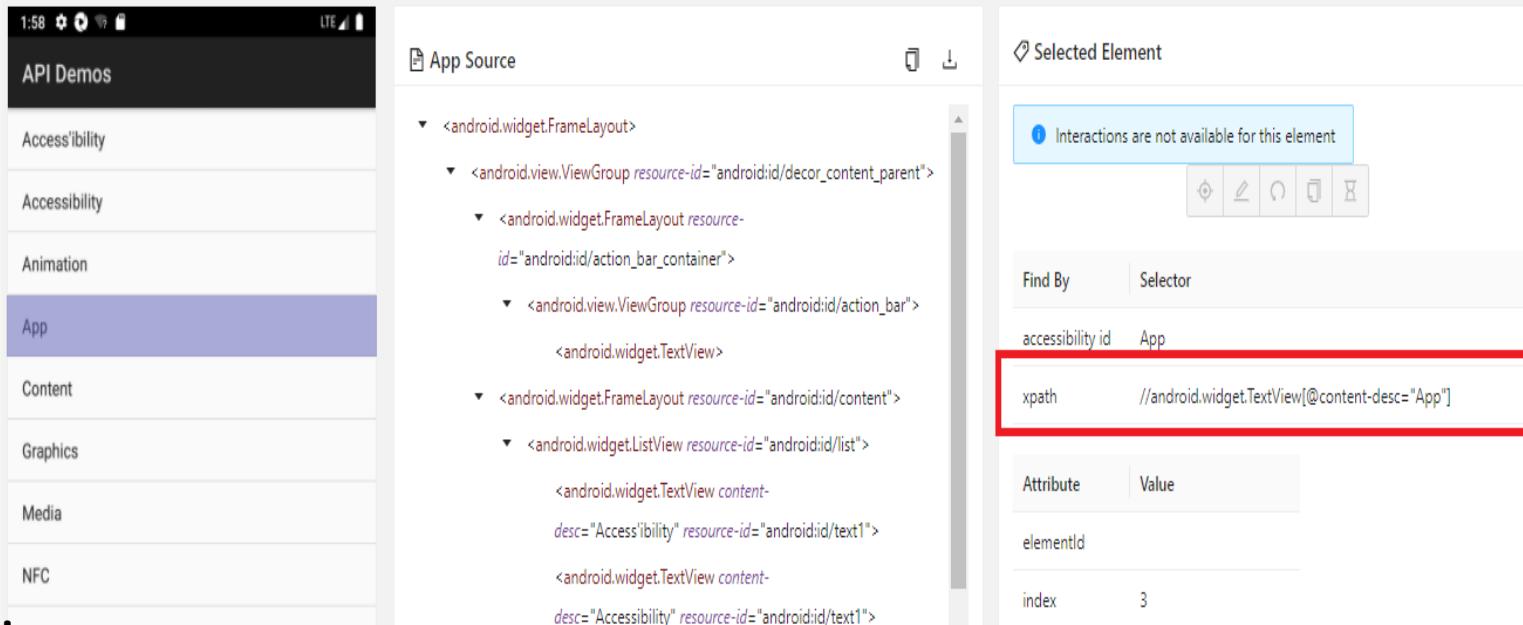
Android: Finding Elements using different Locator Strategies

5) XPath

- This locator strategy analyzes the XML structure of the app and locates the element with respect to the other elements.
- The XPath is originally designed to allow for the navigation of XML data with the purpose of locating unique UI elements.
- XPath selectors are not cross-platform.
- This strategy should only be used when there is no Accessibility Id, Id or Name assigned to an UI Element. XPath has performance and stability issues but is very “brittle” changing across platforms and even device manufacturers.
- This strategy comes to the rescue when you’ve tried the above strategies and failed. As it depends on Parent XML nodes it’s really very fragile because when any new UI element gets added or removed, the XML structure is changed rendering your locators broken.
- If you are using the Appium Inspector for inspection of the Application XML structure then Appium will give you the XPath directly without any extra effort.

Android: Finding Elements using different Locator Strategies

5) XPath



The screenshot shows the Android Studio XML Editor. On the left, the 'App Source' tree view displays the structure of an XML file, with the 'App' category highlighted. On the right, the 'Selected Element' panel shows the selected element and its properties. The XPath query `//android.widget.TextView[@content-desc='App']` is highlighted with a red border in the 'Find By' section.

Java:

```
driver.findElement(By.xpath("//android.widget.TextView[@content-desc=\"App\"]")).click();
```

[Click here for program](#)

Android: Finding Elements using different Locator Strategies

6) Image

- Appium supports Image Comparison as a locator strategy which is using the OpenCV library in the backend.
- The strings which are being used by this locator strategy are Base64- encoded image files.
- So you need to convert image files into Base64-encoded image files first and you need to pass that String into the locator.

Java:

```
String base64Image = //Code which will to convert Image file to Base-64 String  
WebElement element = driver.findElementByImage(base64Image);
```

Android: Finding Elements using different Locator Strategies

7) Android UiAutomator (UiAutomator2 only)

- Appium supports Image Comparison as a locator strategy which is using the OpenCV
- This is an Android Platform specific locator strategy.
- This is rarely used to find the element locators as it requires to have prior knowledge of the UiSelector API (and of course it's Android only).
- It's performance is slightly better than XPath.
- In this example we find the first Button element having the text Login:

Java:

```
String selector = "new UiSelector().text(\"Cancel\")  
.className(\"android.widget.Button\")); MobileElementelement=(MobileElement)  
driver.findElement(MobileBy.AndroidUIAutomator(selector));
```

Android: Finding Elements using different Locator Strategies

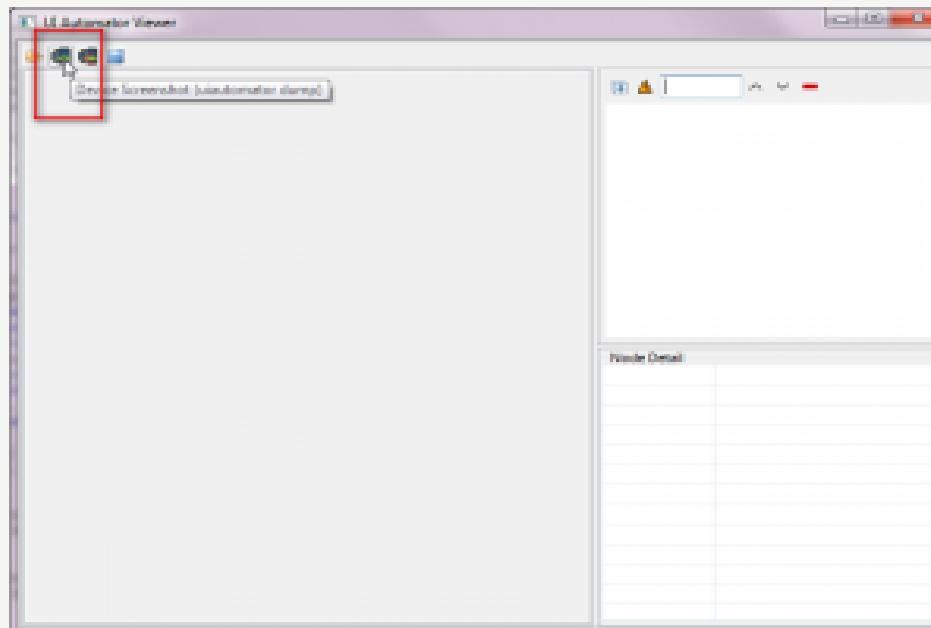
8) Android View Tag (Espresso only)

- This is also an Android Platform specific locator strategy.
- It locates an element using it's view tag

Finding Elements using UiAutomator (Native Techniques)

How to inspect using UIAutomator?

- Enable the debugging mode in our computer and connect to the android device.
- Then, Open the UI automator tool by running the ‘uiautomatorviewer.bat’ from the SDK folder location.
- Open the Calculator app in the Phone and take the screenshot of the app using UI automator.



Finding Elements using UiAutomator (Native Techniques)

- Enable the debugging mode in our computer and connect to the android device.
- After the screenshot is captured, the UI automator will show the calculator app's UI on the left side.
- Right side of the tool have two parts, top parts displays the node structure of the app's UI elements. Bottom Part displays property detail of the selected elements.



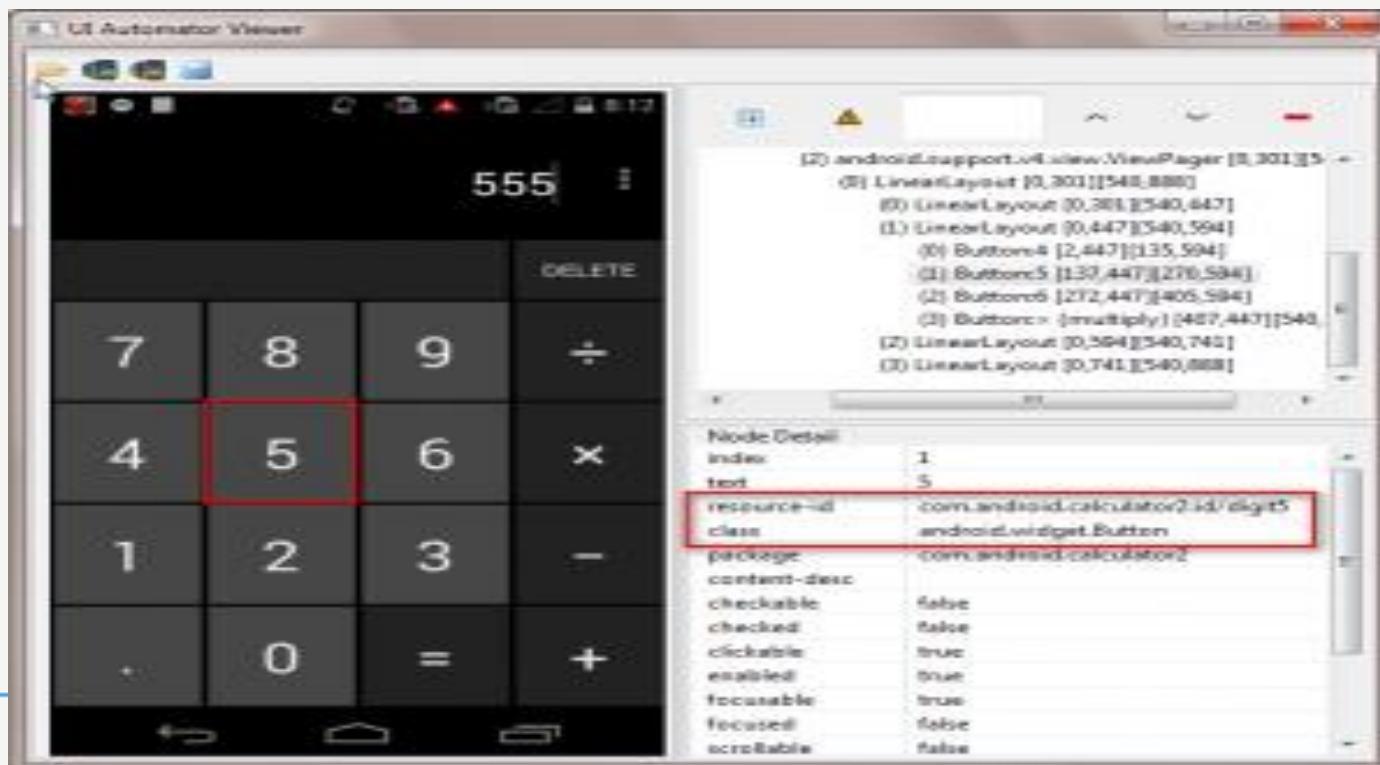
Finding Elements using UiAutomator (Native Techniques)

- Now Let me brief some ways to locate android app elements. Consider button '5' in app. See the different ways to write Xpath.
- Xpath using class and text attribute.



Finding Elements using UiAutomator (Native Techniques)

- X path can be created using text attribute value with class name.
- *XPath = xpath("//android.widget.Button[@text='5']")*
- XPath using Class and Resource ID.

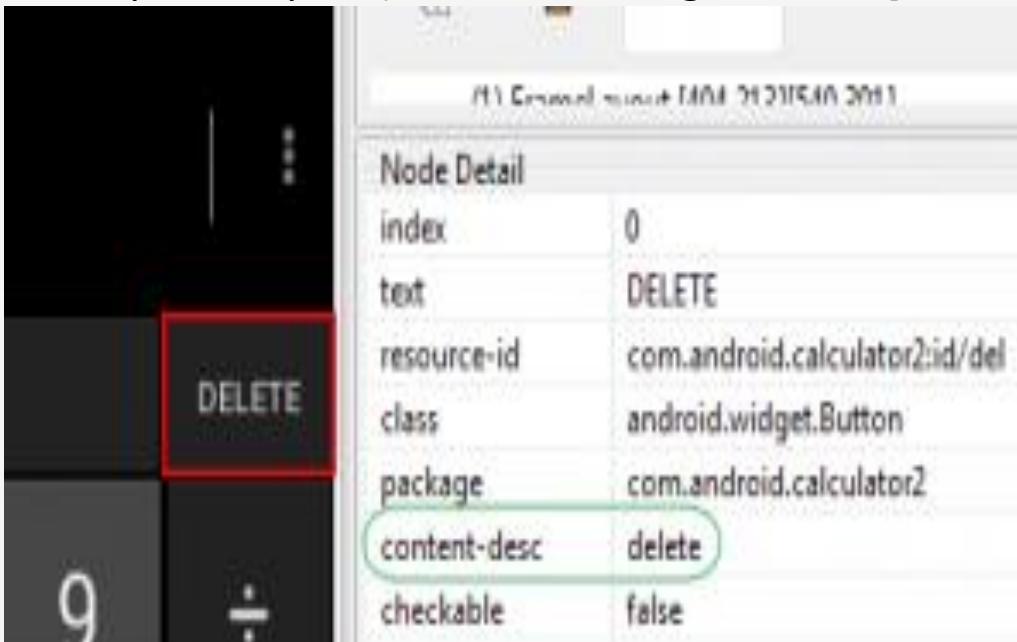


Finding Elements using UiAutomator (Native Techniques)

- Here we use contains function to get the Xpath like shown below.
- *Xpath = xpath("//android.widget.Button[contains(@resource-id,'digit5')]")*
- Xpath using Class, Text attribute and Resource ID.
- *XPath = xpath("//android.widget.Button[contains(@resource-id,'digit5') and @text='5']")*
- Xpath using class, text attribute and index.
- *Xpath = xpath("//android.widget.Button[@text='5' and @index='1']")*
- XPath using parent and Child hierarchy.
- Here, Parent class android.widget.LinearLayout class with index = 1 has buttons 4, 5, 6 and X. So we can locate that specific row by it's class index. Child element is member of class android.widget.Button with index = 1. So we can format XPath using parent and child class hierarchy as bellow.
- *Xpath =
xpath("//android.widget.LinearLayout[@index='1']/android.widget.Button[@index='1']")*
- Xpath using content-desc.

Finding Elements using UiAutomator (Native Techniques)

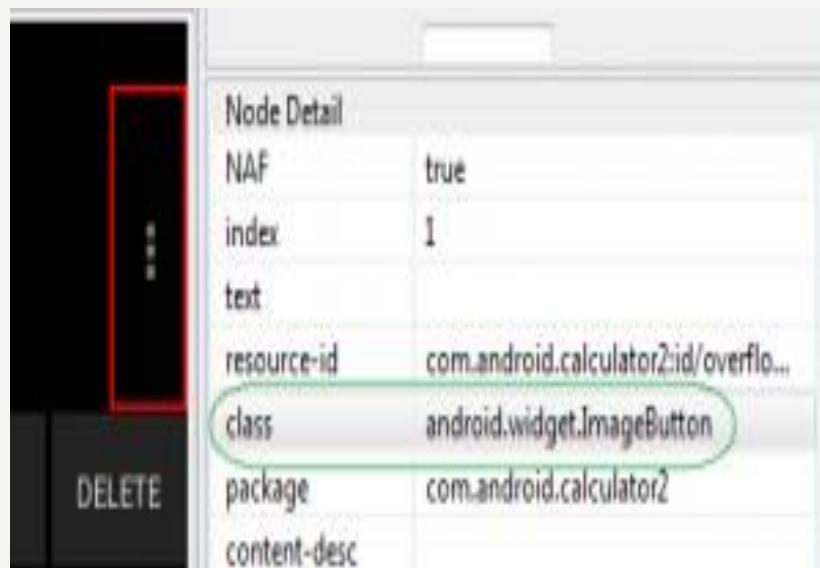
- Content description is unique for a android app element. So this can also be used to create xpath.
- Here, delete button have unique content-desc = delete.
- Xpath = xpath("//android.widget.Button[@content-desc='delete'])")*



[Click here for program](#)

Finding Elements using UiAutomator (Native Techniques)

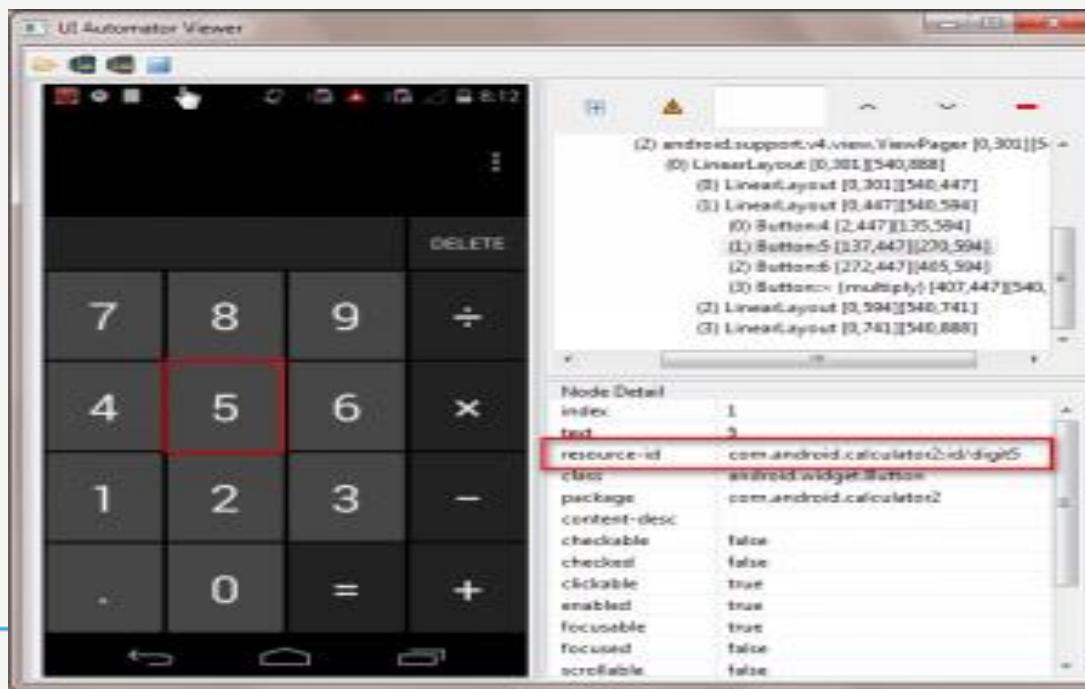
- Xpath using class name.
- If class name is unique, we can use this method to find the xpath.
- Xpath = `xpath("//android.widget.ImageButton")`



[Click here for program](#)

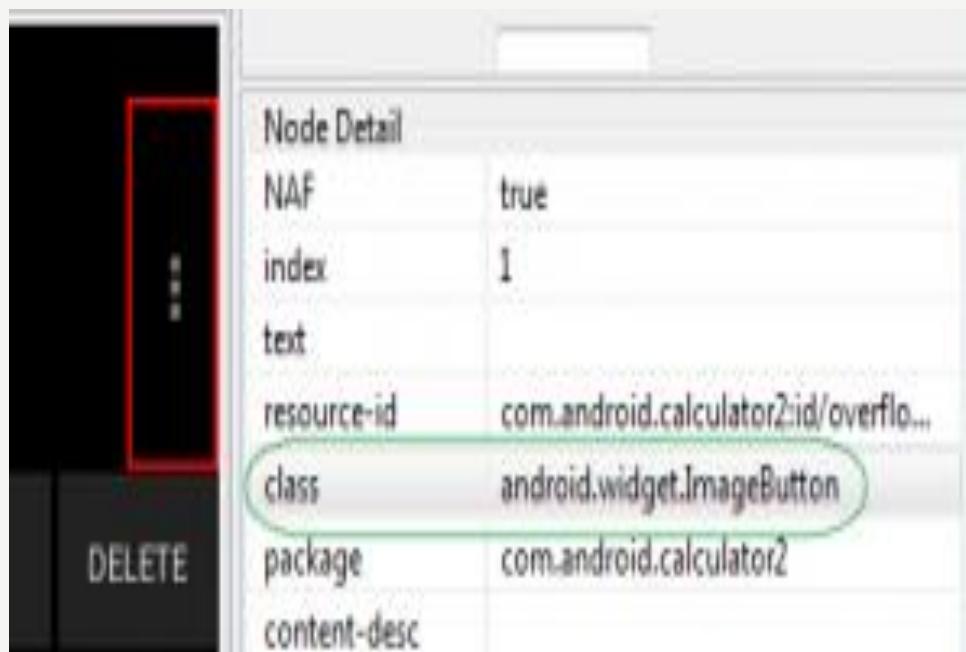
Finding Elements using UiAutomator (Native Techniques)

- Locating element by ID
- Resource id is used for this type of locator.
- Here, resource id of button '5' is "com.android.calculator2:id/digit5".
- Xpath = id("com.android.calculator2:id/digit5")



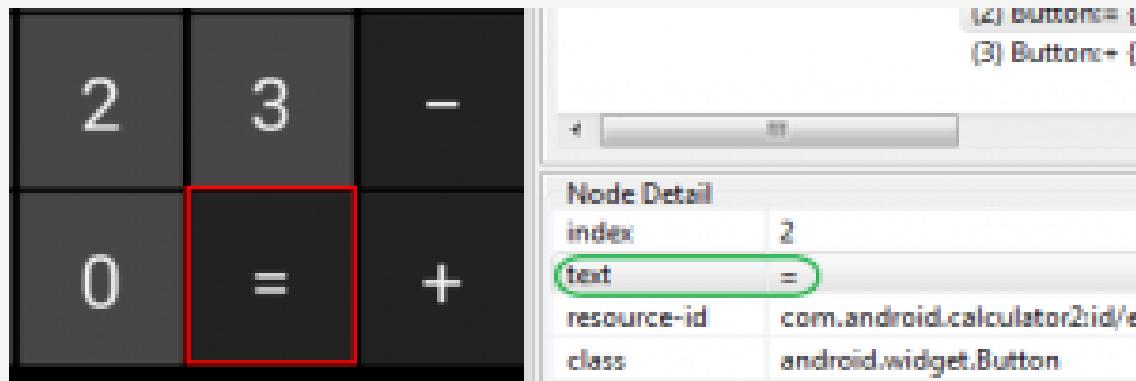
Finding Elements using UiAutomator (Native Techniques)

- Locating by class name.
- If class name is unique, we can use it to locate elements.
- Here element can located by “`By.className("android.widget.ImageButton")`”.



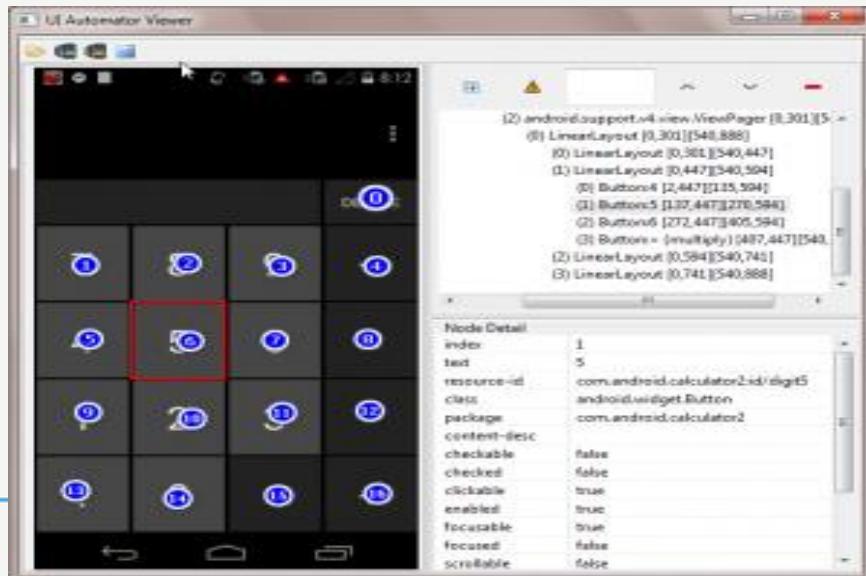
Finding Elements using UiAutomator (Native Techniques)

- Locating android elements by Name.
- If the element contains unique text, it is possible to locate by its name.
- Here '=' can be located by "By.name("=")".



Finding Elements using UiAutomator (Native Techniques)

- Locating android elements by Name.
- Locating elements by find elements.
- For calculator app, we can see that all buttons have the same class “android.widget.Button”.
- Using ‘find elements’ we can get list of all buttons and store the list using java List interface. Then, we can get any element using get() method of list interface.



```
List<WebElement> calcButtons =  
driver.findElements(By.xpath("//  
android.widget.Button"));
```

The above syntax will store the button list in the list ‘calcButtons’.

Finding Elements using UiAutomator (Native Techniques)

Button	Array index ID
Delete	0
7	1
8	2
9	3
÷	4
4	5
5	6
6	7
x	8
1	9
2	10
3	11
-	12
.	13
0	14
=	15
+	16

Finding Elements using UiAutomator (Native Techniques)

How to use touch actions in Appium

- Up until now we have looked into basic Appium automation, such as finding and clicking on a button or typing text into a text field.
- However, “real world” mobile applications are more sophisticated and contain many complex UI elements that require user interactions such as double tap, long press, swipe left/right, pull up/down and even multi-touch actions.

Finding Elements using UiAutomator (Native Techniques)

Appium supports the following gestures:

- Tap on an element.
- Tap on x, y coordinates.
- Press an element for a particular duration.
- Press x, y coordinates for a particular duration.
- Horizontal swipe: Using start and end percentage of the screen height and width.
- Vertical swipe: Using start and end percentage of the screen height and width.
- Drag(Swipe) one element to another element.
- Multitouch for an element.
- Appium supports these gestures using the **Touch Actions** class.

```
TouchAction touchAction = new TouchAction(driver);
```

Finding Elements using UiAutomator (Native Techniques)

Appium supports the following gestures:

METHOD NAME	PURPOSE
press(PointOption pressOptions)	Press action on the screen.
longPress(LongPressOptions longPressOptions)	Press and hold the at the center of an element until the context menu event has fired.
tap(PointOption tapOptions)	Tap on a position.
moveTo(PointOption moveToOptions)	Moves current touch to a new position.
cancel()	Moves current touch to a new position.
perform()	Perform this chain of actions on the performsTouchActions.

Finding Elements using UiAutomator (Native Techniques)

Official Appium API docs: [Click Here..!](#)

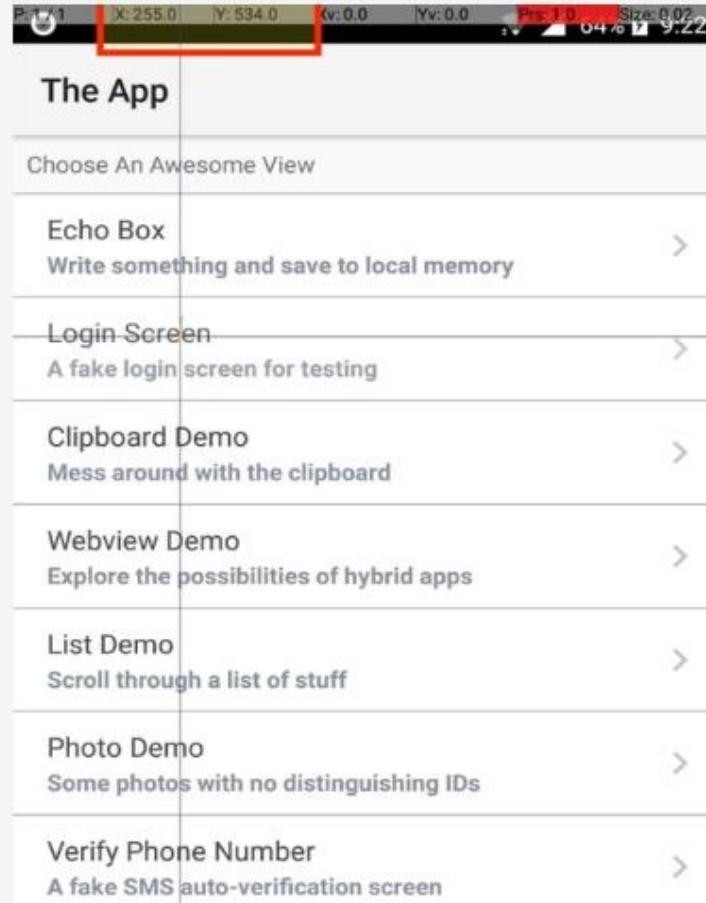
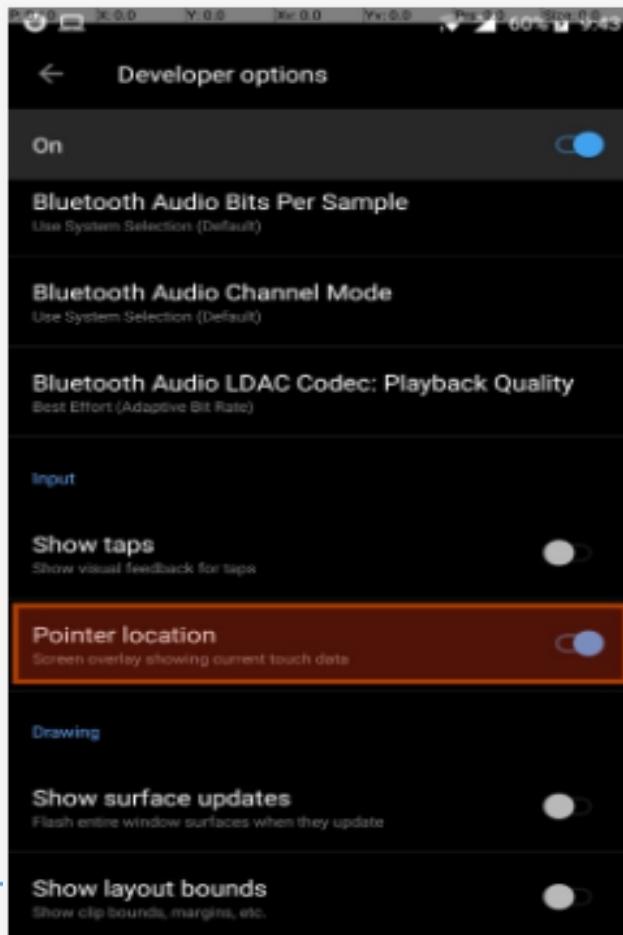
- Before exploring each mentioned action we need to understand the significance of perform() as it plays a vital role.
- The Appium client simply records all the instructions and actions on the client side and stores the intermediate values in a local data structure.
- The perform() method is used to send all actions to the appium server – as soon as perform() is called, the intermediate actions and instructions are converted to JSON and sent to the appium server, and then actual action is being performed.
- So for any gesture code the last method called would be perform

Finding Elements using UiAutomator (Native Techniques)

Now the question is how can you get the x,y coordinate?

- It depends...
- Because you can get the Pointer location in Android but you can not get it in iOS devices.
- Getting the pointer location in Android:
 1. Move to Settings > Developer options
 2. Enable the Pointer location.
 3. Now move to any application for which you need the coordinates of a particular location. Tap on the location and you will get the coordinates for that place at top of the screen.

Finding Elements using UiAutomator (Native Techniques)



Finding Elements using UiAutomator (Native Techniques)

Getting the pointer location in iOS:

- iOS does not support the pointer location and there aren't even any third party apps or tools which come to the rescue.
- Therefore you need to calculate it using screen resolution and a little bit of prediction.
- In case you don't get success at first, you can use trial and error to get the needed location.

Finding Elements using UiAutomator (Native Techniques)

Now let's look into each gesture one by one:

1) Tap on element:

Method: tap(TapOptions tapOptions)

Usage: It is the simplest action, as the name suggests it will simply click/tap on a particular location. It is a combination of press() and release()

```
TouchAction touchAction = new TouchAction(driver);
touchAction.tap(tapOptions()
    .withElement(element(androidElement)))
    .perform()
```

NOTE: Here you can also put the wait along with the tap action, for example:

```
new TouchAction(driver)
    .tap(tapOptions().withElement(element(androidElement)))
    .waitAction(waitOptions(Duration.ofMillis(millis)))
    .perform();
```

Finding Elements using UiAutomator (Native Techniques)

Now let's look into each gesture one by one:

2) Tap on x, y coordinates:

Method: tap(PointOption pointOptions)

Usage: It is used to tap on a particular x,y coordinate point.

```
TouchAction touchAction = new TouchAction(driver);
touchAction.tap(PointOption.point(1280, 1013))
    .perform();
```

NOTE: Similar like Tap on element you can put the wait along with the tap action, for example:

```
new TouchAction(driver)
    .tap(tapOptions().withElement(element(androidElement)))
    .waitAction(waitOptions(Duration.ofMillis(millis)))
    .perform();
```

Finding Elements using UiAutomator (Native Techniques)

Now let's look into each gesture one by one:

3) Press an element for a particular duration.

Method: press(PointOption pressOptions)

Usage: It is used to apply the press action. After the press action you also need to release so that the state would be in press mode. You do so by calling the release() function after calling press().

```
TouchAction touchAction = new TouchAction(driver);
touchAction.press(element(element))
    .waitAction(waitOptions(ofSeconds(seconds)))
    .release()
    .perform();
```

Finding Elements using UiAutomator (Native Techniques)

Now let's look into each gesture one by one:

4) Press x, y coordinates for a particular duration.

Method: press(PointOption pressOptions)

Usage: Similar to Press an element for a particular duration, here you just need to pass x, y coordinates instead of an element and don't forget to call the release() function after calling press().

```
TouchAction touchAction = new TouchAction(driver);
touchAction.press(point(x,y))
    .waitAction(waitOptions(ofSeconds(seconds)))
    .release()
    .perform();
```

Automating Swipe Actions in Appium

- Before we look into the Horizontal swipe let's understand how we can automate swipe actions generally.
- Swiping is a combination of tapping + moving actions.
- Appium does not provide a direct method for swiping, so you need to combine a few methods in order to achieve swiping.
- For example if you want to perform swiping then first you have to press on a particular point and then specify the particular amount of time during which you want to perform the swiping action and at last you to move to another point – and don't forget to call the release method which used to release all the actions.
- So it's actually simple: first press -> wait(duration of swiping) -> move to (moveTo()) particular location.
- You might be thinking why can't we directly use moveTo() ?

Automating Swipe Actions in Appium



- If you recall, using the press method requires you to eventually call the release method. So we are basically mimicking a swipe by entering the press state, moving to a location, and THEN releasing.
- Swiping can have an up/down/right/left direction so you need to apply the right logic and have to provide the x, y coordinates for the press() and moveTo() methods.
- And also please note these appium methods to get the device screen measurements:

```
int heightOfScreen =
driver.manage().window().getSize().getHeight();

int widthOfScreen =
driver.manage().window().getSize().getWidth();

int middleHeightOfScreen = heightOfScreen/2;
// To get 50% of width

int x = widthOfScreen * 0.5;
// To get 50% of height

int y = heightOfScreen * 0.5;

Here Width → X and Height → Y coordinates.
```

Automating Swipe Actions in Appium

Now let's look into each gesture one by one:

5) Horizontal swipe: Using start and end percentage of the screen height and width.

Method and Usage: As we discussed above there is no particular method for Horizontal swipe, and you need to perform the combination of `press()->wait()->moveTo()`.

The `moveTo()` method is new to us – it is used to move to particular location. Its syntax is: `moveTo(PointOption pressOptions)`

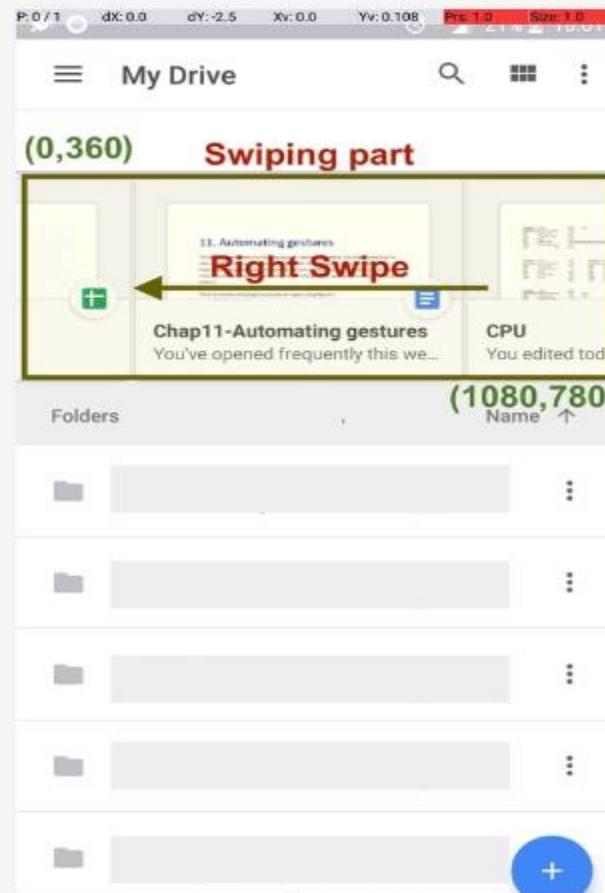
The secret to `moveTo` lies in the coordinates – you need to mention the starting and ending x,y coordinates in such a way that swiping can done from left → right OR right → left direction. Note that when we say swipe right, we mean moving the content from right to left, but the physical gesture is moving to the left.

Automating Swipe Actions in Appium

5) Horizontal swipe: Using start and end percentage of the screen height and width.

This is just one scenario for achieving a swipe gesture. Ideally first we need to make a decision as to what is the “right side of the screen”.

We do this by considering 90% of the screen width. For example, if the screen resolution is 1920 x 1080, 1080 is the width of Screen and 90% of that width would equals to 972, so we have got our X coordinate for what we consider the “Right” side.



Automating Swipe Actions in Appium

- In a similar manner we will need the X coordinate for the Left side and this time we can consider 10% of the width which would give us an X coordinate of about 108.
- So we have got X coordinates for Left and Right direction. For the Y coordinate we can choose any value as long as it falls in the swiping area – for example, let's say our swiping area is between (0, 360) to (1080, 780), so you can choose any value for the Y coordinate in between 360 to 780.

```
TouchAction swipe = new TouchAction(driver)
    .press(PointOption.point(972,500))
    .waitAction(waitOptions(ofMillis(800)))
    .moveTo(PointOption.point(108,500))
    .release() .perform();
```

Automating Swipe Actions in Appium

6) Vertical swipe(scroll): Using start and end percentage of the screen height and width.

Usage: Scroll is the same as swipe but the direction is different. In swiping we are dealing with horizontal direction where as in scrolling we are dealing with vertical direction – but the rest of the logic will remain the same. Scrolling can done in the up → down OR down → up direction.

1) Find the swiping area.

Starting point = (0,360)

Ending point = (1080,1920)

2) Mark the scrolling points (We will use the height from scrolling area only. As per below image, the scrolling area is starting from approximately 30% of the screen height and ending at the end of the screen).

Automating Swipe Actions in Appium

6) Vertical swipe(scroll): Using start and end percentage of the screen height and width.

Down area point:

i) X = Middle of Screen= $0.5 \times 1080 = 540$ (This will be same for starting and ending location)

ii) Y = 95% height of Screen = $0.95 \times 1920 = 1824$.

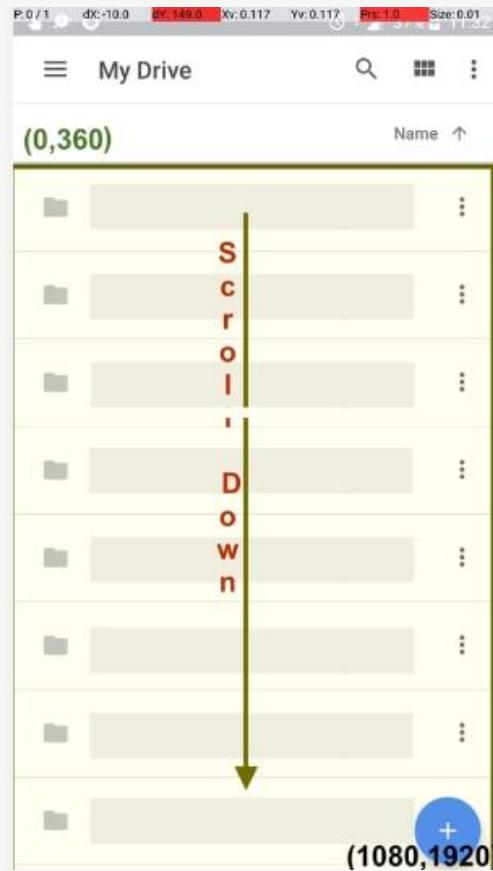
Location = (540,1824)

3) Perform scroll action using Appium.

```
TouchAction swipe = new TouchAction(driver)
    .press(PointOption.point(540,1824))
    .waitAction(waitOptions(ofMillis(800)))
    .moveTo(PointOption.point(540,672))
    .release()
    .perform();
```

Automating Swipe Actions in Appium

6) Vertical swipe(scroll): Using start and end percentage of the screen height and width.



Automating Swipe Actions in Appium

7) Drag(Swipe) one element to an another element.

Dragging one element to another element is one kind of swiping action.

But here location in coordinates would not matter as we have both of the elements(1. Element which needs to be dragged, 2. Element upon which another element will be dragged).

```
TouchAction swipe = new TouchAction(driver)
    .press(ElementOption.element(element1))
    .waitAction(waitOptions(ofSeconds(2)))
    .moveTo(ElementOption.element(element2))
    .release()
    .perform();
```

Automating Swipe Actions in Appium

8) MultiTouch.

As the name suggests it means multiple touches happening at the same time.

For example on iOS if you want to move to the Main screen, you need to use 5 fingers and do a swipe.

Multi Touch is handled by MultiTouchAction class.

It has a add(TouchActions touchActions) method so in which we need to pass a TouchActions object. So let say you want to press on 5 different points at a time then first you need to create 5 TouchActions, but here the important thing is we are not having a perform method at the end.

We just need to call the release method for the TouchAction object, and then pass those values into the add method of the MultiTouchAction class.

Automating Swipe Actions in Appium

You can perform Multi Touch for:

Multiple touches at a time.

```
TouchAction touchActionOne = new TouchAction();  
touchActionOne.press(PointOption.point(100, 100));  
touchActionOne.release();  
  
TouchAction touchActionTwo = new TouchAction();  
touchActionTwo.press(PointOption.point(200, 200));  
touchActionTwo.release();  
  
MultiTouchAction action = new MultiTouchAction();  
action.add(touchActionOne);  
action.add(touchActionTwo);  
action.perform();
```

Swiping using multiple fingers

Getting the pointer location in iOS:

- **NOTE:** As mentioned earlier only `MultiTouchAction` should call the `perform()` method at the end.
- For `TouchActions` `perform()` method should not be called otherwise instructions will be sent to Appium server and the click will happen before the Multi Touch action.
- In this chapter we have looked into the most used scenarios in the Appium world.
- These methods all work on both Android and iOS. More details about all the different `TouchAction` methods can be found on the official appium docs.
https://appium.github.io/java-client/io/appium/java_client/TouchAction.html

Synchronization using Waits

- If you recall, we touched on the wait method and its significance and we promised to get back to it.
- So in this chapter we will understand how wait (or Synchronization) performs a vital role in Automation.
- If two or more components are working together in parallel at the same pace or rate, synchronization comes into play.
- We see it in almost every application whenever the screen changes it takes a few milliseconds (or seconds) to load, and if you do not manage the proper synchronization in your code then you might face the dreaded “ElementNotFoundException” or “NoSuchElementException” exceptions.
- This is because the screen hasn't finished loading and is not synchronized with your test code.

Synchronization using Waits

- That is, your test code is over eager and starts trying to perform an action on an element that hasn't been loaded yet.
- To avoid this we need to implement proper synchronization in our automation script.
- We can categorize synchronization in two types:
 - 1) Unconditional,
 - 2) Conditional.

Synchronization using Waits

Unconditional synchronization :

- Unconditional Synchronization is also known as Static Synchronization or Static Wait.
- As the name suggests, it specifies a particular fixed (static) time to wait before starting the execution.
- Here Appium(or any program) will wait the specified amount of time and then it will resume the execution.
- The standard example of Unconditional Synchronization is below:

```
try
{
    Thread.sleep(1000);
} catch (InterruptedException e)
{
    e.printStackTrace();
}
```

Synchronization using Waits

Unconditional synchronization :

- Here the Thread.sleep(1000) function would take 1000 ms to execute.
- Key to note is that this wait will be absolute, even if the underlying condition you were waiting on has been met.
- For example, you may put in a wait for 3 seconds waiting for the screen to load. Even if that screen loads in 2 seconds, the system will still wait for the additional second.
- The converse is also true - Sometimes the wait finishes before the underlying operation and execution proceeds.

Synchronization using Waits

Conditional synchronization :

- Conditional synchronization depends on some underlying condition.
- So in addition to a specified absolute time to wait, the condition is also passed into the method.
- Here the script(or program) will resume execution as soon as the condition is met - or, in the event the condition isn't met, it will resume after the specified time.
- Appium (or Selenium) provides 3 (mainly 2) types of conditional synchronization.
 - 1) Implicit wait**
 - 2) Explicit wait**
 - 3) Fluent wait**

Synchronization using Waits

1) Implicit wait:

- Implicit wait tells the Appium's webdriver object to poll the DOM for the specified amount of time while trying to find the element before throwing an "ElementNotVisibleException" or "NoSuchElementException" exceptions.
- The big advantage of using Implicit wait is it's lifespan. As we apply Implicit wait on the Webdriver object, it will be valid for the webdriver object's lifespan.
- Below is the code to apply the Implicit wait of 10 seconds on the Webdriver object.

```
AppiumDriver driver = new AppiumDriver(new URL(APPIUM_SERVER_URL),  
capabilities);
```

```
// Set Implicit wait upon AppiumDriver(WebDriver)  
driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
```

Also you can use other Time Units such as

TimeUnit.NANOSECONDS | TimeUnit.MICROSECONDS | TimeUnit.MILLISECONDS

Please remember that Implicit wait works with only driver.findElement(...) and driver.findElements(...) methods – it won't work for other methods

Synchronization using Waits

2) Explicit wait:

- Explicit waits are the best synchronization methods for dynamic responses in the application.
- Explicit wait informs the AppiumDriver(WebDriver) to wait
 - 1) Until the specified condition is met OR
 - 2) The specified time has elapse
...before throwing the “ElementNotVisibleException” or “NoSuchElementException” exceptions.
And if the AppiumDriver is able to meet the condition within the specified amount of time then the code will get executed.
- In explicit wait we need to tell the WebDriver object to wait for a specific condition using the ExpectedConditions class.
- So, actually this wait is specific to a particular single element rather than the whole WebDriver object (unlike implicit wait).
- The WebDriverWait class will call the ExpectedCondition every 500 milliseconds by default until the output is True.

[Click here for program](#)

Synchronization using Waits

2) Explicit wait:

- Explicit waits are the best synchronization methods for dynamic responses in the application.
- So if you have given 10 seconds of timeout, ExpectedCondition would be called 20 times at 500 milliseconds intervals to check if the condition has been met.

```
WebDriverWait webDriverWait = new WebDriverWait(driver, 30);
webDriverWait.until(ExpectedConditions.visibilityOfElementLocated(By.id
("menubutton")));
```

- When we initialize new object of WebDriverWait class, we need to pass 2 parameters: 1) WebDriver object.

2) Number of seconds

After creation of the WebDriver object you need to call the until() method and need to pass the ExpectedConditions.() inside it.

There are many conditions are defined in ExpectedConditions class, but we can list down a few popular ones

Condition Name	Purpose
<code>elementToBeClickable(By locator)</code> Example: <code>ExpectedConditions.elementToBeClickable(By.id("loginButton"));</code>	An expectation for checking an element is visible and enabled such that you can click it. In this method you need to pass the object of By class.
<code>elementToBeClickable(WebElement element)</code> Example: <code>ExpectedConditions.elementToBeClickable(driver.findElement(By.id("menubutton")));</code>	An expectation for checking an element is visible and enabled such that you can click it. In this method you need to pass the object of WebElement class.
<code>presenceOfElementLocated(By locator)</code>	An expectation for checking that an element is present on the DOM of a page. This does not necessarily mean that the element is visible.
<code>visibilityOfElementLocated(By locator)</code>	An expectation for checking that an element is either invisible or not present on the DOM.
<code>elementToBeSelected(WebElement element)</code>	An expectation for checking if the given element is selected.
<code>numberOfElementsToBe(By locator, java.lang.Integer number)</code>	An expectation for checking number of WebElements with given locator.
<code>titles(java.lang.String title)</code>	An expectation for checking the title of a page. <i>This is not applicable to Appium Mobile Application.</i>
<code>textToBePresentInElement(WebElement element, java.lang.String text)</code>	An expectation for checking if the given text is present in the specified element.

Synchronization using Waits

3) Fluent wait:

- Explicit waits are the best synchronization methods for dynamic responses in the application.
- Fluent wait is part of WebDriverWait, The only difference is it's more configurable than Explicit wait.
- You can configure the :

Poll frequency:

- The is the Time Interval to check whether the expected condition for the webelement is met or not. So if poll frequency is 1 second and total wait time is 10 seconds, fluent will check if the condition is met or not at every 1 second for a maximum of 10 times.
- **Ignore the Exception:** If you want to ignore a specific exception such as NoSuchElementExceptions while searching for an element.
- **Maximum wait time:** The total maximum amount of time to wait for a condition is met before throwing an exception.

Synchronization using Waits

3) Fluent wait:

Below is the example of Fluent wait:

```
FluentWait<AppiumDriver> webDriverWait = new  
FluentWait<AppiumDriver>(driver);  
  
webDriverWait.pollingEvery(Duration.ofSeconds(1));  
webDriverWait.ignoring(NoSuchElementException.class);  
webDriverWait.withTimeout(Duration.ofSeconds(10));
```

Which types of Wait you should use When?

Wait Type	Purpose
Implicit	When you need to apply common wait without any condition.
Explicit	When you need to test expected condition for an element.
Fluent	When you need to test expected condition for an element after a specific amount of time like every x seconds/minutes.

API Testing

Modules

- [Api Introduction](#)
- [Download and install postman](#)
- [Postman navigation](#)
- [Postman other concept](#)

Module - 1 [API Introduction]

Agenda

- Appium Introduction
- Role of A Software Tester in API Testing
- API Testing and Unit testing

API Introduction

What is API?

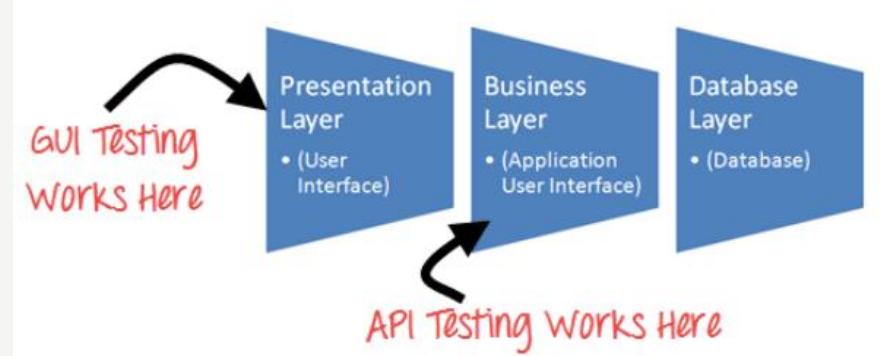
- API stands for ***Application Programming Interface***.
- Talking in technical terms an API is a set of procedures, functions, and other points of access that an application, an operating system, a library, etc., makes available to programmers in order to allow it to interact with other software.
- Didn't get it? Well, neither did I. Let's break these terms and explore more about APIs.

or

- **API TESTING** is a software testing type that validates Application Programming Interfaces (APIs).
- The purpose of API Testing is to check the functionality, reliability, performance, and security of the programming interfaces.
- In API Testing, instead of using standard user inputs(keyboard) and outputs, you use software to send calls to the API, get output, and note down the system's response

More...

- API tests are very different from GUI Tests and won't concentrate on the look and feel of an application.
- It mainly concentrates on the business logic layer of the software architecture.



- For background, **API (Application Programming Interface)** is a computing interface that enables communication and data exchange between two separate software systems.
- A software system that executes an API includes several functions/subroutines that another software system can perform.
- API defines requests that can be made, how to make requests, data formats that can be used, etc., between two software systems

API Testing Approach

- **API Testing Approach** is a predefined strategy or a method that the QA team will perform in order to conduct the API testing after the build is ready.
- This testing does not include the source code.
- The API testing approach helps to better understand the functionalities, testing techniques, input parameters and the execution of test cases.
- The following points helps the user to do an API Testing approach:
 - 1) Understanding the functionality of the API program and clearly defining the scope of the program.
 - 2) Apply testing techniques such as equivalence classes, boundary value analysis, and error guessing and write test cases for the API.
 - 3) Input Parameters for the API need to be planned and defined appropriately.
 - 4) Execute the test cases and compare expected and actual results.

How to Test API

- API automation testing should cover at least following testing methods apart from the usual SDLC process
 - **Discovery testing:** The test group should manually execute the set of calls documented in the API like verifying that a specific resource exposed by the API can be listed, created and deleted as appropriate
 - **Usability testing:** This testing verifies whether the API is functional and user-friendly. And does API integrates well with another platform as well
 - **Security testing:** This testing includes what type of authentication is required and whether sensitive data is encrypted over HTTP or both
 - **Automated testing:** API testing should culminate in the creation of a set of scripts or a tool that can be used to execute the API regularly
 - **Documentation:** The test team has to make sure that the documentation is adequate and provides enough information to interact with the API. Documentation should be a part of the final deliverable

Best Practices of API Testing:

- API Test cases should be grouped by test category
- On top of each test, you should include the declarations of the APIs being called.
- Parameters selection should be explicitly mentioned in the test case itself
- Prioritize API function calls so that it will be easy for testers to test
- Each test case should be as self-contained and independent from dependencies as possible
- Avoid “test chaining” in your development
- Special care must be taken while handling one-time call functions like – Delete, CloseWindow, etc...
- Call sequencing should be performed and well planned
- To ensure complete test coverage, create API test cases for all possible input combinations of the API.

Types of Bugs that API testing detects

- Fails to handle error conditions gracefully
- Unused flags
- Missing or duplicate functionality
- Reliability Issues. Difficulty in connecting and getting a response from API.
- Security Issues
- Multi-threading issues
- Performance Issues. API response time is very high.
- Improper errors/warning to a caller
- Incorrect handling of valid argument values
- Response Data is not structured correctly (JSON or XML)

Challenges of API Testing

- Challenges of API testing includes:
 - 1) The main challenges in Web API testing are **Parameter Combination, Parameter Selection, and Call Sequencing**
 - 2) There is no GUI available **to test the application, which makes it difficult to give input values**
 - 3) Validating and Verifying the output in a different system is a little difficult for testers
 - 4) Parameters selection and categorization are required to be known to the testers
 - 5) Exception handling function **needs to be tested**
 - 6) Coding knowledge is necessary for testers

Roles & Responsibilities of a Software tester for testing API's

- Should able to use all the web methods like GET, POST, DELETE, etc
- Validate the response, response time, error code
- Able to validate the XML and JSON body by using Json parsers
- Must know to use OAuth and OAuth2 authentication mechanisms
- Load and Security testing on web services
- Able to read and understand the API documentations
- Able to derive a good number of test cases and scenarios.
- Should be good in SQL queries to validate API and DB data elements
- Become master in a tool of your own choice SOAP UI and Postman are not Automation tools. Rest Assured, Rest Sharp, Node modules are the open source libraries for API testing.

Difference between Unit Testing vs API Testing

Unit testing	API testing
Developers perform it	Testers perform it
Separate functionality is tested	End-to-end functionality is tested
A developer can access the source code	Testers cannot access the source code
UI testing is also involved	Only API functions are tested
Only basic functionalities are tested	All functional issues are tested
Limited in scope	Broader in scope
Usually ran before check-in	Test run after the build is created

Module - 2 [Download and Install Postman]

Agenda

- Postman Introduction
- Why use Postman
- Download and install postman

Postman Introduction

What is Postman?

- **Postman** is a scalable API testing tool that quickly integrates into CI/CD pipeline.
- It started in 2012 as a side project by Abhinav Asthana to simplify API workflow in testing and development.
- API stands for Application Programming Interface which allows software applications to communicate with each other via API calls.

Why use Postman?

- With over 4 million users nowadays, Postman Software has become a tool of choice for the following reasons:
- **Accessibility** – To use Postman tool, one would just need to log-in to their own accounts making it easy to access files anytime, anywhere as long as a Postman application is installed on the computer.
- **Use of Collections** – Postman lets users create collections for their Postman API calls. Each collection can create subfolders and multiple requests. This helps in organizing your test suites.
- **Collaboration** – Collections and environments can be imported or exported making it easy to share files. A direct link can also be used to share collections.
- **Creating Environments** – Having multiple environments aids in less repetition of tests as one can use the same collection but for a different environment. This is where parameterization will take place which we will discuss in further lessons.

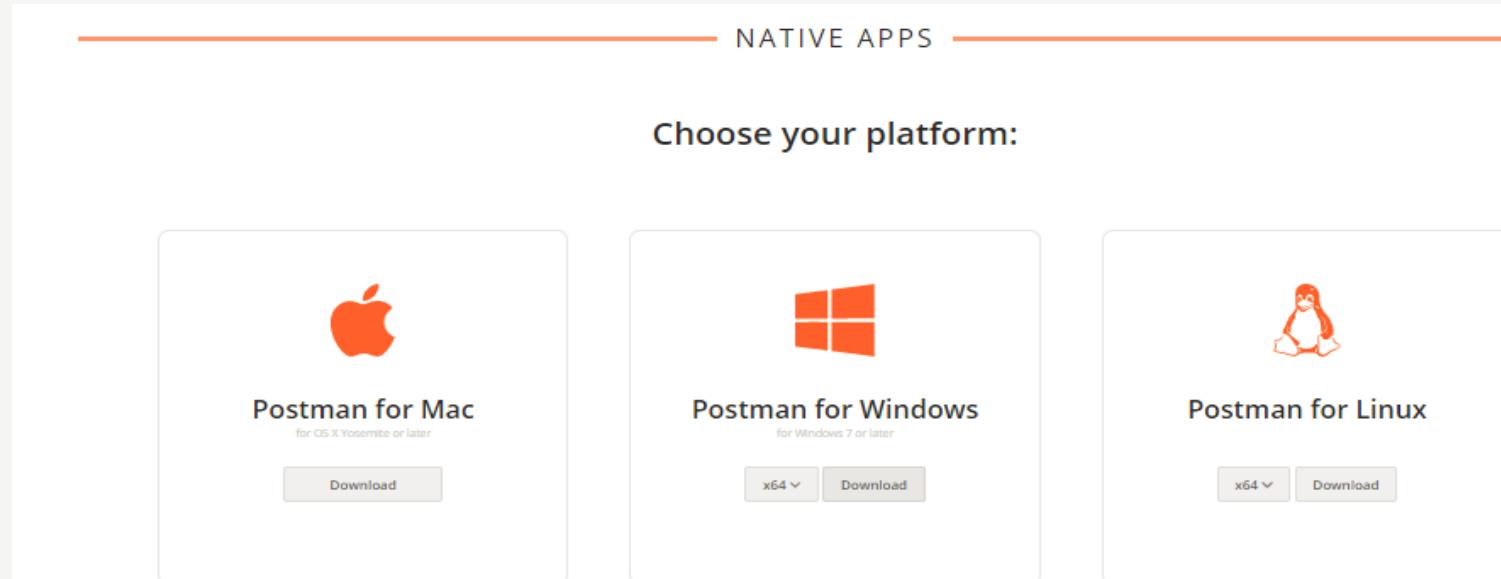
Creation of Tests – Test checkpoints such as verifying for successful HTTP response status can be added to each Postman API calls which help

Why use Postman?

- **Creation of Tests** – Test checkpoints such as verifying for successful HTTP response status can be added to each Postman API calls which help ensure test coverage.
- **Automation Testing** – Through the use of the Collection Runner or Newman, tests can be run in multiple iterations saving time for repetitive tests.
- **Debugging** – Postman console helps to check what data has been retrieved making it easy to debug tests.
- **Continuous Integration** – With its ability to support continuous integration, development practices are maintained.

Download Postman as a Standalone Application

- 1. Go to <https://www.getpostman.com/apps>



The screenshot shows the 'NATIVE APPS' section of the Postman download page. It features three cards for different operating systems:

- Postman for Mac**: For iOS X Yosemite or later. Includes a download button.
- Postman for Windows**: For Windows 7 or later. Includes a dropdown menu for 'x64' and a download button.
- Postman for Linux**: Includes a download button.

Below the cards, a note says: "If you want to be first in line to experience new features, download our latest [Canary builds](#)."

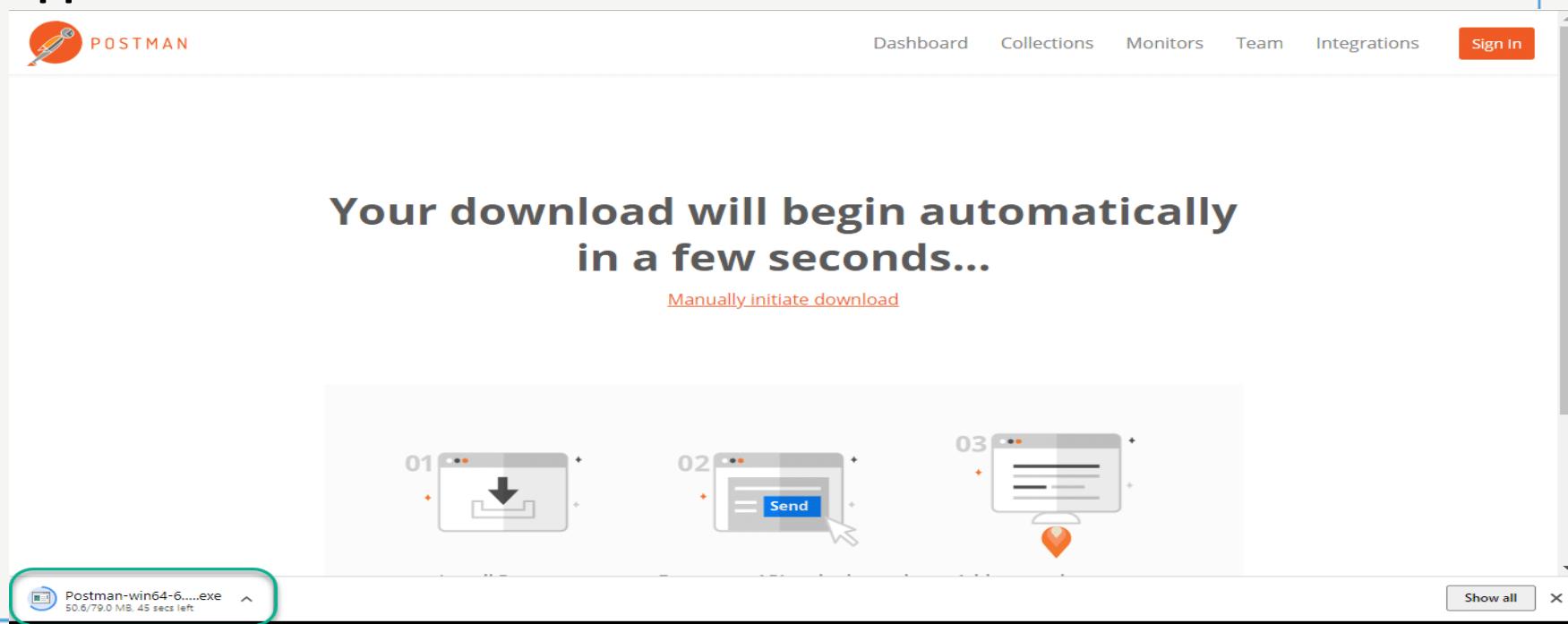
Download Postman as a Standalone Application

2. Choose the *Operating System* on which you want to download PostMan and click on "Download" button. Since I have got Windows 64-bit machine, I am going to install **x64-Windows**.



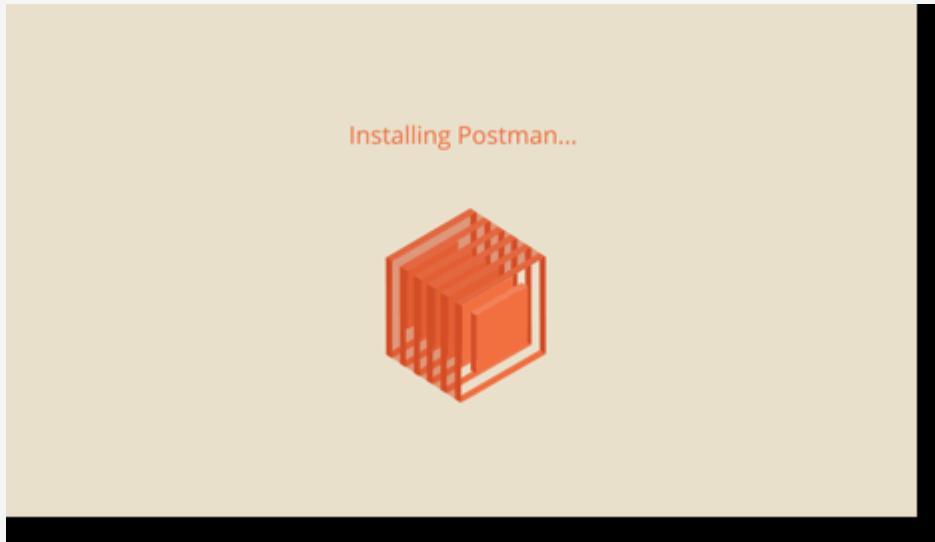
Download Postman as a Standalone Application

3. Once you download the .exe file, you will need to install the application. Since I am using the Chrome browser, the downloaded .exe will appear at the bottom left of the browser.



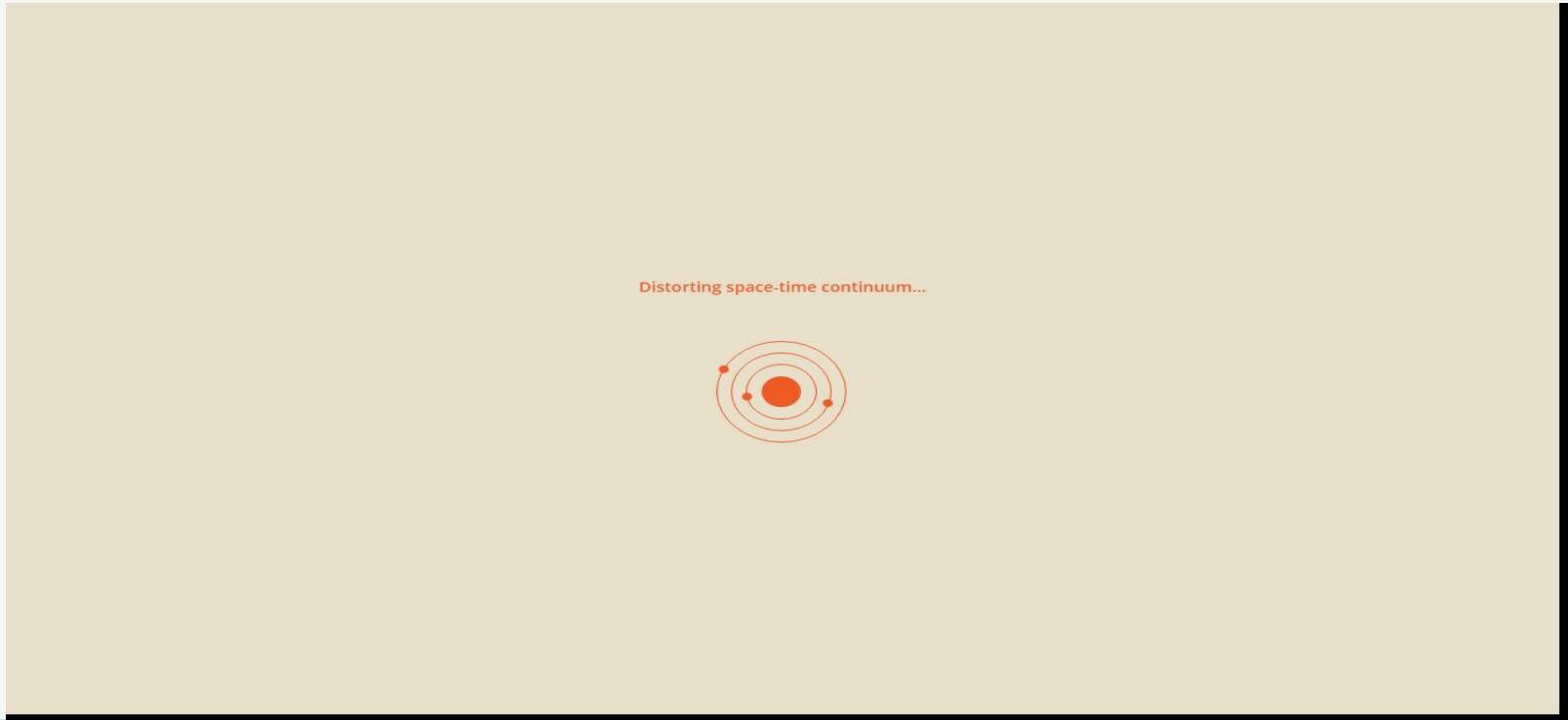
Download Postman as a Standalone Application

4. Click on the ***exe file*** to install it on the system. First, it will install the POSTMAN application.



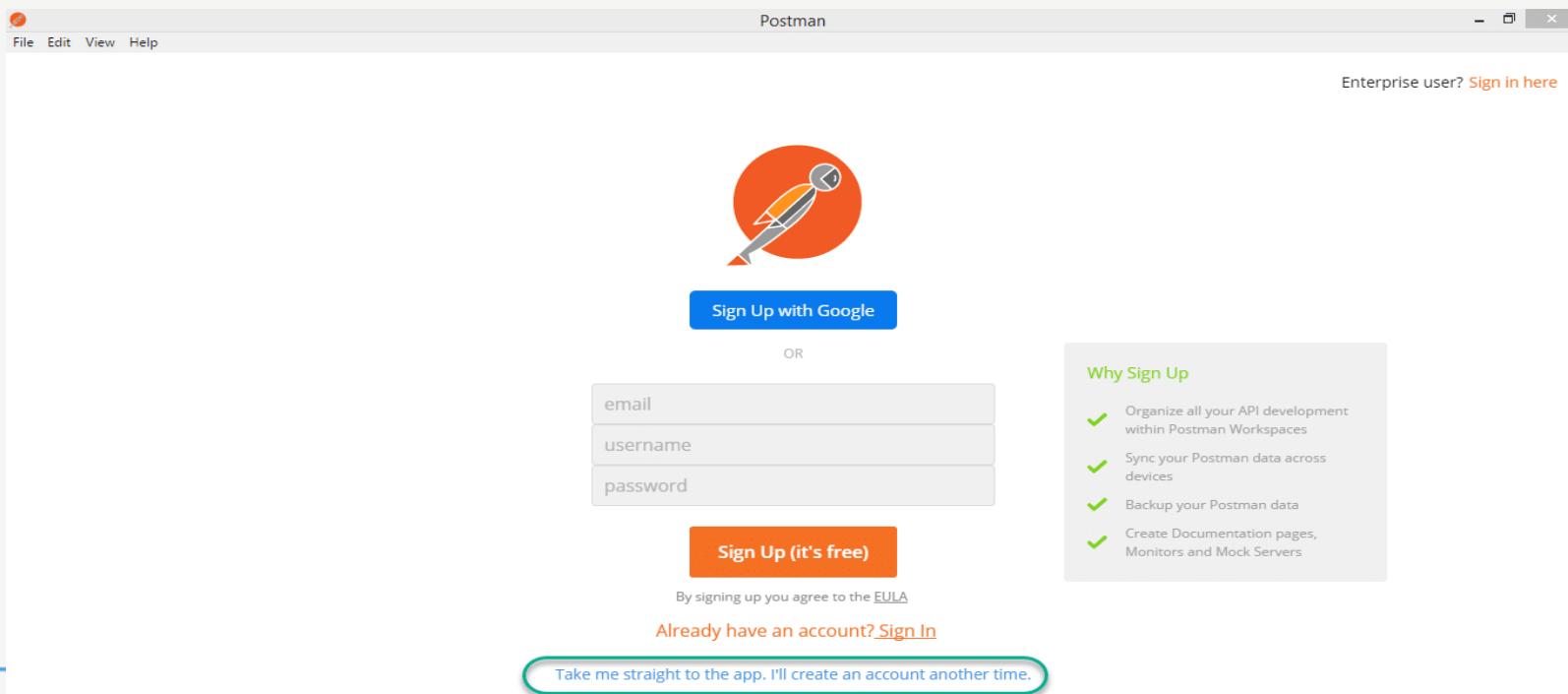
Download Postman as a Standalone Application

5. There are no further steps for installing. After completion, it will automatically start opening the PostMan tool.



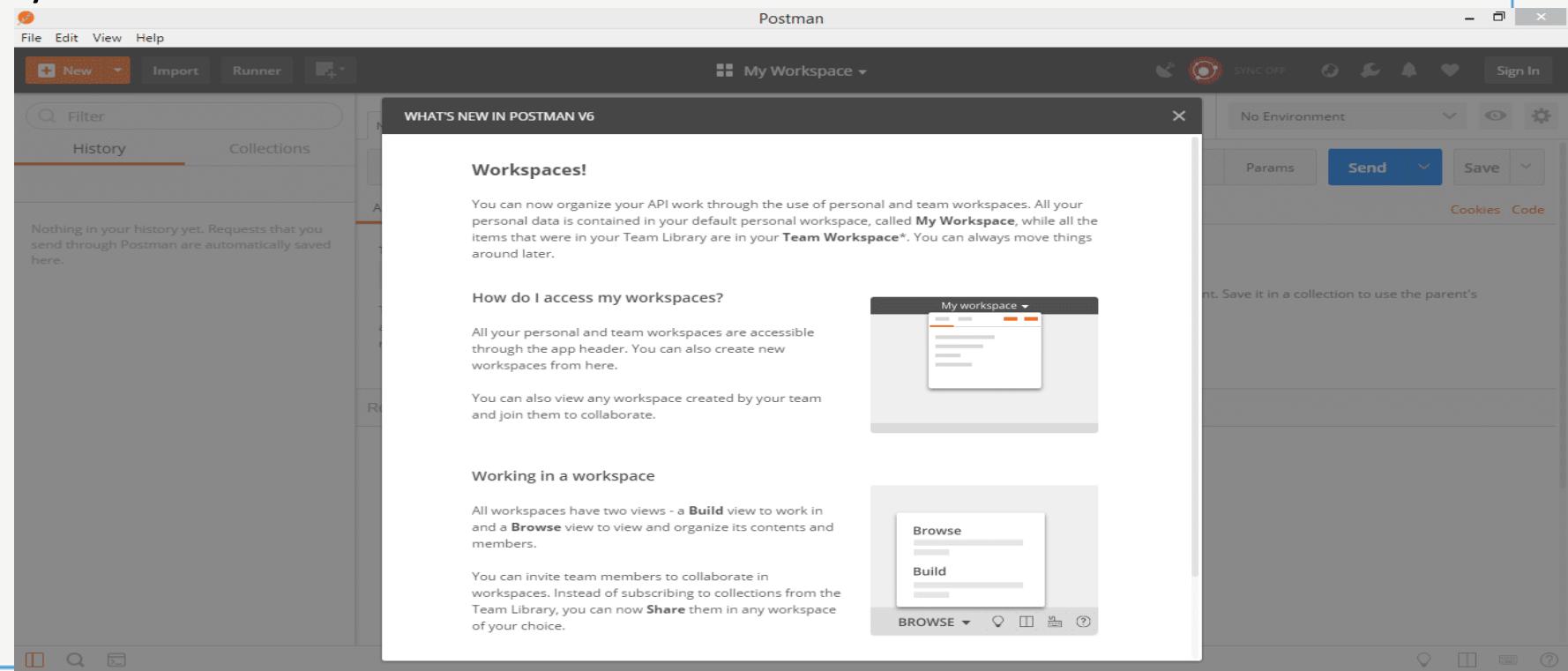
Download Postman as a Standalone Application

6. Once you have the application window up, click on ***Take me straight to the app. I'll create an account another time as highlighted***. Alternatively, you can sign up with google but it does not matter at present.



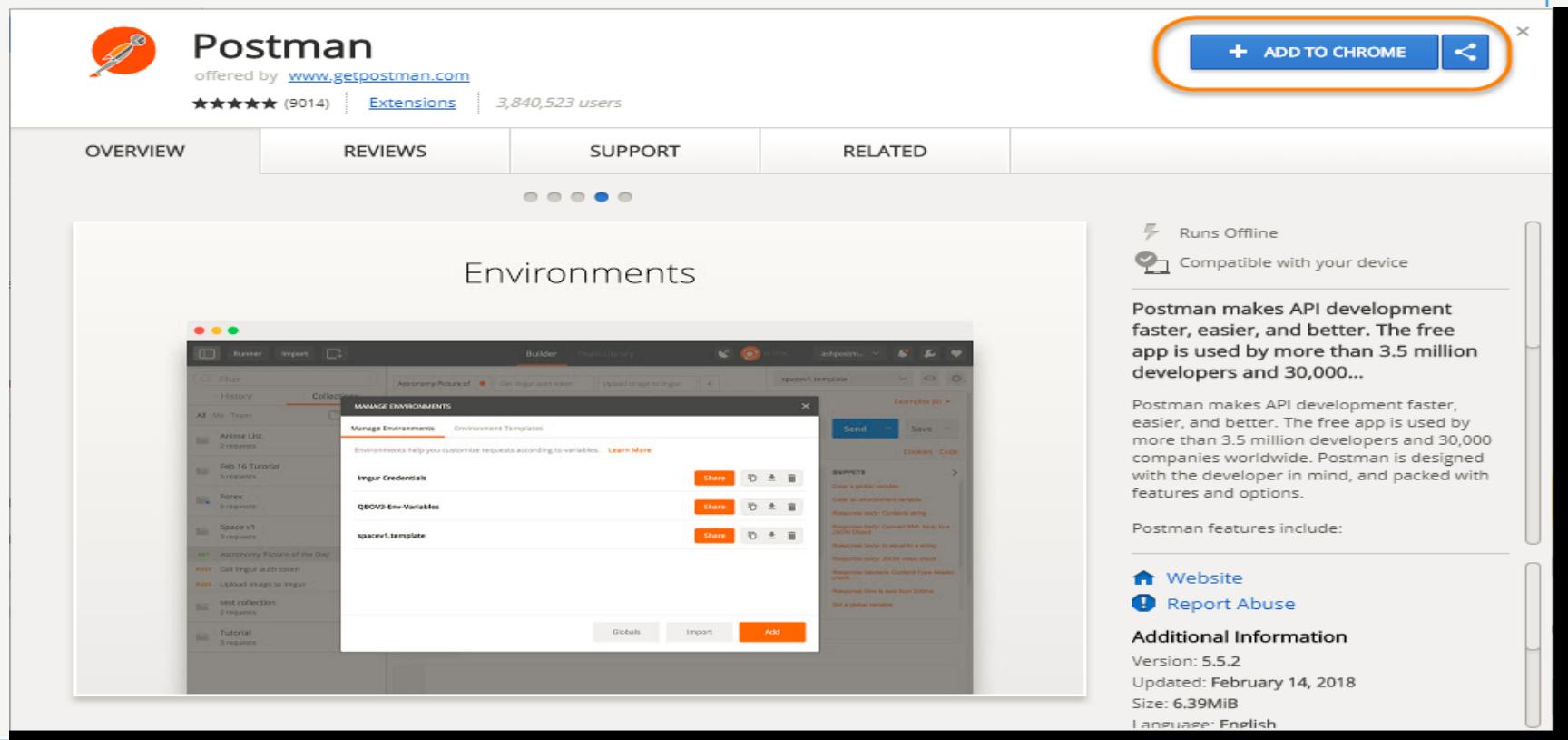
Download Postman as a Standalone Application

If you see this page then you have successfully installed Postman on your system.



Download POSTMAN as a Chrome Extension

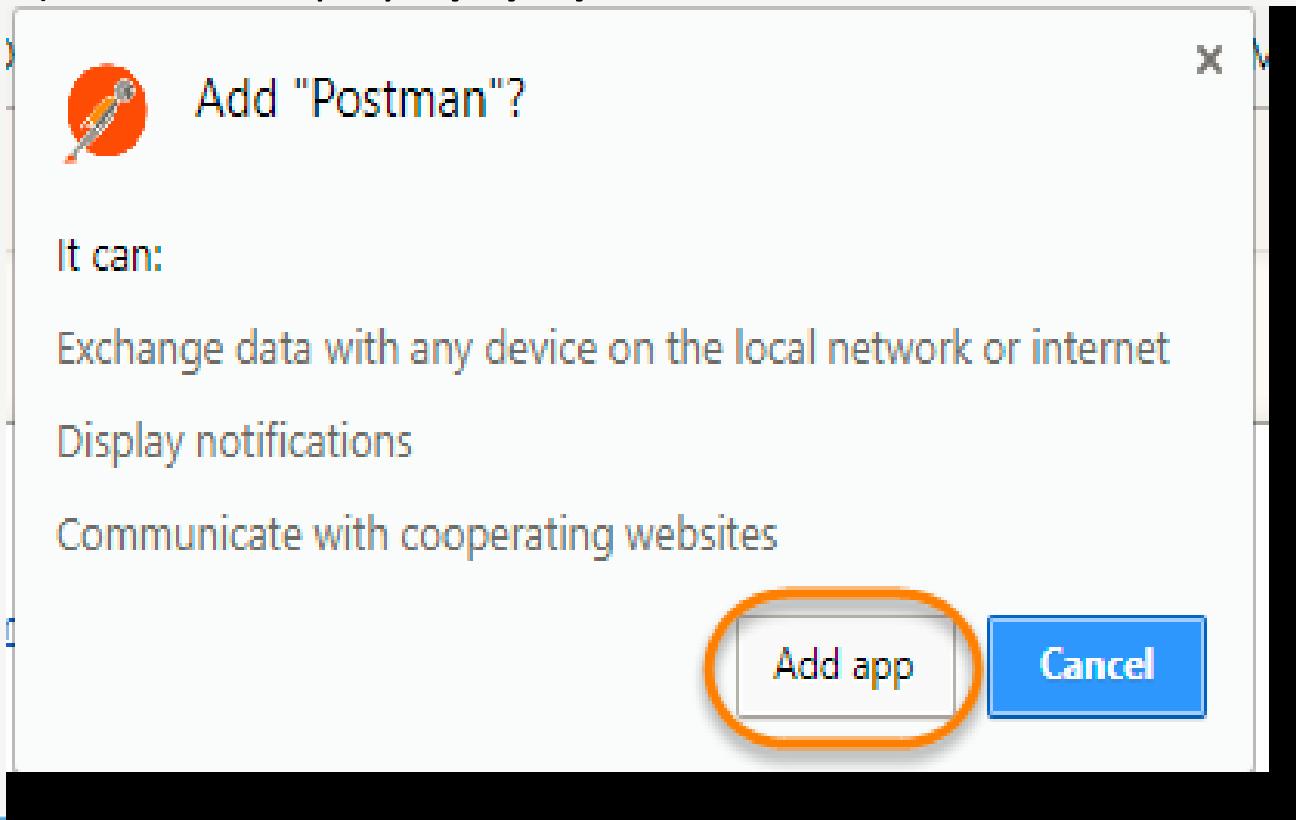
1) Go to [Chrome WebStore - PostMan Tool](#) and click on **Add To Chrome**.



The screenshot shows the Postman extension page on the Chrome Web Store. At the top, there's a large orange button with a white plus sign and the text "ADD TO CHROME". This button is circled in orange. Below it, there's a section with icons for "Runs Offline" and "Compatible with your device". A descriptive text block says: "Postman makes API development faster, easier, and better. The free app is used by more than 3.5 million developers and 30,000...". Another text block below it lists "Postman features include:" followed by a bulleted list: "Website", "Report Abuse", and "Additional Information". The "Additional Information" section provides details about the version (5.5.2), update date (February 14, 2018), size (6.39MiB), and language (English). On the left side of the page, there's a preview window showing the Postman interface with various API requests and environments.

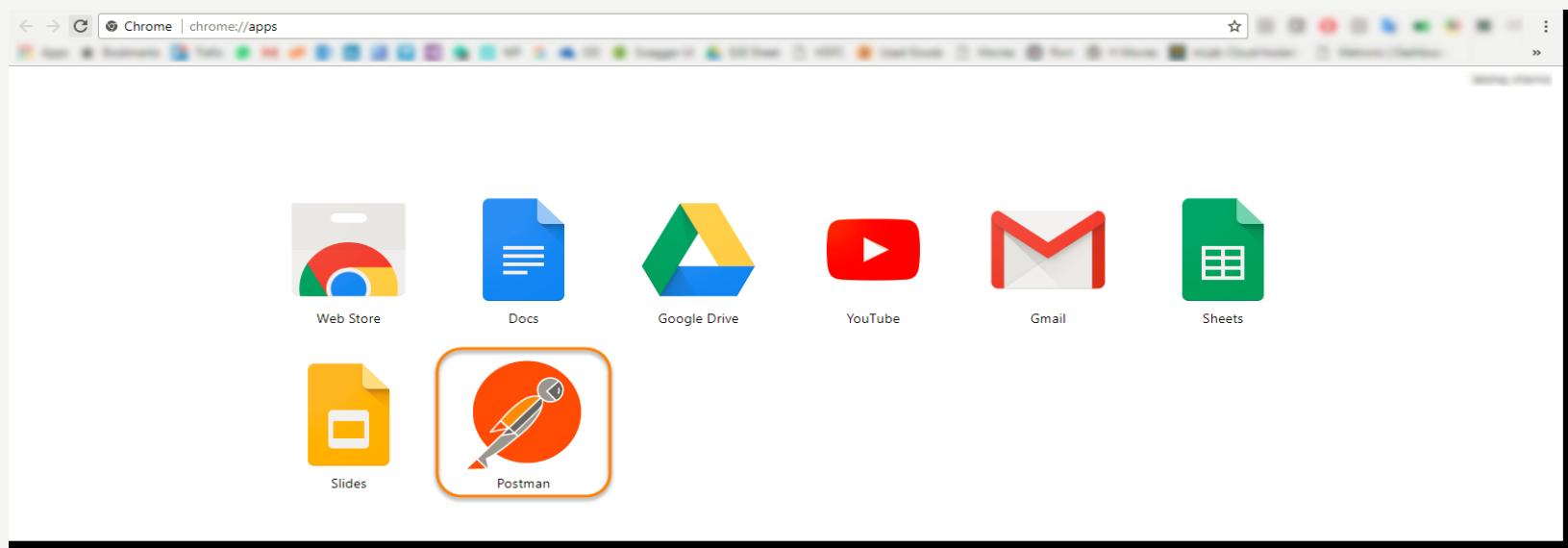
Download POSTMAN as a Chrome Extension

2) This will display a *pop up* to add extension, click on **Add app**.



Download POSTMAN as a Chrome Extension

3) Now it will automatically open a ***Chrome Apps*** page, where it will display all the installed apps on your chrome browser. Simply click on the ***PostMan*** application.



Download POSTMAN as a Chrome Extension

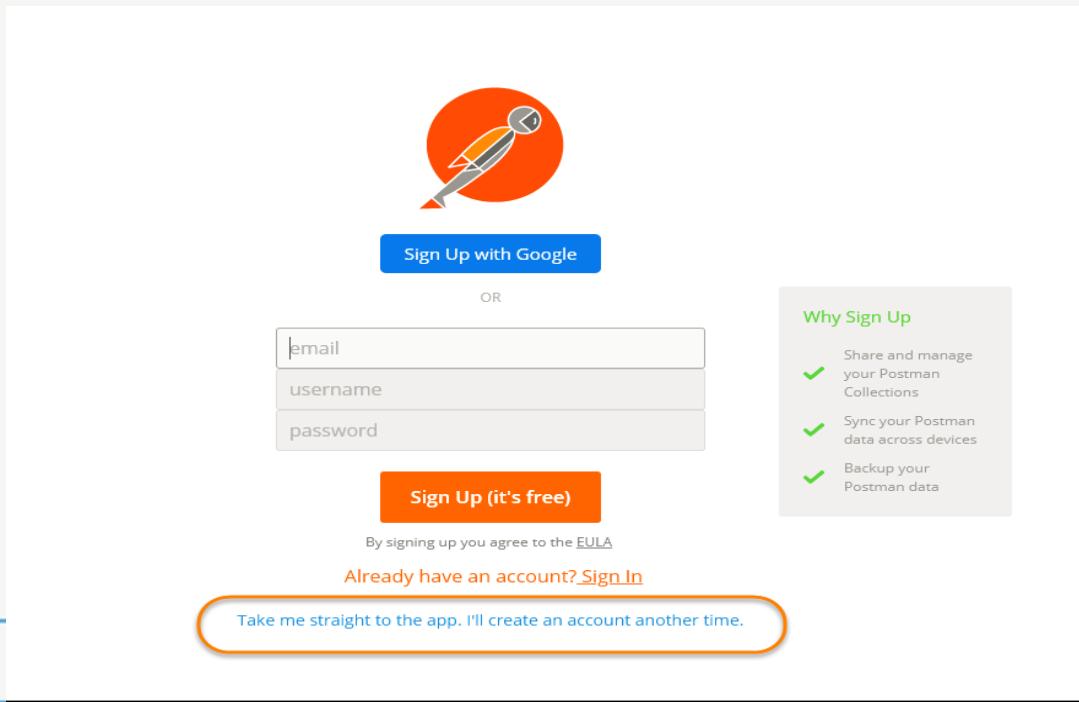
4) It will take a few seconds to start, hold on till then :)

Distorting space-time continuum...



Download POSTMAN as a Chrome Extension

5) Once done, you would see the PostMan application ***registration page***. As mentioned above, we will ignore registration as of now but not to avoid it, *Registration has its own benefits, as it always stores your data and can be accessed from different machines and locations. *But we will cover that later.



Why to prefer Postman as a Stand Alone application

- Although Postman was introduced first as a chrome application and was powerful earlier on, we highly recommend downloading postman as an application for your operative system(*native application*) rather than as an extension for chrome.
- **There are two main reasons for this.**
- ***First of all, postman as an application for chrome does not support all the features that the native app has.***
- ***For example, a proxy cannot be captured in the chrome app. You need to install another extension called **postman interceptor** in order to work the proxy through the browser. A proxy server acts as an intermediary that captures the requests that you send through your browser to the server.***

Why to prefer Postman as a Stand Alone application

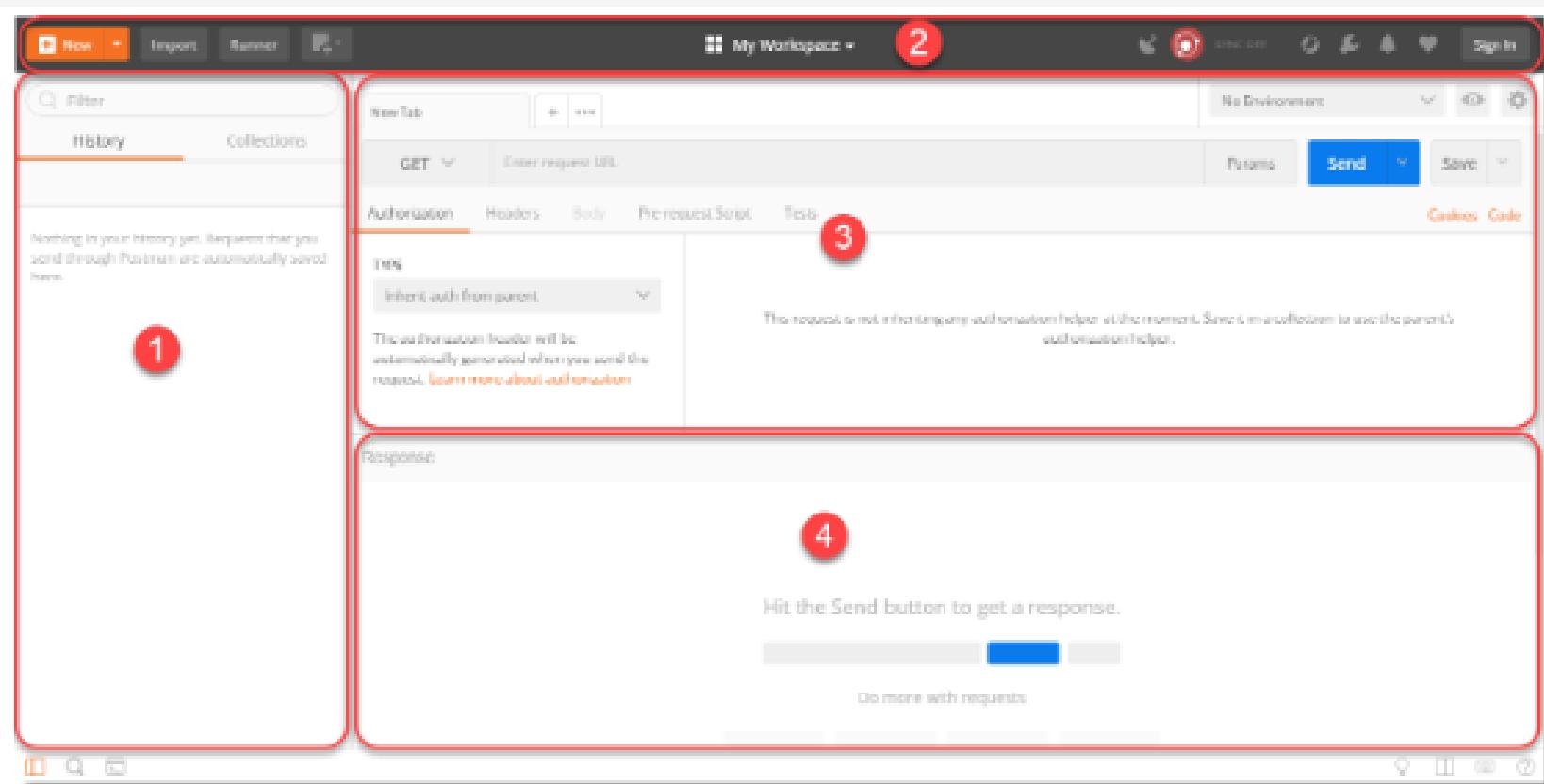
- Requests are anything like any web address any search query or anything that asks for a response from server.
- For example, when you hit "**Search**" in google after writing something in the search bar, it is an API request or you type www.google.co.in in the address bar, it is an API request.
- After installing the postman interceptor only you can capture the requests that you send. Moreover, Postman features fewer menu options in its chrome app, only those which adhere to the chrome standards.
- Therefore you won't be able to enjoy every feature of Postman while using it as a chrome application.

Why to prefer Postman as a Stand Alone application

- *The second reason is the main concern for us to not recommend you, download postman, for chrome.*
- *As stated on the Postman website, Postman builders have stopped the support for the chrome application.*
- *This in simple terms means that from January 2018 onwards, there will be no updates, no bug fixes, no improvements in the chrome application. The app that is currently available on the web-store is the final application.*
- *Moreover, if you are stuck on any issue, there will be no one to resolve it from Postman. This step is taken by Postman after Google announced that it will be ending support for the Chrome apps for Windows, Linux and Mac.*

Postman Navigation

- Postman navigation can be divided into four UI structures as shown below



Postman Navigation

1) Sidebar section

History

Collections

2) Header section

New

Import

Interceptor

Sync

3) Builder section: These items will help users to create a new Request. We will learn about these items in detail in the coming chapters

Tabs

HTTP Method type

URL bar

Header's list

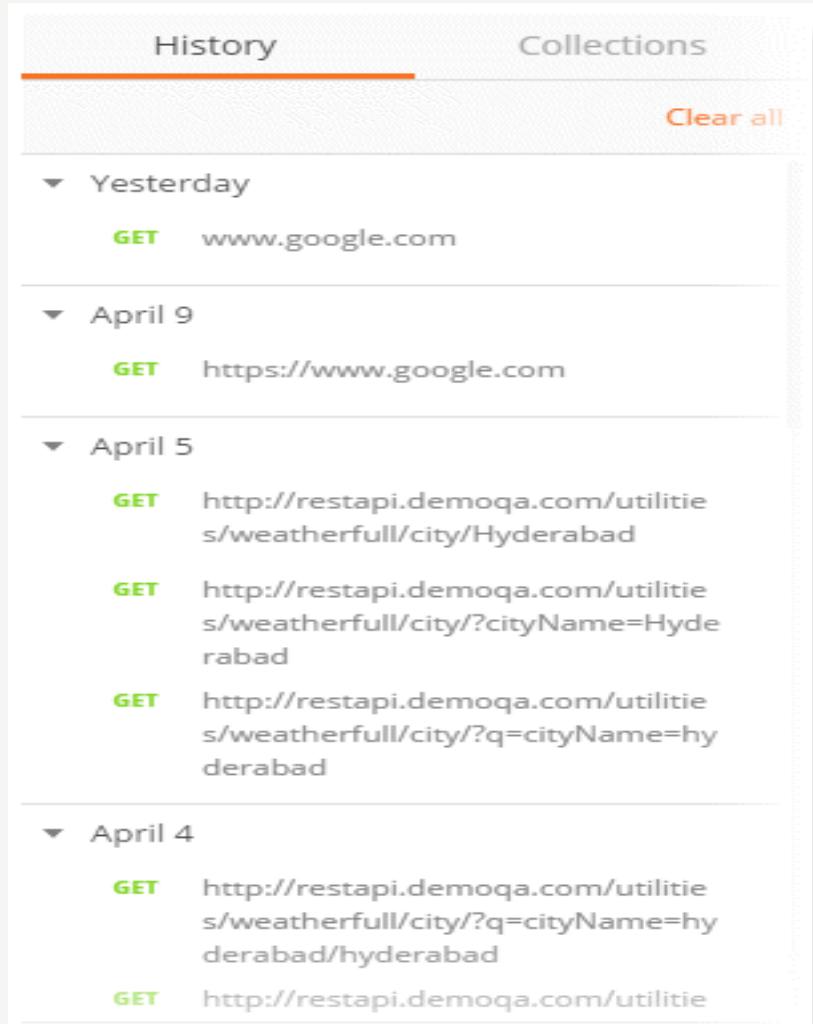
4) Response section: It is filled only when invoking a REST request. This section will be populated with the details of the received Response. We will learn more about it in the coming chapters. Now let us see individual sections in detail.

Postman Navigation

- **PostMan - Left Sidebar Section**
- The sidebar is a very important part of the Postman. The sidebar has two main parts or tabs which are ***History*** and ***Collections***.
- ***History Tab*** : Postman records a history of your API request just like any other web browser automatically.
 - As soon as you invoke a REST request, it is saved in the history and can be seen below the ***History Tab***.
 - It comes in handy when you have to search for some particular request that you entered in the past without entering again.

Postman Navigation

- PostMan - Left Sidebar Section
- *History Tab :*



The screenshot shows the 'History' tab in Postman's sidebar. At the top, there are tabs for 'History' (which is selected) and 'Collections'. A red underline highlights the 'History' tab. Below the tabs, there is a 'Clear all' button. The main area displays a list of requests grouped by date. For each date, there is a expandable arrow icon. Under each date, the request method (e.g., GET), URL, and a brief description of the endpoint are listed.

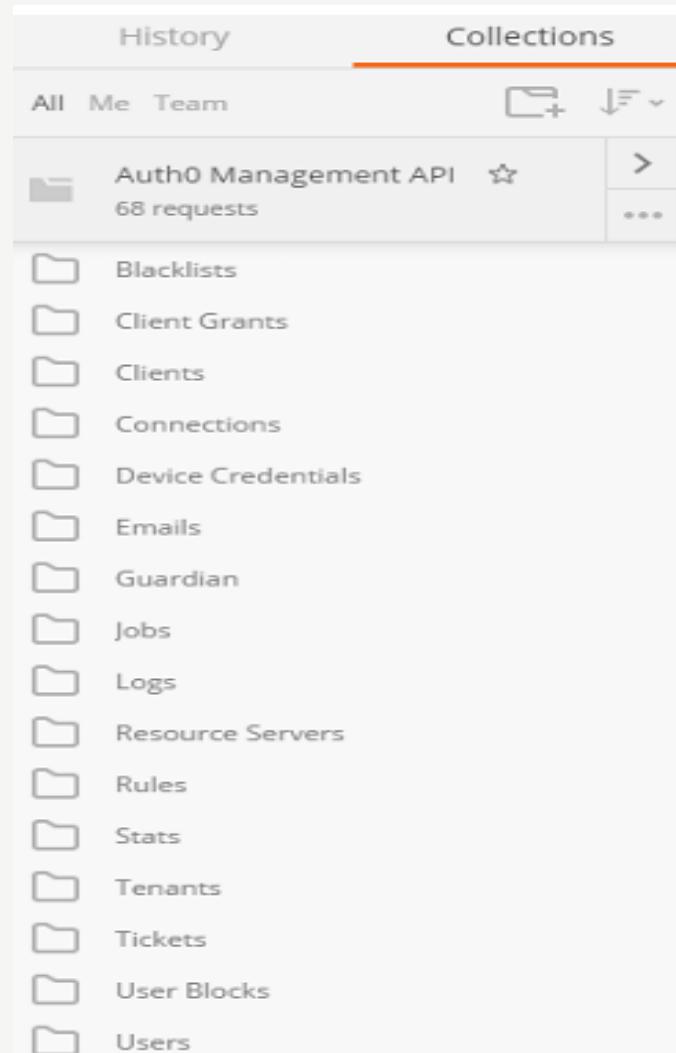
Date	Method	URL	Description
Yesterday	GET	www.google.com	
April 9	GET	https://www.google.com	
April 5	GET	http://restapi.demoqa.com/utilities/weatherfull/city/Hyderabad	
	GET	http://restapi.demoqa.com/utilities/weatherfull/city/?cityName=Hyderabad	
	GET	http://restapi.demoqa.com/utilities/weatherfull/city/?q=cityName=hyderabad	
April 4	GET	http://restapi.demoqa.com/utilities/weatherfull/city/?q=cityName=hyderabad/hyderabad	
	GET	http://restapi.demoqa.com/utilities/weatherfull/city/	

Postman Navigation

- PostMan - Left Sidebar Section
- **Collections** :The concept of grouping requests is called **Collections** and each **Collection** is displayed under the **Collection** Tab.
 - As shown in the image below. A collection in Postman can be imagined similar to a folder in your system.
 - You create a folder, for example, movies, and keep movies in it so that you know where all your movies are.
 - Similarly in Postman, we save the similar kind of requests under some collection name (*that we define*) and when we open any collection we get all the Requests under that heading, As shown in the below image

Postman Navigation

- PostMan - Left Sidebar Section
- *Collections*



Postman Navigation

- PostMan - Header Section
- The below image shows just the Header of the Postman application.



New :

- Choosing this option will let you choose what "**new**" you want to start. For example, a collection would open the panel where you can enter a new collection to start and its corresponding requests.
- Selecting "**request**" in **New** will open the request panel where you can enter and save the requests into the collection of your choice.
- A new option lets you create the following:

Request

Collection

Environment

Documentation

Mock Server

Monitor

Postman Navigation

- **PostMan - Header Section**
- ***import***: Import option lets you import files of different formats. Importing means choosing the files located in your system or through a link and running it through Postman. As can be seen from the image it allows you to import a ***Postman Collection, Environment, Curl command***, etc.
- ***Interceptor*** : Recall we learned that if you are installing the application from chrome then a separate interceptor is required for the proxy server. This interceptor is inbuilt in the native app. You can set a proxy server here to capture all the API requests that you send through your browser.
- ***Sync*** : Sync option is for synchronizing the API requests that you have sent on any machine to the Postman cloud. When you are working in Postman and making changes or sending requests, if you ***Sync*** is on, it will automatically be saved in your ***Postman's cloud storage***.

Postman Navigation

- PostMan - Builder Section

A builder part of the **Postman** is basically what a CPU is to a computer. It is the main part that controls all the functionalities and methods to be incorporated inside the API.



- builder part has the following main parts:

Request Type:

Endpoint Address Bar:

Params:

Postman Navigation

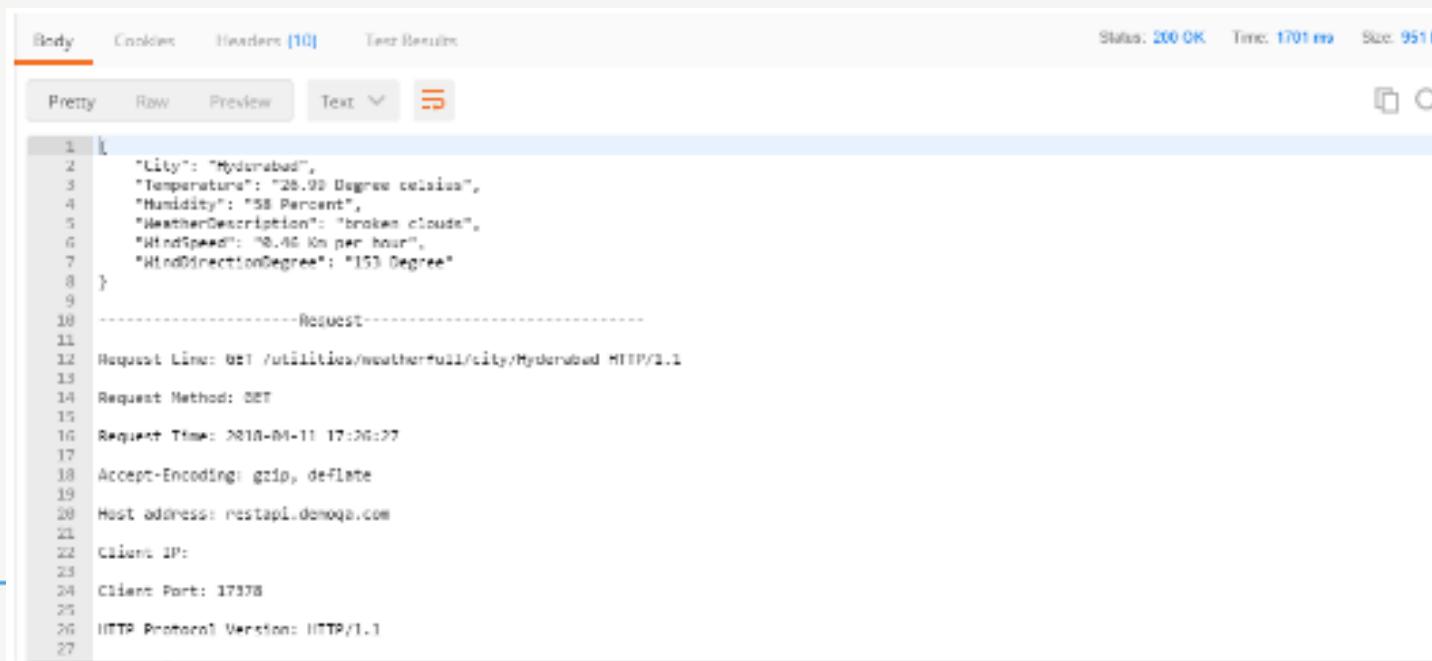
- **PostMan - Builder Section**
- **Request Type** : This is the request type method for the API. It indicates the type of **HTTP Request** that has been sent. There are different kinds of requests which we will discuss as we proceed further, but just to know, there are four main types of requests namely **GET, POST, PUT** and **DELETE**.
- **Endpoint Address Bar** : This is the box, besides the request type option, to enter the **EndPoint (API)**. It acts just like a browser with a similar interface for the New tab. We enter our required endpoint into the bar which is our main URL.
- **Params** : Params are the parameter option that allows us to write the parameters of the URL. The parameters are embedded into a URL and are very important to get the desired result. They also help us in getting efficient usage of the memory and bandwidth. This will be discussed in a complete chapter later on.

Postman Navigation

- **PostMan - Builder Section**
- **Authorization :** The authorization process verifies whether you have permission to access the data you want from the server. Not all data is available for everyone inside a company, so there lies the solution as Authorization. With the authorization, the server first checks whether the data you are asking for can be shown to you. If it can be, you get the desired response.
- **Header :** A header in the HTTP request or response is the additional information that is needed to be conveyed between the client-server. HTTP headers are mainly intended for the communication between the server and client in both directions.

Postman Navigation

- **PostMan - Response Section**
- A response box is a box that shows the response from the server that we receive after requesting through API. A response box has many options in it, which won't be feasible to explain here in this chapter. In the coming chapters you will learn about the response, although if you want you can visit the chapter *here*.

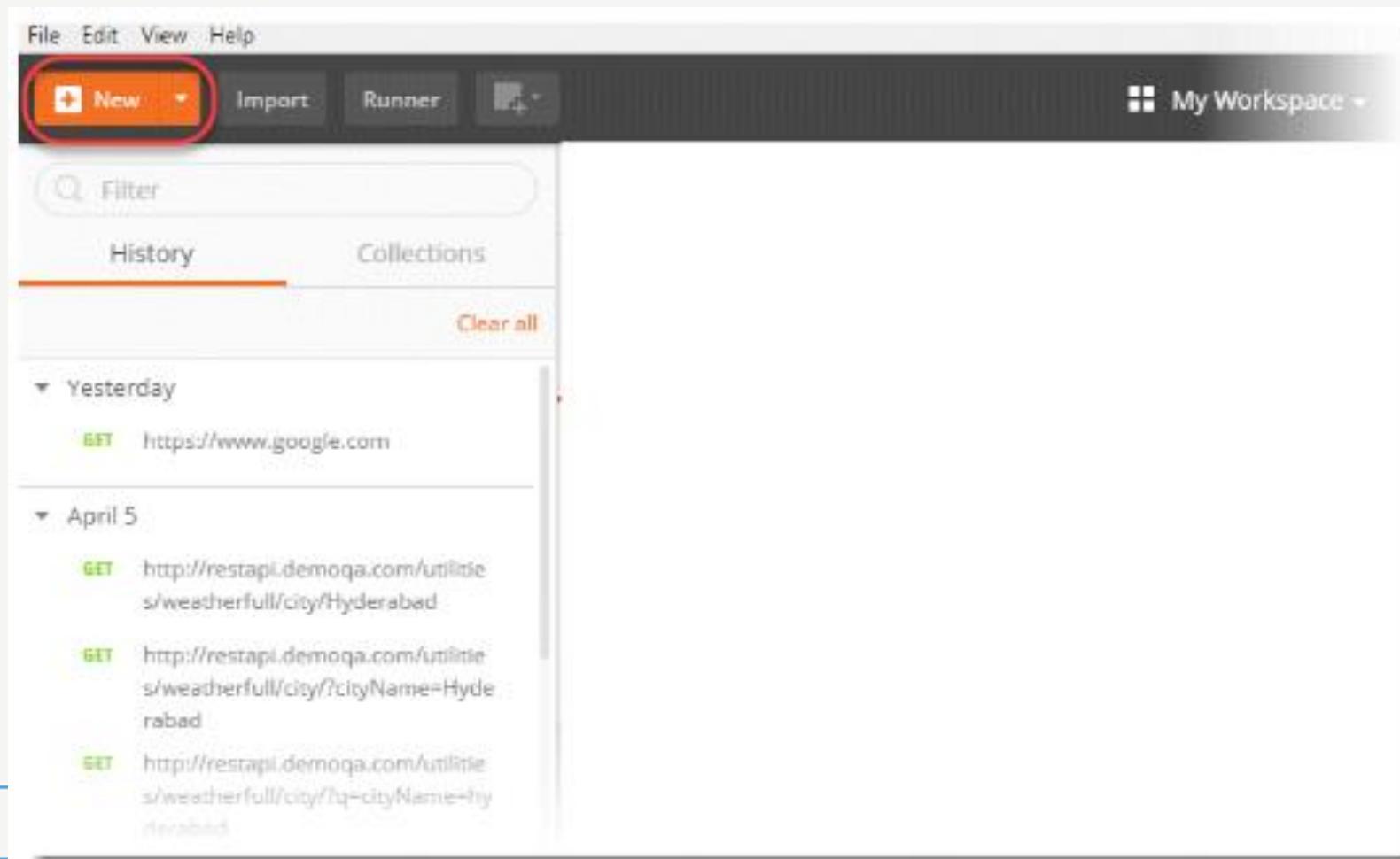


The screenshot shows the Postman application interface. At the top, there are tabs for Body, Cookies, Headers [10], and User Results. On the right, status information is displayed: Status: 200 OK, Time: 1701 ms, Size: 951 B. Below these tabs, there are buttons for Pretty, Raw, Preview, and Text, along with a JSON icon. The main area contains two sections: 'Response' and 'Request'. The 'Response' section shows a JSON object representing weather data for Hyderabad. The 'Request' section displays the HTTP request details, including the request line, method, time, headers, host address, client IP, port, and protocol version.

```
1  {
2   "City": "Hyderabad",
3   "Temperature": "26.92 Degree celsius",
4   "Humidity": "58 Percent",
5   "WeatherDescription": "broken clouds",
6   "WindSpeed": "10.45 Km per hour",
7   "WindDirectionDegree": "153 Degree"
8 }
-----Request-----
9
10 Request Line: GET /utilities/weather/full/city/Hyderabad HTTP/1.1
11
12 Request Method: GET
13
14 Request Time: 2018-04-11 17:06:27
15
16 Accept-Encoding: gzip, deflate
17
18 Host address: restapi.demqa.com
19
20 Client IP:
21
22 Client Port: 17378
23
24 HTTP Protocol Version: HTTP/1.1
25
26
27
```

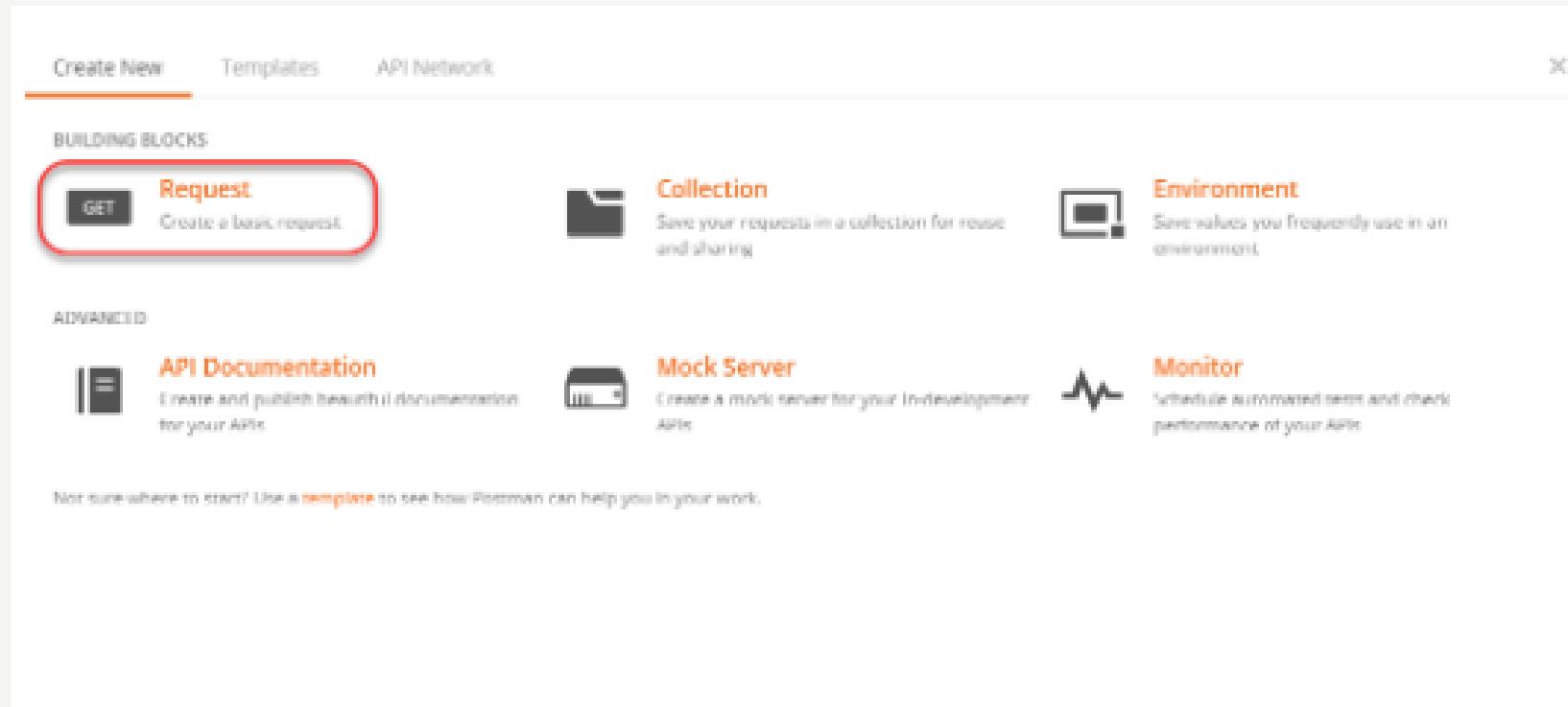
Create New Request in Postman

- 1) Click on the **NEW** option in the header part.



Create New Request in Postman

- 2) Click on *Request*.



The screenshot shows the 'Create New' screen in Postman. At the top, there are three tabs: 'Create New' (which is selected and highlighted in orange), 'Templates', and 'API Network'. Below the tabs, there's a section titled 'BUILDING BLOCKS' containing six items:

- Request** (highlighted with a red border): A button labeled 'GET' with the sub-label 'Create a basic request'.
- Collection**: An icon of a folder. Description: 'Save your requests in a collection for reuse and sharing'.
- Environment**: An icon of a monitor. Description: 'Save values you frequently use in an environment'.
- API Documentation**: An icon of a document. Description: 'Create and publish beautiful documentation for your APIs'.
- Mock Server**: An icon of a server. Description: 'Create a mock server for your development APIs'.
- Monitor**: An icon of a graph. Description: 'Schedule automated tests and check performance of your APIs'.

At the bottom left, there's a note: 'Not sure where to start? Use a [template](#) to see how Postman can help you in your work.'

Create New Request in Postman

3) Enter a meaningful **Request Name**, like **First Api** we are using. You can also use the description about the API to remember later about what that API did for other teammates and yourself, but it's optional and we won't be using that in this tutorial.

Requests in Postman are saved in collections (a group of requests).

[Learn more about creating collections](#)

Request name

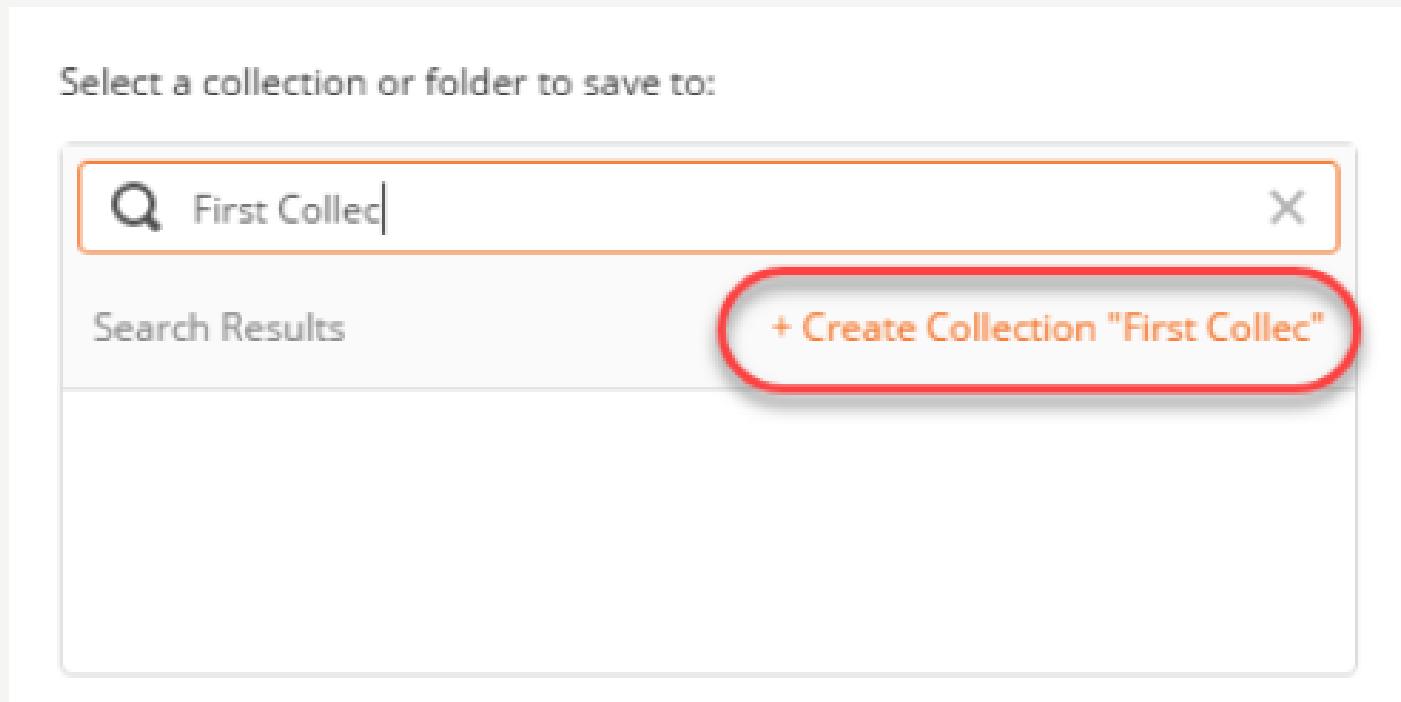
First Api

Request description (Optional)

Adding a description makes your docs better

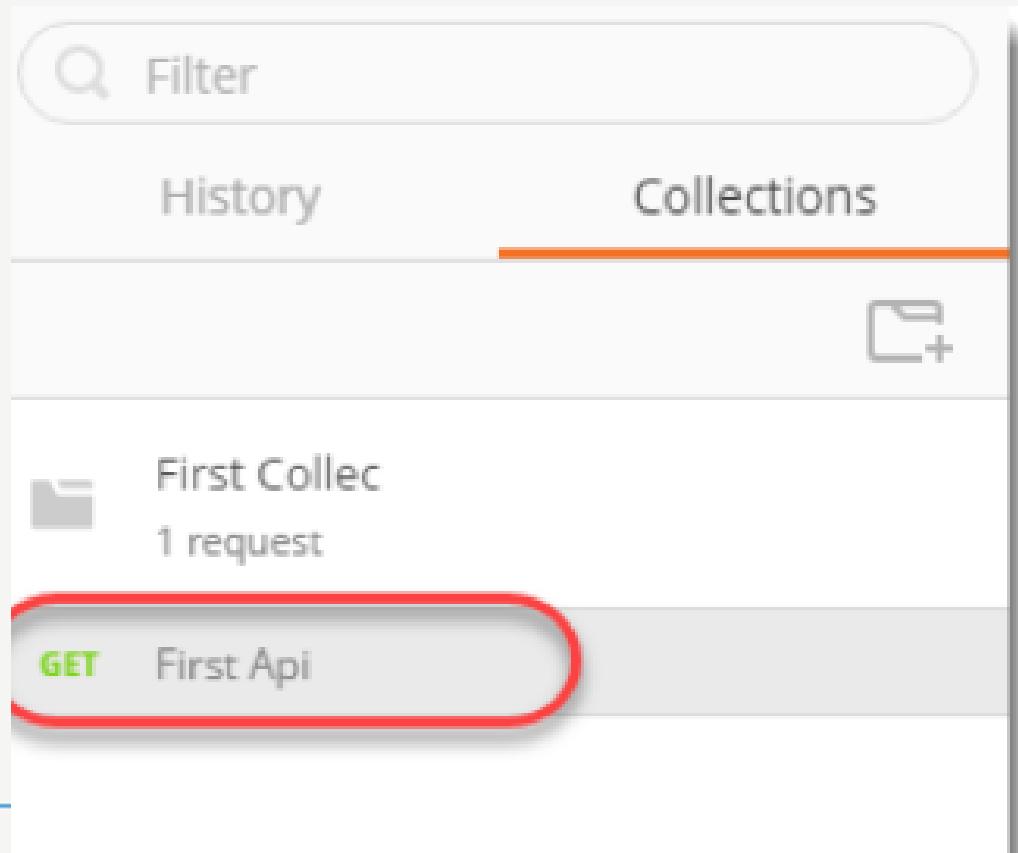
Create New Request in Postman

- 4) Enter a meaningful Collection name in the bottom panel, like ***First Collec*** we are using, and select **+Create Collection** as shown. Press **Save**.



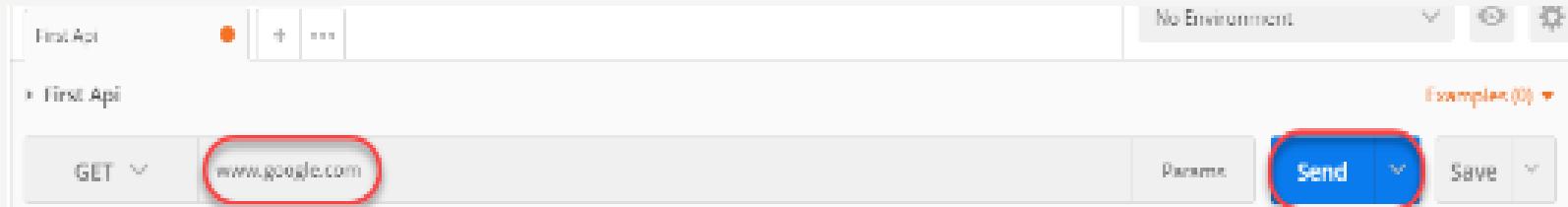
Create New Request in Postman

5) Select *Collections* tab in the sidebar, then you will notice all the collections folders, select *First Collec* and then select *First Api* under the *First Collec* tab.



Create New Request in Postman

6) Enter **www.google.com** in the **Address Bar** and press **Send**.



7) Press **Save** if you wish to overwrite "First API" or press the dropdown as shown and Save as a new request.



The **Save As** option opens the same panel which opened through **New Request** at the start of this tutorial. It gives the option to enter the name and associate the request to some collection.

This way you have created a Request and saved it under the desired collection. In the next tutorial, we will send our first **GET** request.

GET Requests in Postman

- Since we have now walked through Postman and seen ***How to Create and Save a new Request in postman***, it's time to get our hands on the first ***GET Request in Postman***.
- When we request from a client machine (*User*) to a server machine, we follow an architecture and HTTP Protocol.
- I suggest you go through the below tutorials to establish a nice understanding of ***HTTP Protocol, Request & Response***. These can be viewed here:

Client Server Architecture and HTTP Protocol

HTTP Request

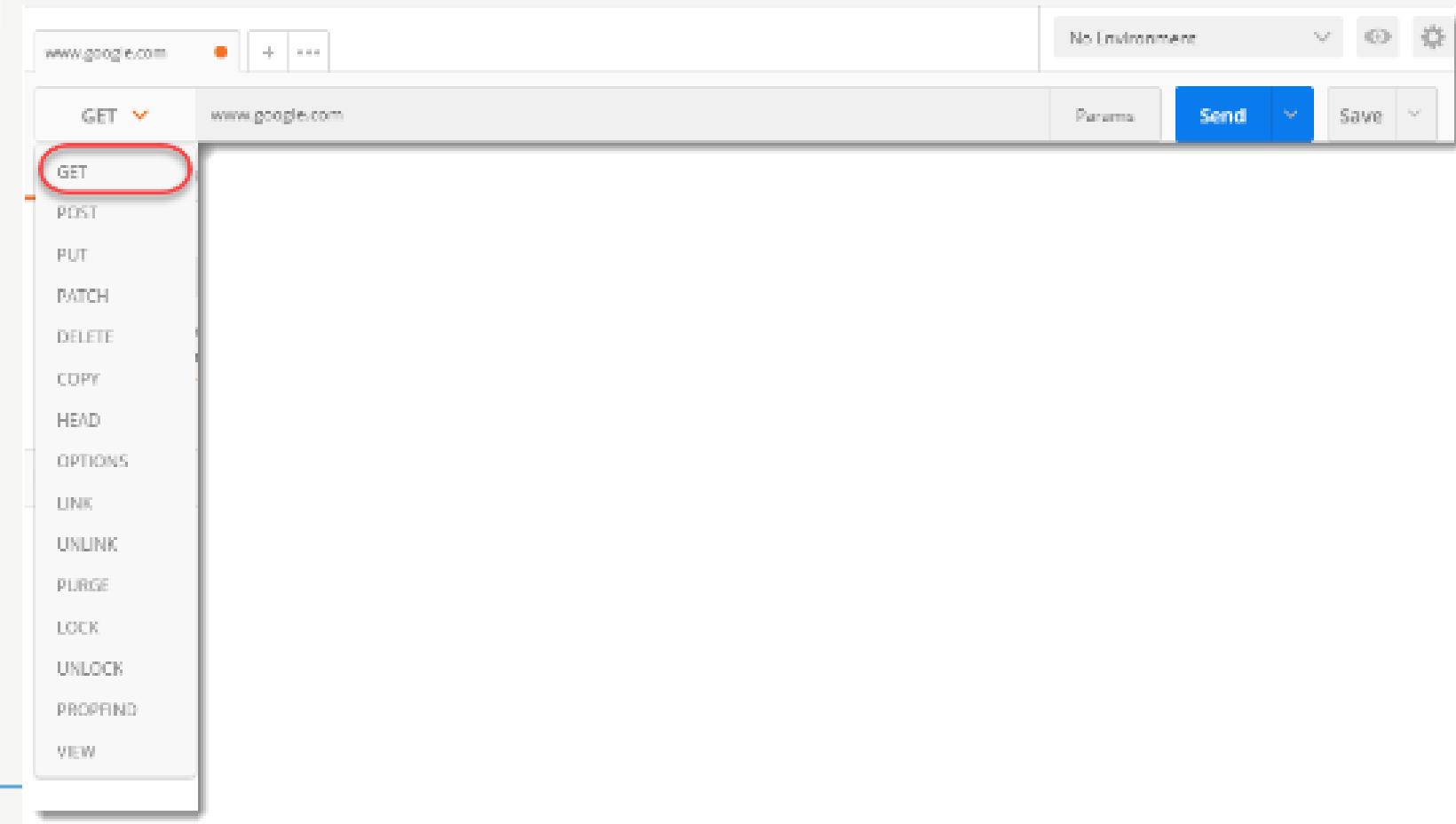
HTTP Response

GET Requests in Postman

- Assuming you are now familiar with the HTTP protocols and architecture, we will now talk about one specific type of request which is a ***GET*** request.
- A ***GET*** request is used to get the information from the server and does not have any side effects on the server.
- *Side-effects mean there is no updation/deletion/addition of data on the server when you are making this type of request, you just request from the server and the server responds to the request.*
- A ***GET*** request has all its information inside the ***URL***, and since URL is visible all the time, it is advisable not to use this type of request while you send some sensitive information such as passwords.
- *For example, when you press search after writing anything in the search box of google.com, you actually go for a ***GET*** request because there is no sensitive information and you are just requesting the page with search results, you notice the same search string in URL.*

GET Requests in Postman

1) Select **GET** from the list of request types.



GET Requests in Postman

2) Enter **www.google.com** in the address bar as written in the above image and **Press Send**. Now, look at the **Status Code**.



The screenshot shows the Postman interface with the following details:

- Body** tab is selected.
- Cookies (0)**, **Headers (1)**, and **Test Results** tabs are visible.
- Status: 200 OK** is highlighted with a red oval.
- Time: 1007 ms** and **Sent: 02:23:10** are displayed.
- Preview** tab is selected.
- Raw**, **Preview**, and **HTML** buttons are present.
- Raw** content is displayed as a large block of JSON-like data representing the Google homepage response.

GET Requests in Postman

- Different status codes have different meanings and it does not matter whether it is a GET request or any other type of request.
- In this scenario, we have status code **200 OK** which means that EndPoint is correct and it has returned the desired results. We will show some more status codes later.
- The colorful text inside the box below is the Response from the server. If you observe closely inside the response box you will see the page code has been sent to us.
- The above tab says **Body**. Body means you have selected to view the response body which is been shown inside the box. In Body, you will see three options.

GET Requests in Postman

- **Pretty:** In this code will be shown in a colorful manner with different keywords colored differently and will be indented for some of the formats for good reading.
 - **Raw:** Same as pretty part with no colors and in single lines.
 - **Preview:** This shows the preview of the page that has been sent. Don't worry about the google doodle if it has not been loaded properly. Try any other website by yourself.

Response in Postman

- ***What is Response?***

- A ***Response*** is a message that is received by the server in return to a ***Request*** that we send.
- When we request something, the server acts upon the ***Request*** and sends back a packet of the requested information.
- A response depends on the request mainly. Every request has a different kind of response and it is very important that we extract useful information from all of the responses.
- Postman has a beautiful interface for response and is very user-friendly.
- We can see a lot of information in the Postman for any response without doing much effort, or any if I might say.
- You may also go through the recording of the Postman Tutorial where our experts have explained the concepts in depth.

Response in Postman

Response information blocks

- Response Status and Information
- Response Body
- Response Cookies
- Response Header

Response in Postman

Response information blocks

- Response Status and Information
- 1) **Status Code :** A **status code** tells you the status of the request. There can be a lot of mistakes in the request and without looking at the status code, we might not always get what went wrong to our request. Sometimes, there can be a typing mistake in the URL or there can be a problem at the server-side, status code help us know about what went wrong (if something went wrong).
 - 2) **Time :** **Time** is the duration which the response took after we sent the request and received the response. This is very important sometimes because many projects have Service Level Agreements(SLA) for the time
 - 3) **Size :** **Size** is just the response size when it will be saved inside the memory. This response size is the size of complete response and headers and cookies and everything that has been sent along with the response.

Response in Postman

Response information blocks

- Response Body : A **body** depicts the body of the response, which is the main response content, that has been sent from the server.
- In this case as you can see it is a web page code being sent to us as a response.
 - 1) Pretty :
 - 2) Raw
 - 3) Preview
 - 4) Format Type

Response in Postman

Response information blocks

- **Response Cookie :** Cookies are the small files which are related to the server files (*website pages*).
- Once you visit a website for the first time, a cookie is downloaded on the client's machine.
- This cookie contains the information which can be used by the same website when you visit again.
- This helps the website to get you the specific response and specific information based on your last visit.
- In postman we can clearly see the cookies that have been sent from the server as a response.
- This makes it easy for the client to see what cookies are being saved inside his browser.
- We cannot manipulate this cookies since they are sent from server, Postman is used just to separate it from the response and have a clear view.

Response in Postman

Response information blocks

- Response Header :**Headers** in an HTTP request or response is the additional information that is transferred to the user or the server.
- In postman, the headers can be seen in the **Headers** tab.
- Once you click on header you can see different information such as below.
 - 1) **Content-Type** : *This is the content type of the response. In the above example when we used www.google.com the content type is given as **text/HTML** because the response is being sent in the HTML which is one of the options.*
 - 2) **Date** : *This option shows the date, day and time of the response along with the time zone.*
 - 3) **Server** : *This option tells the name of the server which has responded to the request. In the above example, the server name is shown as **gws** which corresponds to **Google Web Server**.*
 - 4) **Cookie expire time** : *As the name suggests, this option tells the expire time of the cookie that has been sent along with the response.*

Request Parameters in Postman

What are parameter in postman?

- Request Parameters are part of the URL which is used to send additional data to the Server. Let us analyze a simple URL:
<https://www.google.com/search?q=Rahulsanghavi>
- In this URL Request parameter is represented by the "*q=Rahulsanghavi*" part of the URL.
- Request parameter starts with a question mark (?).
- Request parameters follow "**Key=Value**" data format. In our example "*q*" is the Key and "*Rahulsanghavi*" is the value.
- The server reads the Request parameter from the URL and sends a Response based on the Request Parameter.
- **Parameters** can be passed in **GET Request**, if you are not sure how to do a GET Request using Postman

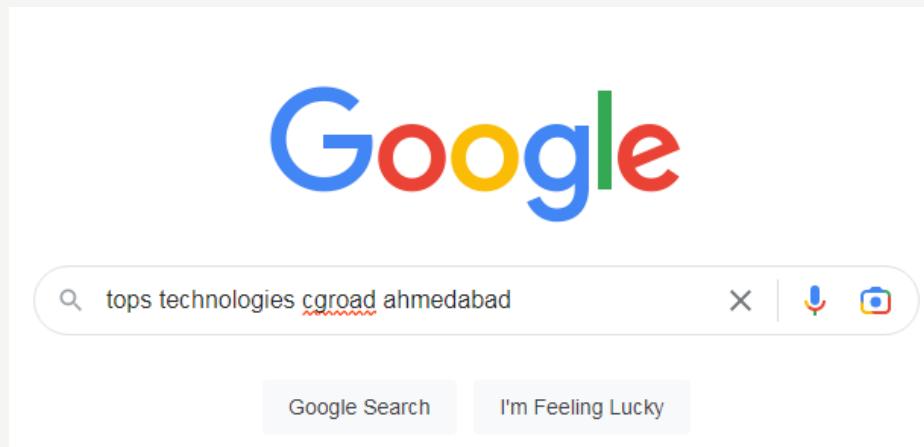
Request Parameters in Postman

What are parameter in postman?

- 1) Go to your browser and write **www.google.com** in your address bar



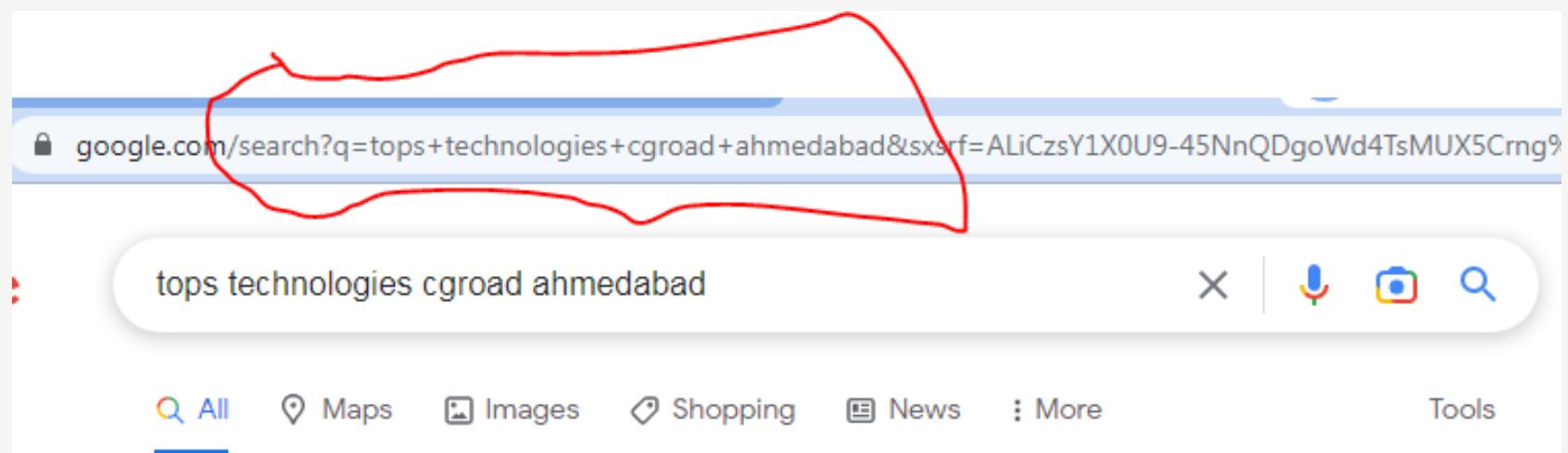
- 2) You will see the response page from Google. Type **Tops Technologies cgroad ahmedabad** in the search bar and press **Google Search**.



Request Parameters in Postman

What are parameter in postman?

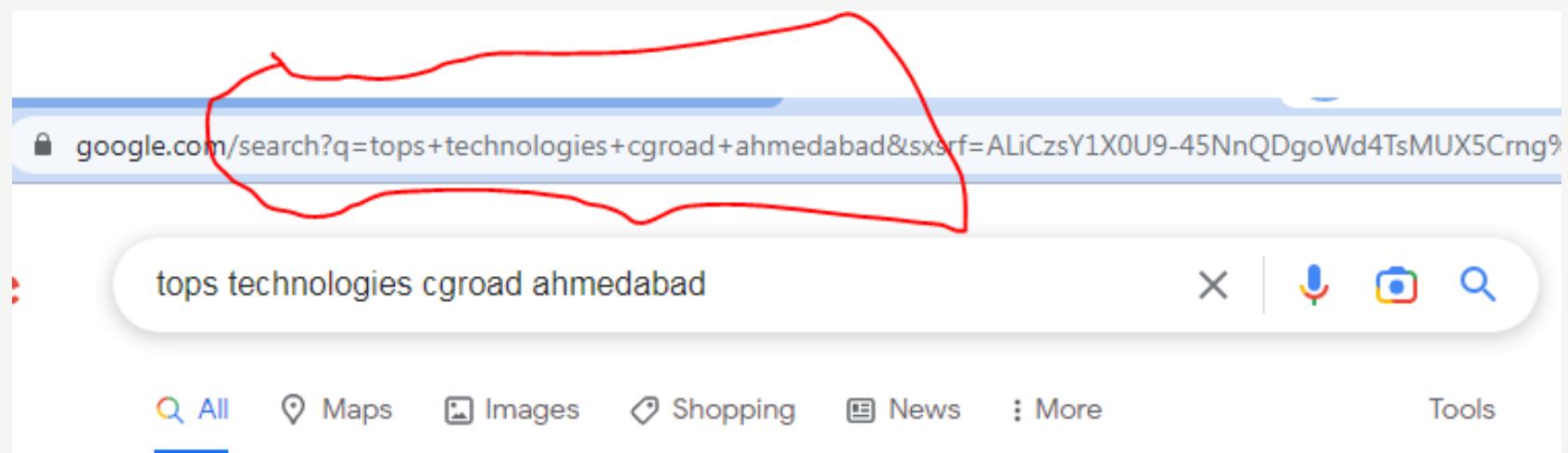
3) Look at the preview, you would see that instead of the google home page we have received a response for a specific search query which is **Tops technologies cgroad ahmedabad**. Instead of **Tops technologies cgroad ahmedabad** you could write anything and receive its response. This indicates that we have passed some information (*Parameters*) about the result we wish to see.



Request Parameters in Postman

What are parameter in postman?

3) Look at the preview, you would see that instead of the google home page we have received a response for a specific search query which is **Tops technologies cgroad ahmedabad**. Instead of **Tops technologies cgroad ahmedabad** you could write anything and receive its response. This indicates that we have passed some information (*Parameters*) about the result we wish to see.

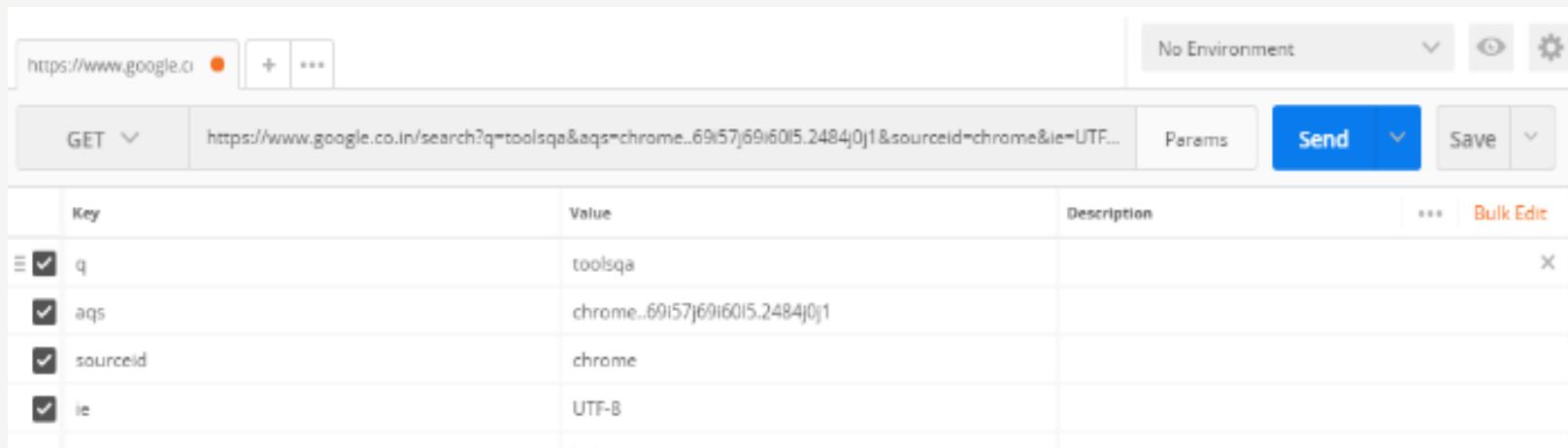


Request Parameters in Postman

Copy parameters to another Postman Request

- Another interesting feature about *Params* is that Postman removes the headache of remembering and entering the same parameters again and again to every query, instead it lets you enter once and forget about entering the same parameters again

1) Click on **Bulk Edit**, you will see the list of all parameters



The screenshot shows the Postman interface for a GET request to <https://www.google.co.in/search?q=toolsqa&aqs=chrome..69i57j69i60i5.2484j0j1&sourceid=chrome&ie=UTF-8>. The 'Params' tab is selected. A table lists four parameters:

Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/> q	toolsqa			
<input checked="" type="checkbox"/> aqs	chrome..69i57j69i60i5.2484j0j1			
<input checked="" type="checkbox"/> sourceid	chrome			
<input checked="" type="checkbox"/> ie	UTF-8			

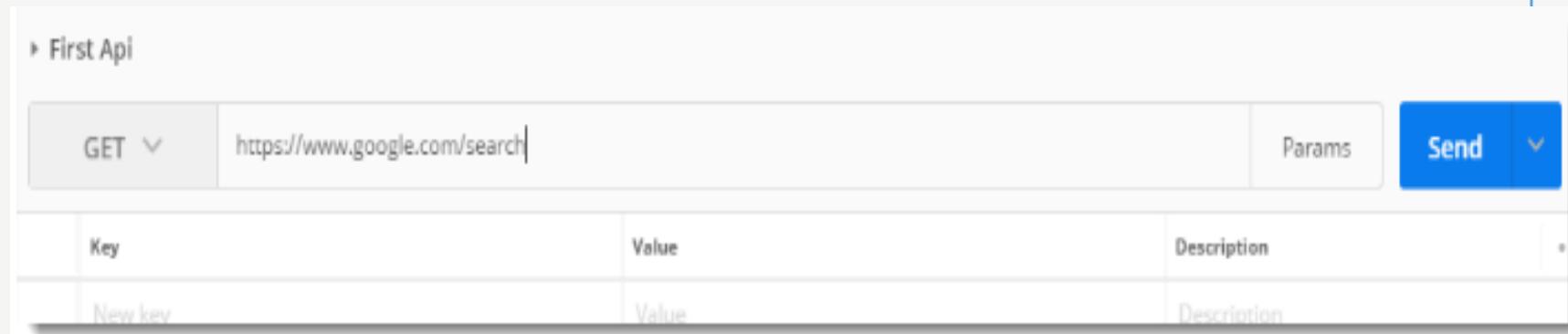
Request Parameters in Postman

Copy parameters to another Postman Request

2) Copy everything

```
q:toolsqa  
aq:chrome,.69i57j69i66l5.2484j0j1  
sourceid:chrome  
ie:UTF-8
```

3) Open a new tab and write your URL which is ***www.google.com/search*** in this case



The screenshot shows the Postman application interface. At the top, there's a header with 'First Api' and a dropdown menu. Below it, a search bar contains 'GET' and the URL 'https://www.google.com/search'. To the right of the URL are buttons for 'Params' and 'Send'. A blue dropdown arrow is visible next to the 'Send' button. Below the search bar is a table with three columns: 'Key', 'Value', and 'Description'. There is one row in the table with the text 'New key' in the 'Key' column.

Request Parameters in Postman

Copy parameters to another Postman Request

- 4) Click on **Params**, then **Bulk Edit**

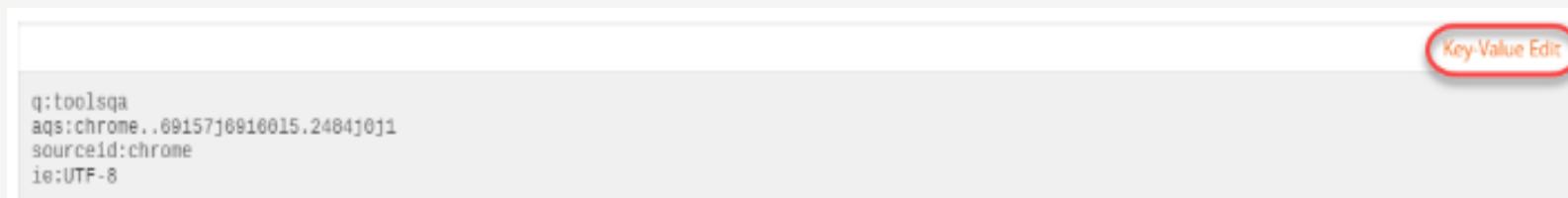


The screenshot shows the Postman interface for a 'First Api'. The request method is 'GET' and the URL is 'https://www.google.com/search'. In the top right, there are buttons for 'Params' (circled in red with number 1), 'Send', 'Save', and 'Examples (0)'. Below these, there's a table for parameters:

Key	Value	Description
New key	Value	Description

A 'Bulk Edit' button is also circled in red with number 2.

- 5) **Paste** everything you copied in the editor and click on **Key-Value edit**



The screenshot shows the 'Key-Value Edit' interface. It contains the following text:
q:toolsqa
aq:chrome..69157j6910015.2484j0j1
sourceid:chrome
ie:UTF-8

A 'Key-Value Edit' button is circled in red.

POST Request using Postman

What is a POST Request?

- A ***POST*** is an ***HTTP Verb*** similar to a ***GET*** request, this specifies that a client is posting data on the given ***Endpoint***.
- A ***POST*** request is a method that is used when we need to send some additional information inside the body of the request to the server.
- When we send a POST request we generally intend to have some modification at the server such as ***updation, deletion, or addition***.
- ***One of the classic example of a POST request is the Login page.***
- ***When you first Sign Up for anything, let's say Facebook, you send your personal information such as a password to the server.***
- ***The server creates a new account with the same details and that account is added permanently on the Facebook server.***
- You just created a new resource on to the server. ***POST*** requests are very popular and are mostly used whenever you are sending some sensitive information

POST Request using Postman

POST Request in Postman

- Every **REST endpoint** has its own **HTTP verb** associated with it. If an endpoint specifies that it should be called using the POST HTTP verb, then clients are bound to call the Endpoint with **POST HTTP verb** only.
- Let's first check what happens when we request the **GET** method instead of the **POST** method for a **POST Endpoint**. Also to check what happens when we do **POST Request** without **Body**.

GET Request on POST Endpoint

- Use the API **`http://restapi.demoqa.com/customer/register`** (*This API is used for registering a new customer*) in the Postman endpoint bar and press **Send**. Make sure that **GET** is selected in the Method type dropdown.
- See the HTTP status code, it will be **405 Method not allowed**. Which means that we are hitting the endpoint with incorrect method type. The below image shows the details.

POST Request using Postman

POST Request in Postman

- See the response below under the **Body** tab and focus on ***fault error***.
- It means that the method type we used is not valid and another method type is expected.
- So we will try to change that and see if we get the correct response.