What is Software Testing?

Software testing is the process of evaluating and verifying that a software product or application does what it is supposed to do. The benefits of testing include preventing bugs, reducing development costs and improving performance.

Software Testing is a method to check whether the actual software product matches expected requirements and to ensure that software product is Defect free. It involves execution of software/system components using manual or automated tools to evaluate one or more properties of interest. The purpose of software testing is to identify errors, gaps or missing requirements in contrast to actual requirements.

Why Software Testing is Important?

Software Testing is Important because if there are any bugs or errors in the software, it can be identified early and can be solved before delivery of the software product. Properly tested software product ensures reliability, security and high performance which further results in time saving, cost effectiveness and customer satisfaction.

Software testing is the process of evaluating and verifying that a software product or application does what it is supposed to do. The benefits of testing include preventing bugs, reducing development costs and improving performance.

Benefits of Software Testing

Cost-Effective: It is one of the important advantages of software testing. Testing any IT project on time helps you to save your money for the long term. In case if the bugs caught in the earlier stage of software testing, it costs less to fix.

Security: It is the most vulnerable and sensitive benefit of software testing. People are looking for trusted products. It helps in removing risks and problems earlier.

Product quality: It is an essential requirement of any software product. Testing ensures a quality product is delivered to customers.

Customer Satisfaction: The main aim of any product is to give satisfaction to their customers. UI/UX Testing ensures the best user experience.

Types of Testing

Typically Testing is classified into three categories.

Functional Testing

Non-Functional Testing or Performance Testing

Maintenance (Regression and Maintenance)

What is Functional Testing?

FUNCTIONAL TESTING is a type of software testing that validates the software system against the functional requirements/specifications. The purpose of Functional tests is to test each function of the software application, by providing appropriate input, verifying the output against the Functional requirements.

Functional testing mainly involves black box testing and it is not concerned about the source code of the application. This testing checks User Interface, APIs, Database, Security, Client/Server communication and other functionality of the Application under Test. The testing can be done either manually or using automation.

What do you test in Functional Testing?

The prime objective of Functional testing is checking the functionalities of the software system. It mainly concentrates on –

- **Mainline functions**: Testing the main functions of an application
- **Basic Usability**: It involves basic usability testing of the system. It checks whether a user can freely navigate through the screens without any difficulties.
- **Accessibility**: Checks the accessibility of the system for the user
- **Error Conditions**: Usage of testing techniques to check for error conditions. It checks whether suitable error messages are displayed.

How to do Functional Testing

Following is a step by step process on **How to do Functional Testing** :

- Understand the Functional Requirements
- Identify test input or test data based on requirements
- Compute the expected outcomes with selected test input values
- Execute test cases
- Compare actual and computed expected results

Functional Testing Tools

Here is a list of popular **Functional Testing Tools**. They are explained as follows:

testRigor – most advanced codeless UI end-to-end functional testing tool. Automate test cases in plain English, no matter how long or complex they are.

- Selenium – Popular Open Source Functional Testing Tool
- QTP – Very user-friendly Functional Test tool by HP
- JUnit– Used mainly for Java applications and this can be used in Unit and System Testing
- soapUI – This is an open source functional testing tool, mainly used for Web service testing. It supports multiple protocols such as HTTP, SOAP, and JDBC.
- Watir – This is a functional testing tool for web applications. It supports tests executed at the web browser and uses a ruby scripting language

What is Non-Functional Testing?

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

An excellent example of non-functional test would be to check how many people can simultaneously login into a software.

Non-functional testing is equally important as functional testing and affects client satisfaction.

Objectives of Non-functional testing

- Non-functional testing should increase usability, efficiency, maintainability, and portability of the product.
- Helps to reduce production risk and cost associated with non-functional aspects of the product.
- Optimize the way product is installed, setup, executes, managed and monitored.
- Collect and produce measurements, and metrics for internal research and development.
- Improve and enhance knowledge of the product behaviour and technologies in use.

Characteristics of Non-functional testing

- Non-functional testing should be measurable, so there is no place for subjective characterization like good, better, best, etc.
- Exact numbers are unlikely to be known at the start of the requirement process
- Important to prioritize the requirements
- Ensure that quality attributes are identified correctly in Software Engineering.

1) Security:

The parameter defines how a system is safeguarded against deliberate and sudden attacks from internal and external sources. This is tested via Security Testing.

2) Reliability:

The extent to which any software system continuously performs the specified functions without failure. This is tested by Reliability Testing

3) Survivability:

The parameter checks that the software system continues to function and recovers itself in case of system failure. This is checked by Recovery Testing

4) Availability:

The parameter determines the degree to which user can depend on the system during its operation. This is checked by Stability Testing.

5) Usability:

The ease with which the user can learn, operate, prepare inputs and outputs through interaction with a system. This is checked by Usability Testing

6) Scalability:

The term refers to the degree in which any software application can expand its processing capacity to meet an increase in demand. This is tested by Scalability Testing

7) Interoperability:

This non-functional parameter checks a software system interfaces with other software systems. This is checked by Interoperability Testing

8) Efficiency:

The extent to which any software system can handles capacity, quantity and response time.

9) Flexibility:

The term refers to the ease with which the application can work in different hardware and software configurations. Like minimum RAM, CPU requirements.

10) Portability:

The flexibility of software to transfer from its current hardware or software environment.

11) Reusability:

It refers to a portion of the software system that can be converted for use in another application.

Functional Vs Non-Functional Testing:

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	Non-functional testing describes how good the product works
Easy to do Manual Testing	Tough to do Manual Testing
Examples of Functional testing are • Unit Testing • Smoke Testing • Sanity Testing	Examples of Non-functional testing are • Performance Testing • Load Testing

• Integration Testing • White box testing • Black Box testing • User Acceptance testing • Regression Testing	• Volume Testing • Stress Testing • Security Testing • Installation Testing • Penetration Testing • Compatibility Testing • Migration Testing

Functional Testing

- Unit Testing
- Integration Testing
- Smoke
- UAT (User Acceptance Testing)
- Localization
- Globalization
- Interoperability

Non-Functional Testing

- Performance
- Endurance
- Load
- Volume

- Scalability
- Usability

Maintenance

- Regression
- Maintenance

Unit Testing: This software testing basic approach is followed by the programmer to test the unit of the program. It helps developers to know whether the individual unit of the code is working properly or not.

Integration testing: It focuses on the construction and design of the software. You need to see that the integrated units are working without errors or not.

System testing: In this method, your software is compiled as a whole and then tested as a whole. This testing strategy checks the functionality, security, portability, amongst others.

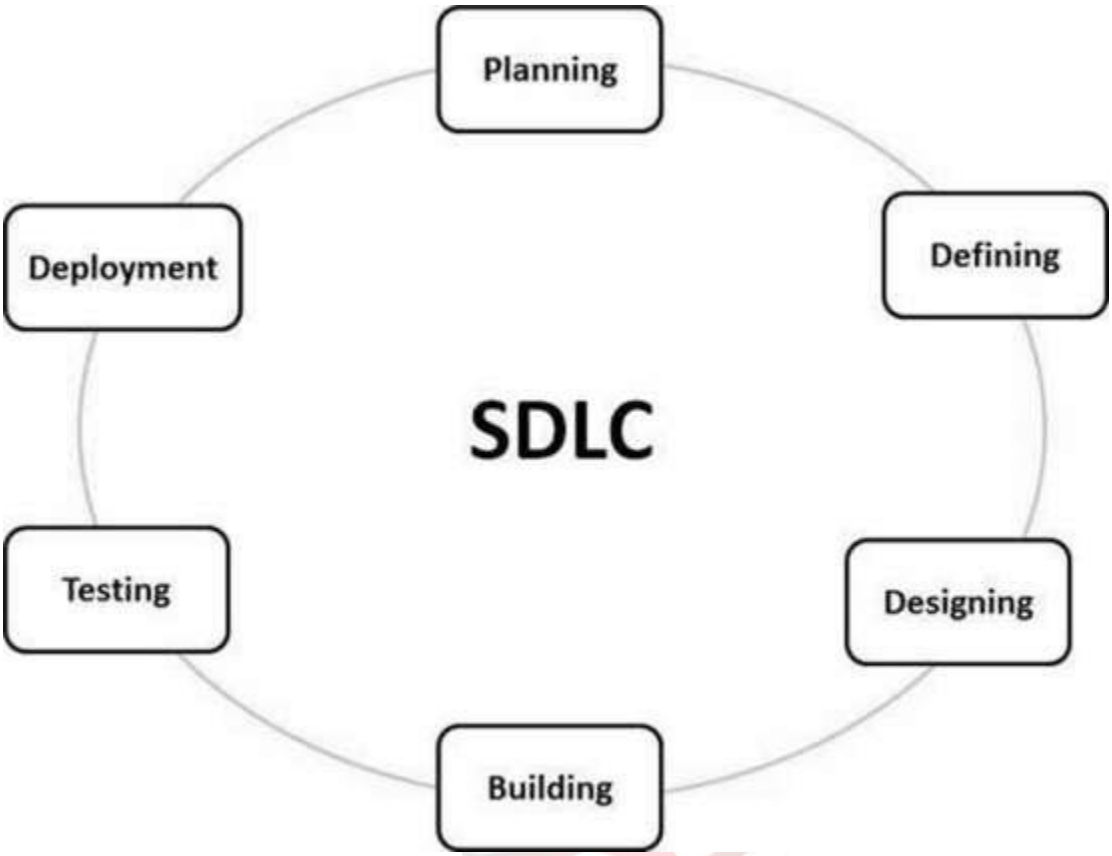
User Acceptance Testing (UAT) involves running a product through a series of specific tests which determines whether the product will meet the needs of its users

SDLC

Software Development life cycle (SDLC) is a spiritual model used in project management that defines the stages include in an information system development project, from an initial feasibility study to the maintenance of the completed application.

There are different software development life cycle models specify and design, which are followed during the software development phase. These models are also called "Software Development Process Models." Each process model follows a series of phase unique to its type to ensure success in the step of software development.

Here, are some important phases of SDLC life cycle:



A typical Software Development Life Cycle consists of the following stages −

Stage 1: Planning and Requirement Analysis

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

Planning for the quality assurance requirements and identification of the risks associated with the project is also done in the planning stage. The outcome of the

technical feasibility study is to define the various technical approaches that can be followed to implement the project successfully with minimum risks.

Stage 2: Defining Requirements

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

Stage 3: Designing the Product Architecture

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

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

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

Stage 4: Building or Developing the Product

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

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

Stage 5: Testing the Product

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

Stage 6: Deployment in the Market and Maintenance

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

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

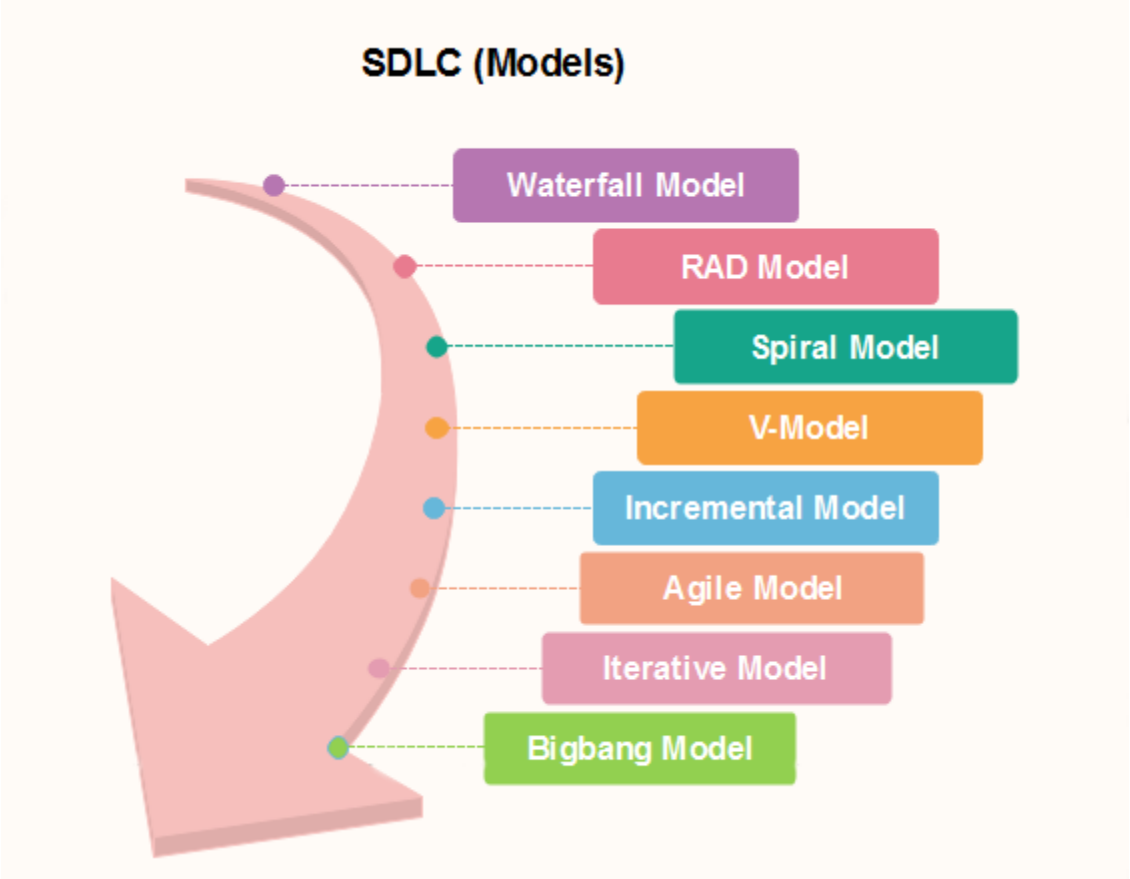
SDLC Models

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

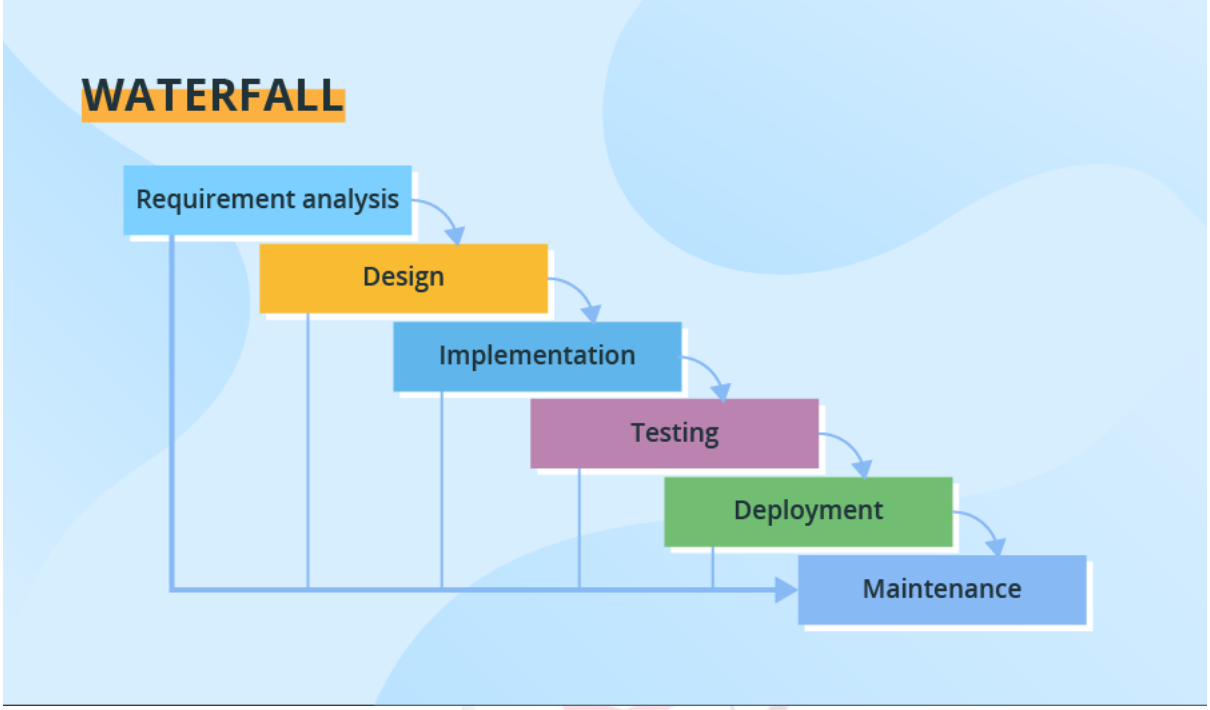
Following are the most important and popular SDLC models followed in the industry −

- Waterfall Model
- Iterative Model
- Spiral Model
- V-Model
- Big Bang Model

Other related methodologies are Agile Model, RAD Model, Rapid Application Development and Prototyping Models.


SDLC (Models)
Waterfall Model
RAD Model
Spiral Model
V-Model
Incremental Model
Agile Model
Iterative Model
Bigbang Model

Waterfall model



Winston Royce introduced the Waterfall Model in 1970. This model has five phases: Requirements analysis and specification, design, implementation, and unit testing, integration and system testing, and operation and maintenance. The steps always follow in this order and do not overlap. The developer must complete every phase before the next phase begins. This model is named "Waterfall Model", because its diagrammatic representation resembles a cascade of waterfalls.

1. Requirements analysis and specification phase: The aim of this phase is to understand the exact requirements of the customer and to document them properly. Both the customer and the software developer work together so as to document all the functions, performance, and interfacing requirement of the

software. It describes the "what" of the system to be produced and not "how. "In this phase, a large document called Software Requirement Specification (SRS) document is created which contained a detailed description of what the system will do in the common language.

2. Design Phase: This phase aims to transform the requirements gathered in the SRS into a suitable form which permits further coding in a programming language. It defines the overall software architecture together with high level and detailed design. All this work is documented as a Software Design Document (SDD).

3. Implementation and unit testing: During this phase, design is implemented. If the SDD is complete, the implementation or coding phase proceeds smoothly, because all the information needed by software developers is contained in the SDD.

During testing, the code is thoroughly examined and modified. Small modules are tested in isolation initially. After that these modules are tested by writing some overhead code to check the interaction between these modules and the flow of intermediate output.

4. Integration and System Testing: This phase is highly crucial as the quality of the end product is determined by the effectiveness of the testing carried out. The better output will lead to satisfied customers, lower maintenance costs, and accurate results. Unit testing determines the efficiency of individual modules. However, in this phase, the modules are tested for their interactions with each other and with the system.

5. Operation and maintenance phase: Maintenance is the task performed by every user once the software has been delivered to the customer, installed, and operational.

When to use SDLC Waterfall Model?

Some Circumstances where the use of the Waterfall model is most suited are:

When the requirements are constant and not changed regularly.

A project is short

The situation is calm

Where the tools and technology used is consistent and is not changing

When resources are well prepared and are available to use.

Advantages of Waterfall model

This model is simple to implement also the number of resources that are required for it is minimal.

The requirements are simple and explicitly declared; they remain unchanged during the entire project development.

The start and end points for each phase is fixed, which makes it easy to cover progress.

The release date for the complete product, as well as its final cost, can be determined before development.

It gives easy to control and clarity for the customer due to a strict reporting system.

Disadvantages of Waterfall model

In this model, the risk factor is higher, so this model is not suitable for more significant and complex projects.

This model cannot accept the changes in requirements during development.

It becomes tough to go back to the phase. For example, if the application has now shifted to the coding phase, and there is a change in requirement, It becomes tough to go back and change it.

Since the testing done at a later stage, it does not allow identifying the challenges and risks in the earlier phase, so the risk reduction strategy is difficult to prepare.

RAD (Rapid Application Development) Model

RAD is a linear sequential software development process model that emphasizes a concise development cycle using an element-based construction approach. If the requirements are well understood and described, and the project scope is a constraint, the RAD process enables a development team to create a fully functional system within a concise time period.

RAD (Rapid Application Development) is a concept that products can be developed faster and of higher quality through:

Gathering requirements using workshops or focus groups

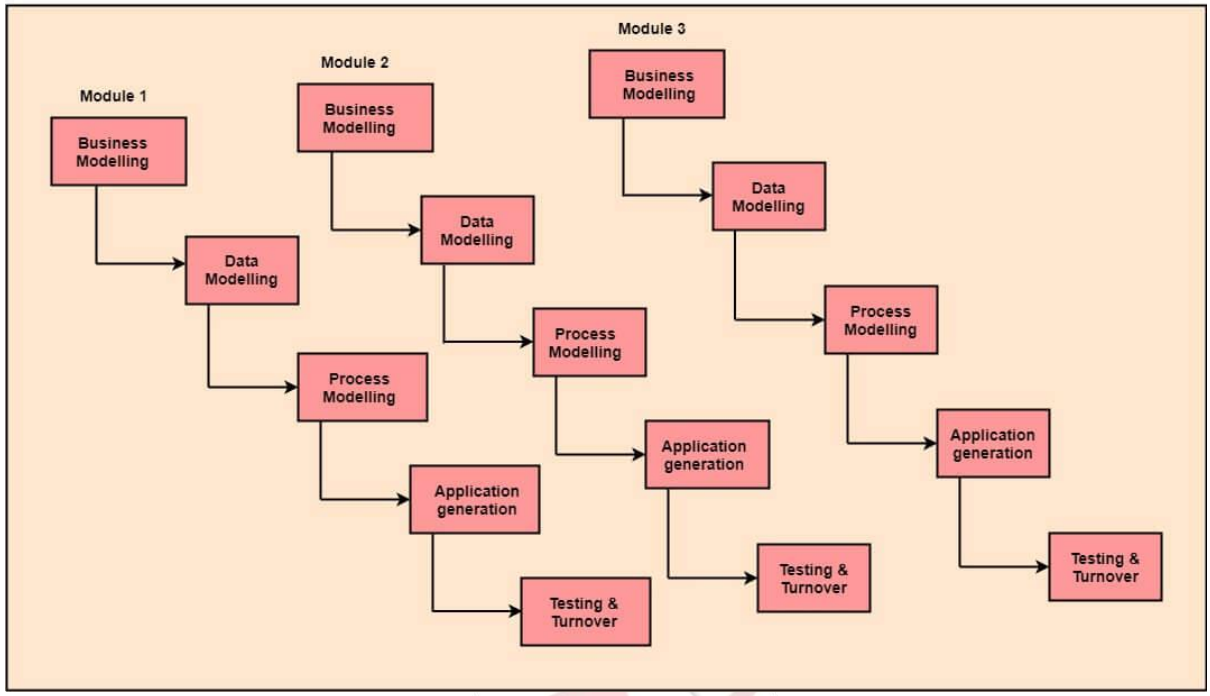
Prototyping and early, reiterative user testing of designs

The re-use of software components

A rigidly paced schedule that refers design improvements to the next product version

Less formality in reviews and other team communication

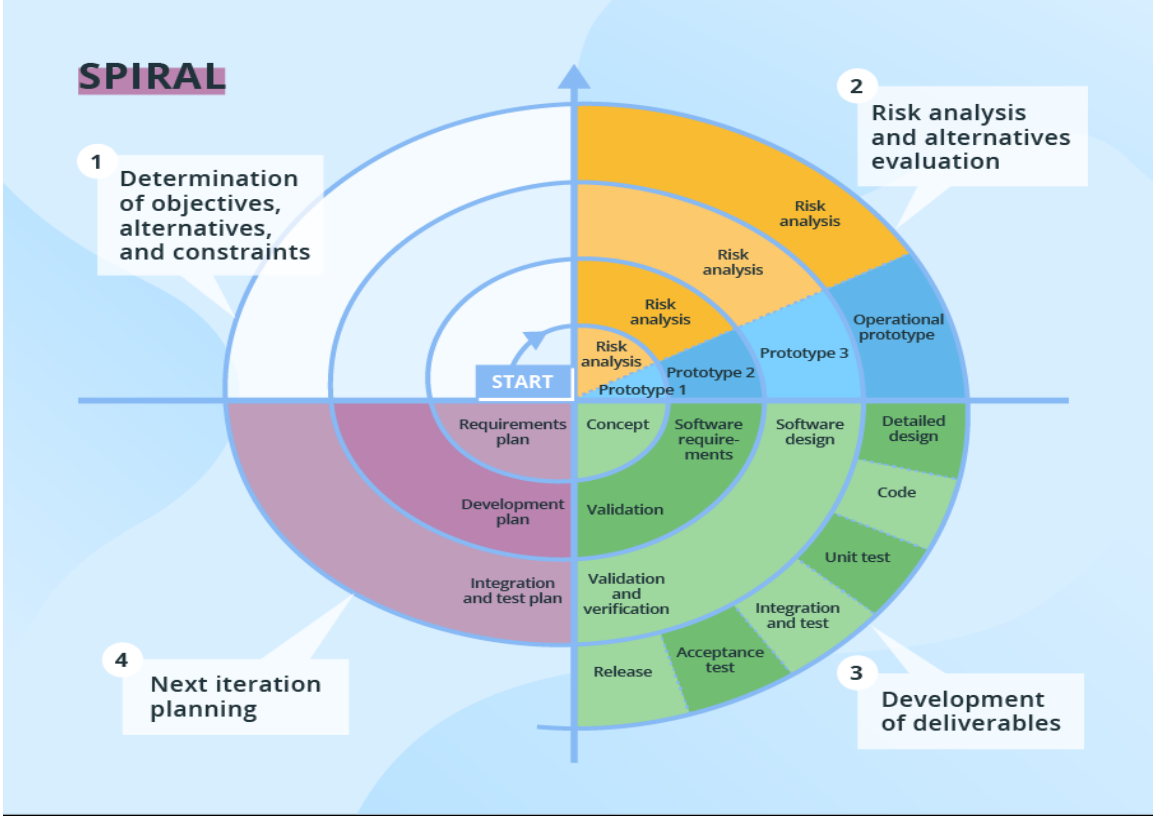
Fig: RAD Model



Spiral Model

The spiral model, initially proposed by Boehm, is an evolutionary software process model that couples the iterative feature of prototyping with the controlled and systematic aspects of the linear sequential model. It implements the potential for rapid development of new versions of the software. Using the spiral model, the software is developed in a series of incremental releases. During the early iterations, the additional release may be a paper model or prototype. During later iterations, more and more complete versions of the engineered system are produced.

The Spiral Model is shown in fig:



Each cycle in the spiral is divided into four parts:

Objective setting: Each cycle in the spiral starts with the identification of purpose for that cycle, the various alternatives that are possible for achieving the targets, and the constraints that exists.

Risk Assessment and reduction: The next phase in the cycle is to calculate these various alternatives based on the goals and constraints. The focus of evaluation in this stage is located on the risk perception for the project.

Development and validation: The next phase is to develop strategies that resolve uncertainties and risks. This process may include activities such as benchmarking, simulation, and prototyping.

Planning: Finally, the next step is planned. The project is reviewed, and a choice made whether to continue with a further period of the spiral. If it is determined to keep, plans are drawn up for the next step of the project.

The development phase depends on the remaining risks. For example, if performance or user-interface risks are treated more essential than the program development risks, the next phase may be an evolutionary development that includes developing a more detailed prototype for solving the risks.

The **risk-driven** feature of the spiral model allows it to accommodate any mixture of a specification-oriented, prototype-oriented, simulation-oriented, or another type of approach. An essential element of the model is that each period of the spiral is completed by a review that includes all the products developed during that cycle,

including plans for the next cycle. The spiral model works for development as well as enhancement projects.

When to use Spiral Model?

- When deliverance is required to be frequent.
- When the project is large
- When requirements are unclear and complex
- When changes may require at any time
- Large and high budget projects

Advantages

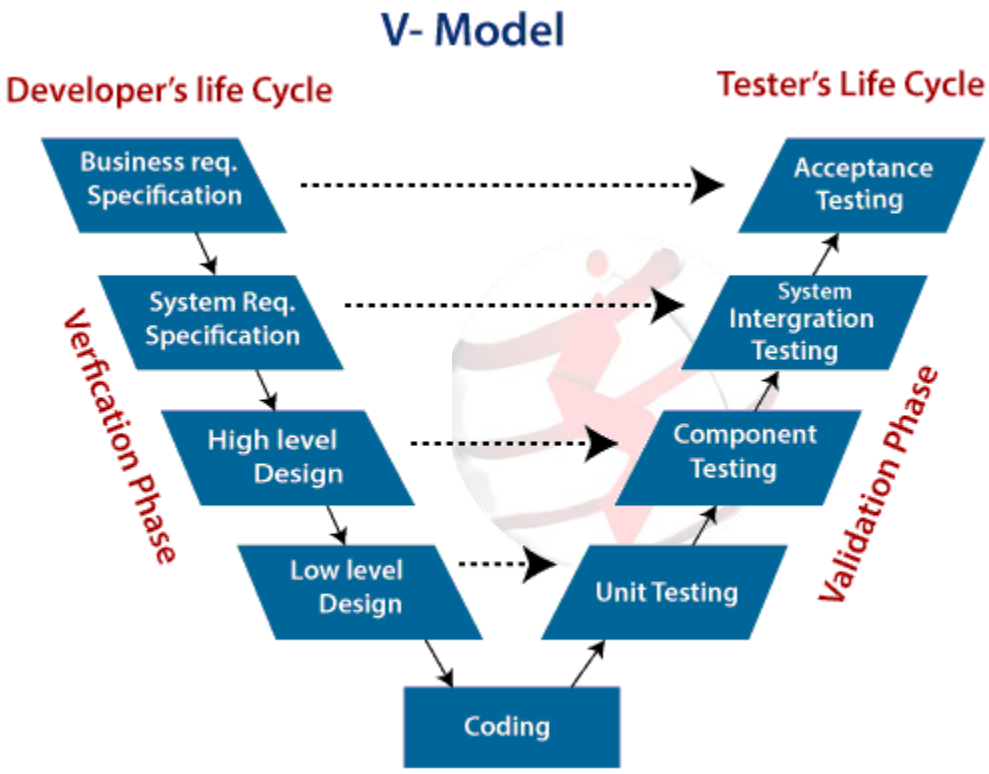
- High amount of risk analysis
- Useful for large and mission-critical projects.

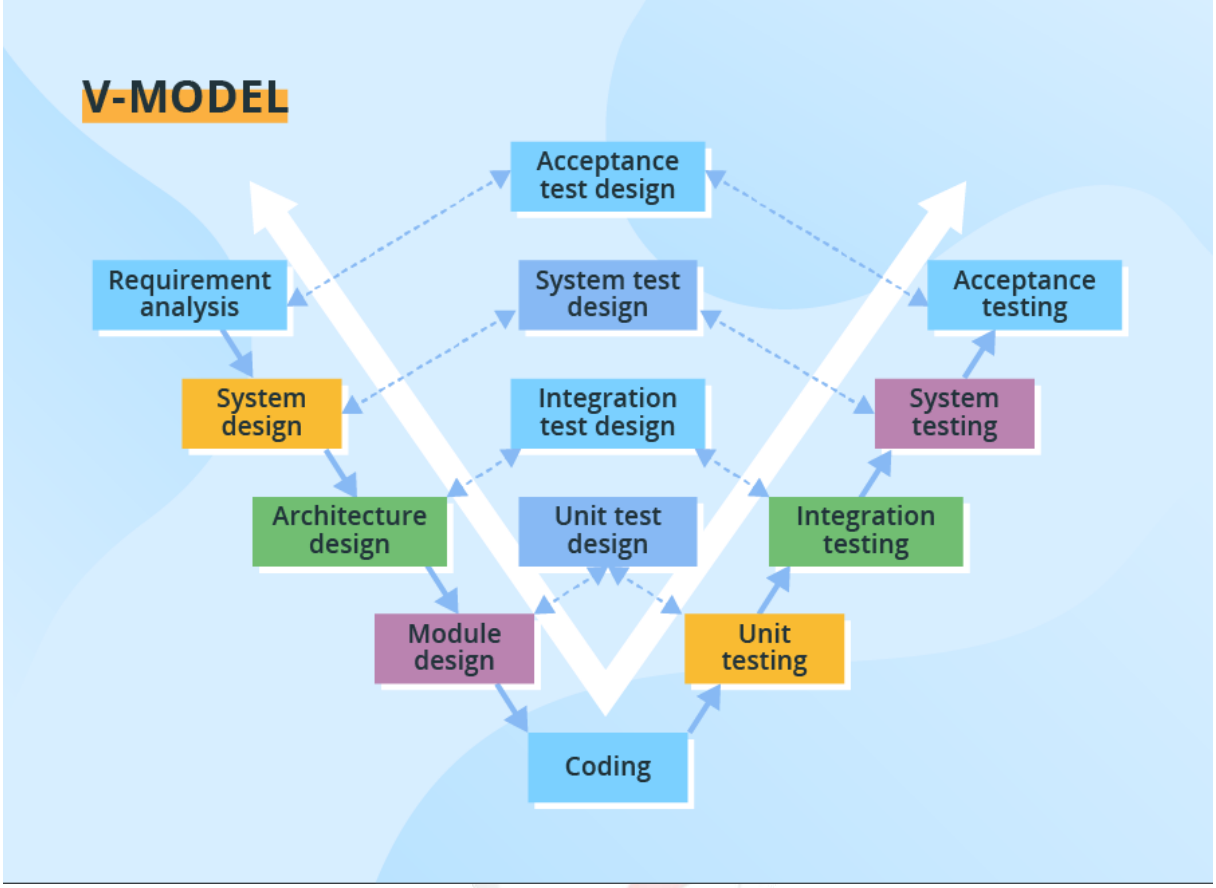
Disadvantages

- Can be a costly model to use.
- Risk analysis needed highly particular expertise
- Doesn't work well for smaller projects.

V-Model

V-Model also referred to as the Verification and Validation Model. In this, each phase of SDLC must complete before the next phase starts. It follows a sequential design process same as the waterfall model. Testing of the device is planned in parallel with a corresponding stage of development.





Verification: It involves a static analysis method (review) done without executing code. It is the process of evaluation of the product development process to find whether specified requirements meet.

Validation: It involves dynamic analysis method (functional, non-functional), testing is done by executing code. Validation is the process to classify the software after the completion of the development process to determine whether the software meets the customer expectations and requirements.

So V-Model contains Verification phases on one side of the Validation phases on the other side. Verification and Validation process is joined by coding phase in V-shape. Thus it is known as V-Model.

There are the various phases of Verification Phase of V-model:

1. **Business requirement analysis:** This is the first step where product requirements understood from the customer's side. This phase contains detailed communication to understand customer's expectations and exact requirements.
2. **System Design:** In this stage system engineers analyze and interpret the business of the proposed system by studying the user requirements document.
3. **Architecture Design:** The baseline in selecting the architecture is that it should understand all which typically consists of the list of modules, brief functionality of each module, their interface relationships, dependencies, database tables, architecture diagrams, technology detail, etc. The integration testing model is carried out in a particular phase.
4. **Module Design:** In the module design phase, the system breaks down into small modules. The detailed design of the modules is specified, which is known as Low-Level Design
5. **Coding Phase:** After designing, the coding phase is started. Based on the requirements, a suitable programming language is decided. There are some guidelines and standards for coding. Before checking in the repository, the

final build is optimized for better performance, and the code goes through many code reviews to check the performance.

There are the various phases of Validation Phase of V-model:

1. **Unit Testing:** In the V-Model, Unit Test Plans (UTPs) are developed during the module design phase. These UTPs are executed to eliminate errors at code level or unit level. A unit is the smallest entity which can independently exist, e.g., a program module. Unit testing verifies that the smallest entity can function correctly when isolated from the rest of the codes/ units.
2. **Integration Testing:** Integration Test Plans are developed during the Architectural Design Phase. These tests verify that groups created and tested independently can coexist and communicate among themselves.
3. **System Testing:** System Tests Plans are developed during System Design Phase. Unlike Unit and Integration Test Plans, System Tests Plans are composed by the client?s business team. System Test ensures that expectations from an application developer are met.
4. **Acceptance Testing:** Acceptance testing is related to the business requirement analysis part. It includes testing the software product in user atmosphere. Acceptance tests reveal the compatibility problems with the different systems, which is available within the user atmosphere. It conjointly discovers the non-functional problems like load and performance defects within the real user atmosphere.

When to use V-Model?

- When the requirement is well defined and not ambiguous.
- The V-shaped model should be used for small to medium-sized projects where requirements are clearly defined and fixed.
- The V-shaped model should be chosen when sample technical resources are available with essential technical expertise.

Advantage (Pros) of V-Model:

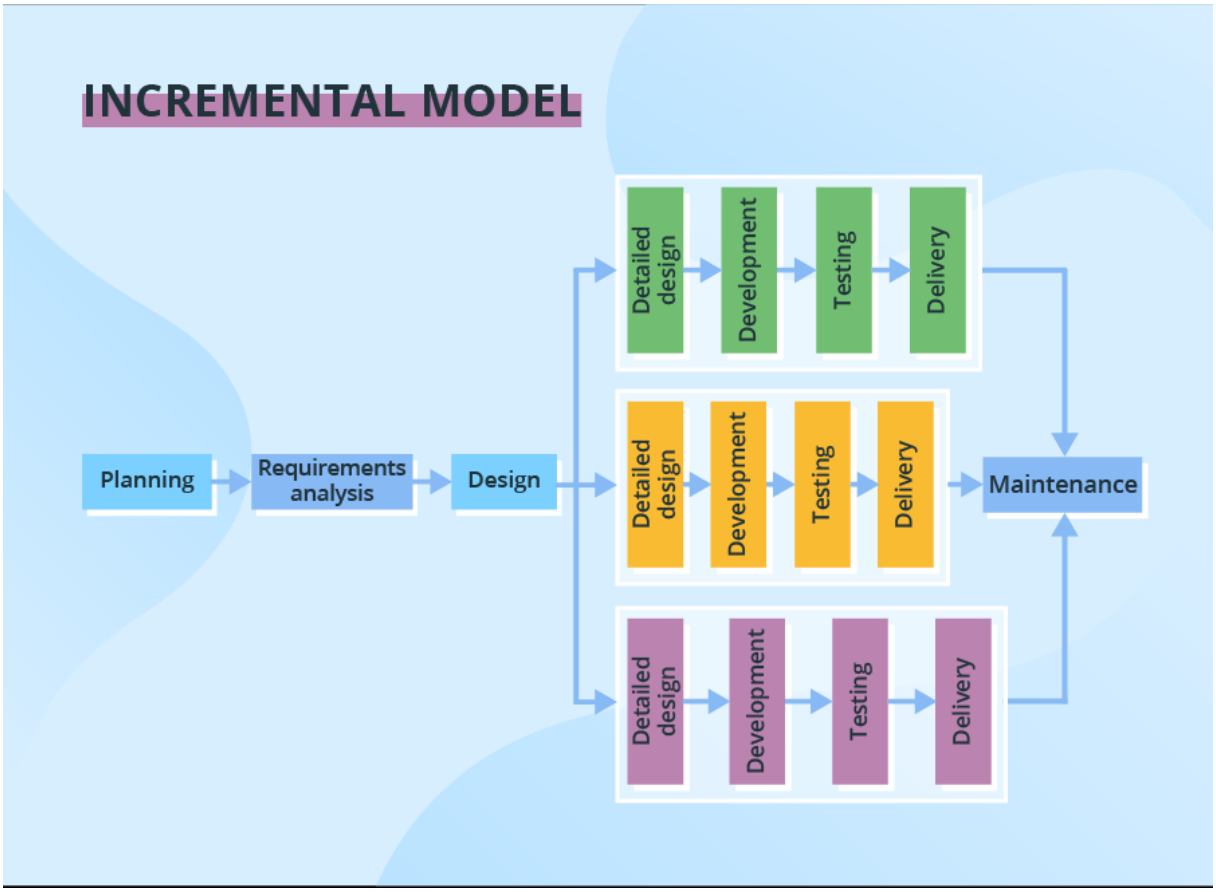
1. Easy to Understand.
2. Testing Methods like planning, test designing happens well before coding.
3. This saves a lot of time. Hence a higher chance of success over the waterfall model.
4. Avoids the downward flow of the defects.
5. Works well for small plans where requirements are easily understood.

Disadvantage (Cons) of V-Model:

1. Very rigid and least flexible.
2. Not a good for a complex project.
3. Software is developed during the implementation stage, so no early prototypes of the software are produced.
4. If any changes happen in the midway, then the test documents along with the required documents, has to be updated.

Incremental Model

Incremental Model is a process of software development where requirements divided into multiple standalone modules of the software development cycle. In this model, each module goes through the requirements, design, implementation and testing phases. Every subsequent release of the module adds function to the previous release. The process continues until the complete system achieved.



The various phases of incremental model are as follows:

1. Requirement analysis: In the first phase of the incremental model, the product analysis expertise identifies the requirements. And the system functional

requirements are understood by the requirement analysis team. To develop the software under the incremental model, this phase performs a crucial role.

2. Design & Development: In this phase of the Incremental model of SDLC, the design of the system functionality and the development method are finished with success. When software develops new practicality, the incremental model uses style and development phase.

3. Testing: In the incremental model, the testing phase checks the performance of each existing function as well as additional functionality. In the testing phase, the various methods are used to test the behavior of each task.

4. Implementation: Implementation phase enables the coding phase of the development system. It involves the final coding that design in the designing and development phase and tests the functionality in the testing phase. After completion of this phase, the number of the product working is enhanced and upgraded up to the final system product

When we use the Incremental Model?

- When the requirements are superior.
- A project has a lengthy development schedule.
- When Software team are not very well skilled or trained.
- When the customer demands a quick release of the product.
- You can develop prioritized requirements first.

Advantage of Incremental Model

- Errors are easy to be recognized.
- Easier to test and debug
- More flexible.
- Simple to manage risk because it handled during its iteration.
- The Client gets important functionality early.

Disadvantage of Incremental Model

- Need for good planning
- Total Cost is high.
- Well defined module interfaces are needed.

Agile Model

The meaning of Agile is swift or versatile. "**Agile process model**" refers to a software development approach based on iterative development. Agile methods break tasks into smaller iterations, or parts do not directly involve long term planning. The project scope and requirements are laid down at the beginning of the development process. Plans regarding the number of iterations, the duration and the scope of each iteration are clearly defined in advance.

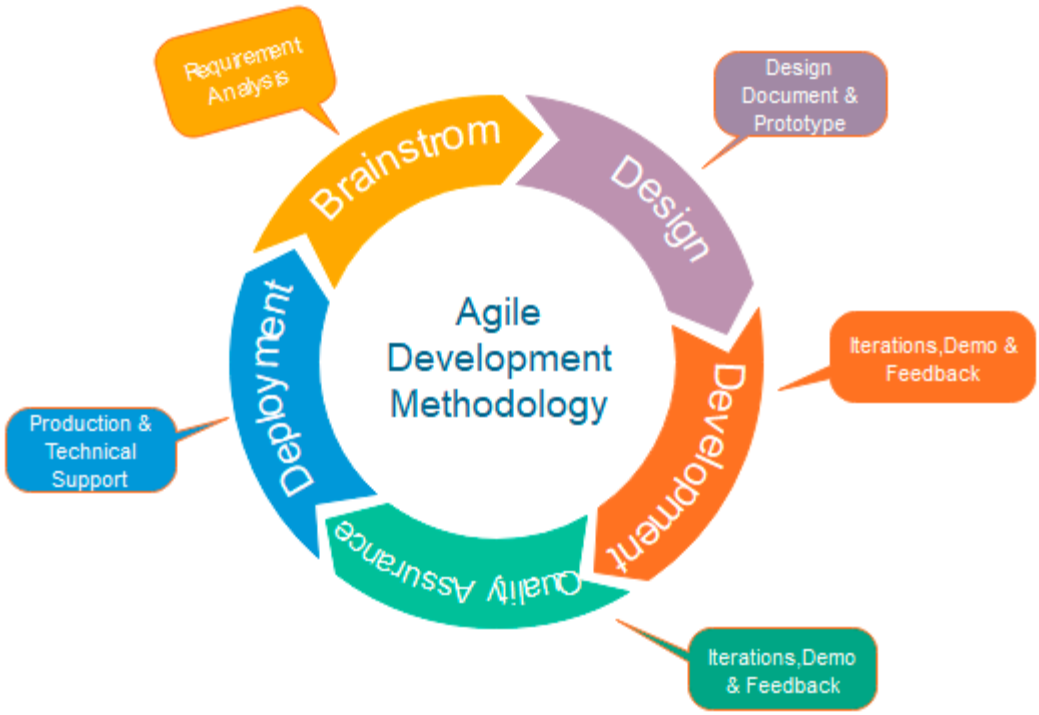


Fig. Agile Model

Each iteration is considered as a short time "frame" in the Agile process model, which typically lasts from one to four weeks. The division of the entire project into smaller parts helps to minimize the project risk and to reduce the overall project delivery

time requirements. Each iteration involves a team working through a full software development life cycle including planning, requirements analysis, design, coding, and testing before a working product is demonstrated to the client.

Phases of Agile Model:

Following are the phases in the Agile model are as follows:

1. Requirements gathering
2. Design the requirements
3. Construction/ iteration
4. Testing/ Quality assurance
5. Deployment
6. Feedback

1. Requirements gathering: In this phase, you must define the requirements. You should explain business opportunities and plan the time and effort needed to build the project. Based on this information, you can evaluate technical and economic feasibility.

2. Design the requirements: When you have identified the project, work with stakeholders to define requirements. You can use the user flow diagram or the high-level UML diagram to show the work of new features and show how it will apply to your existing system.

3. Construction/ iteration: When the team defines the requirements, the work begins. Designers and developers start working on their project, which aims to

deploy a working product. The product will undergo various stages of improvement, so it includes simple, minimal functionality.

4. Testing: In this phase, the Quality Assurance team examines the product's performance and looks for the bug.

5. Deployment: In this phase, the team issues a product for the user's work environment.

6. Feedback: After releasing the product, the last step is feedback. In this, the team receives feedback about the product and works through the feedback.

Agile Testing Methods:

- Scrum
- Crystal
- Dynamic Software Development Method(DSDM)
- Feature Driven Development(FDD)
- Lean Software Development
- eXtreme Programming(XP)

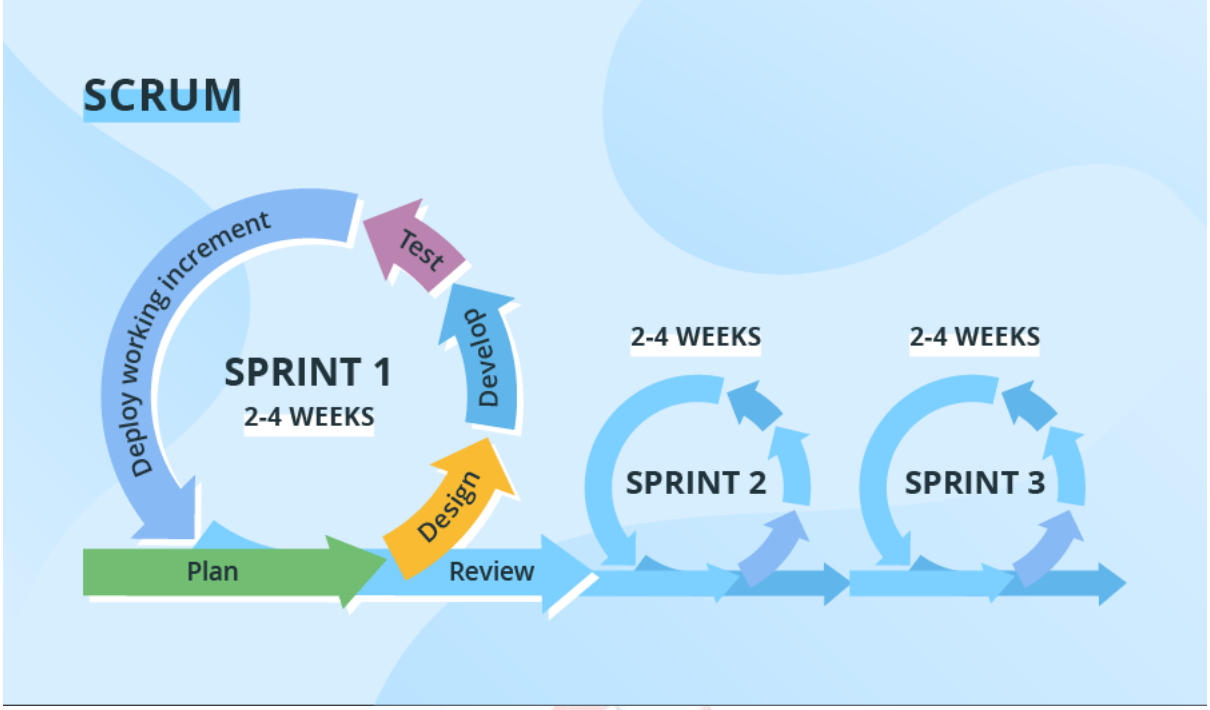
Agile comes in different flavour's. Today, it's most common subtypes are Scrum, Extreme Programming, and Kanban.

Scrum

SCRUM is an agile development process focused primarily on ways to manage tasks in team-based development conditions.

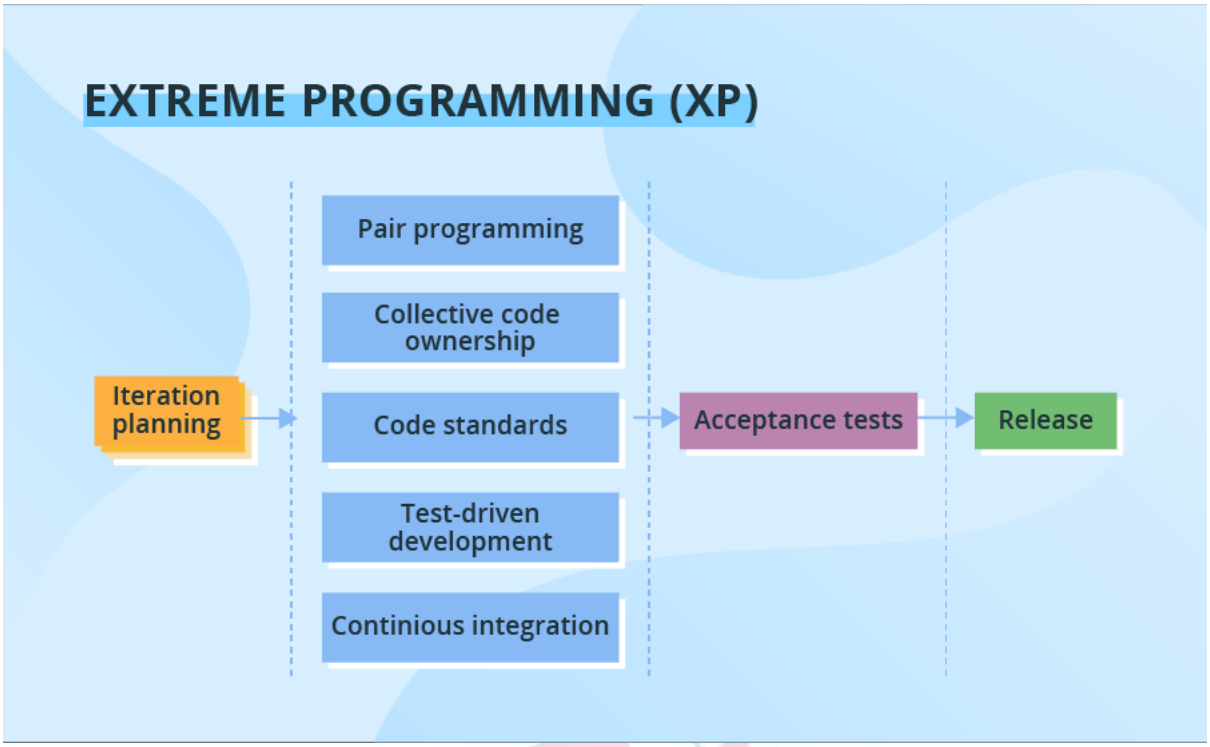
There are three roles in it, and their responsibilities are:

- **Scrum Master:** The scrum can set up the master team, arrange the meeting and remove obstacles for the process
- **Product owner:** The product owner makes the product backlog, prioritizes the delay and is responsible for the distribution of functionality on each repetition.
- **Scrum Team:** The team manages its work and organizes the work to complete the sprint or cycle.



eXtreme Programming(XP)

This type of methodology is used when customers are constantly changing demands or requirements, or when they are not sure about the system's performance.



Crystal:

There are three concepts of this method-

1. Chartering: Multi activities are involved in this phase such as making a development team, performing feasibility analysis, developing plans, etc.
2. Cyclic delivery: under this, two more cycles consist, these are:
    - Team updates the release plan.
    - Integrated product delivers to the users.
3. Wrap up: According to the user environment, this phase performs deployment, post-deployment.

Dynamic Software Development Method (DSDM):

DSDM is a rapid application development strategy for software development and gives an agile project distribution structure. The essential features of DSDM are that users must be actively connected, and teams have been given the right to make decisions. The techniques used in DSDM are:

1. Time Boxing : is the approach for completing the project incrementally by breaking it down into splitting the project in portions, each with a fixed budget and a delivery date
2. MoSCoW Rules

   is a technique for prioritising work items or requirements. It is an acronym that stands for:

- Must have
- Should have
- Could have
- Won't have

3. Prototyping

   refers to the creation of prototypes of the system under development at an early stage of the project

The DSDM project contains seven stages:

1. Pre-project
2. Feasibility Study
3. Business Study
4. Functional Model Iteration
5. Design and build Iteration
6. Implementation
7. Post-project

Feature Driven Development (FDD):

This method focuses on "Designing and Building" features. In contrast to other smart methods, FDD describes the small steps of the work that should be obtained separately per function.

Lean Software Development:

Lean software development methodology follows the principle "just in time production." The lean method indicates the increasing speed of software development and reducing costs. Lean development can be summarized in seven phases.

1. Eliminating Waste
2. Amplifying learning
3. Defer commitment (deciding as late as possible)
4. Early delivery
5. Empowering the team

6. Building Integrity
7. Optimize the whole

When to use the Agile Model?

- When frequent changes are required.
- When a highly qualified and experienced team is available.
- When a customer is ready to have a meeting with a software team all the time.
- When project size is small.

Advantage(Pros) of Agile Method:

1. Frequent Delivery
2. Face-to-Face Communication with clients.
3. Efficient design and fulfils the business requirement.
4. Anytime changes are acceptable.
5. It reduces total development time.

Disadvantages(Cons) of Agile Model:

1. Due to the shortage of formal documents, it creates confusion and crucial decisions taken throughout various phases can be misinterpreted at any time by different team members.

2. Due to the lack of proper documentation, once the project completes and the developers allotted to another project, maintenance of the finished project can become a difficulty.

Iterative Model

In this Model, you can start with some of the software specifications and develop the first version of the software. After the first version if there is a need to change the software, then a new version of the software is created with a new iteration. Every release of the Iterative Model finishes in an exact and fixed period that is called iteration.

The Iterative Model allows the accessing earlier phases, in which the variations made respectively. The final output of the project renewed at the end of the Software Development Life Cycle (SDLC) process.

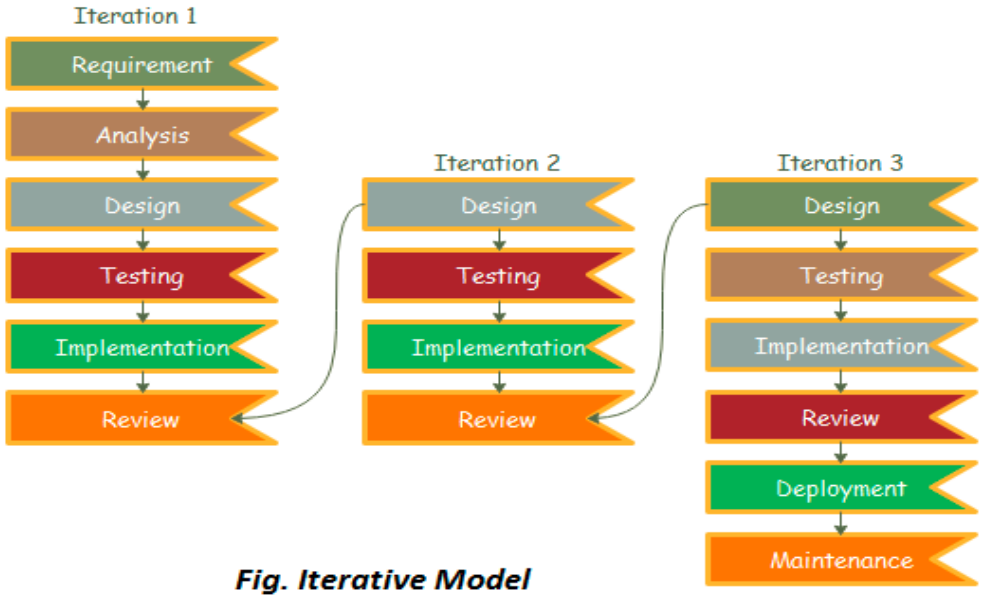


Fig. Iterative Model

Big Bang Model

In this model, developers do not follow any specific process. Development begins with the necessary funds and efforts in the form of inputs. And the result may or may not be as per the customer's requirement, because in this model, even the customer requirements are not defined.

This model is ideal for small projects like academic projects or practical projects. One or two developers can work together on this model.

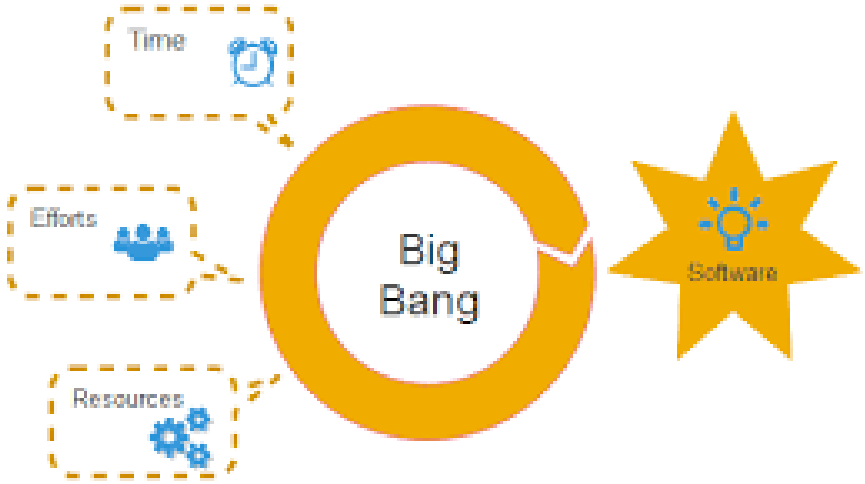


Fig. Big Bang Model

When to use Big Bang Model?

As we discussed above, this model is required when this project is small like an academic project or a practical project. This method is also used when the size of the developer team is small and when requirements are not defined, and the release date is not confirmed or given by the customer.

Advantage (Pros) of Big Bang Model:

1. There is no planning required.
2. Simple Model.
3. Few resources required.
4. Easy to manage.
5. Flexible for developers.

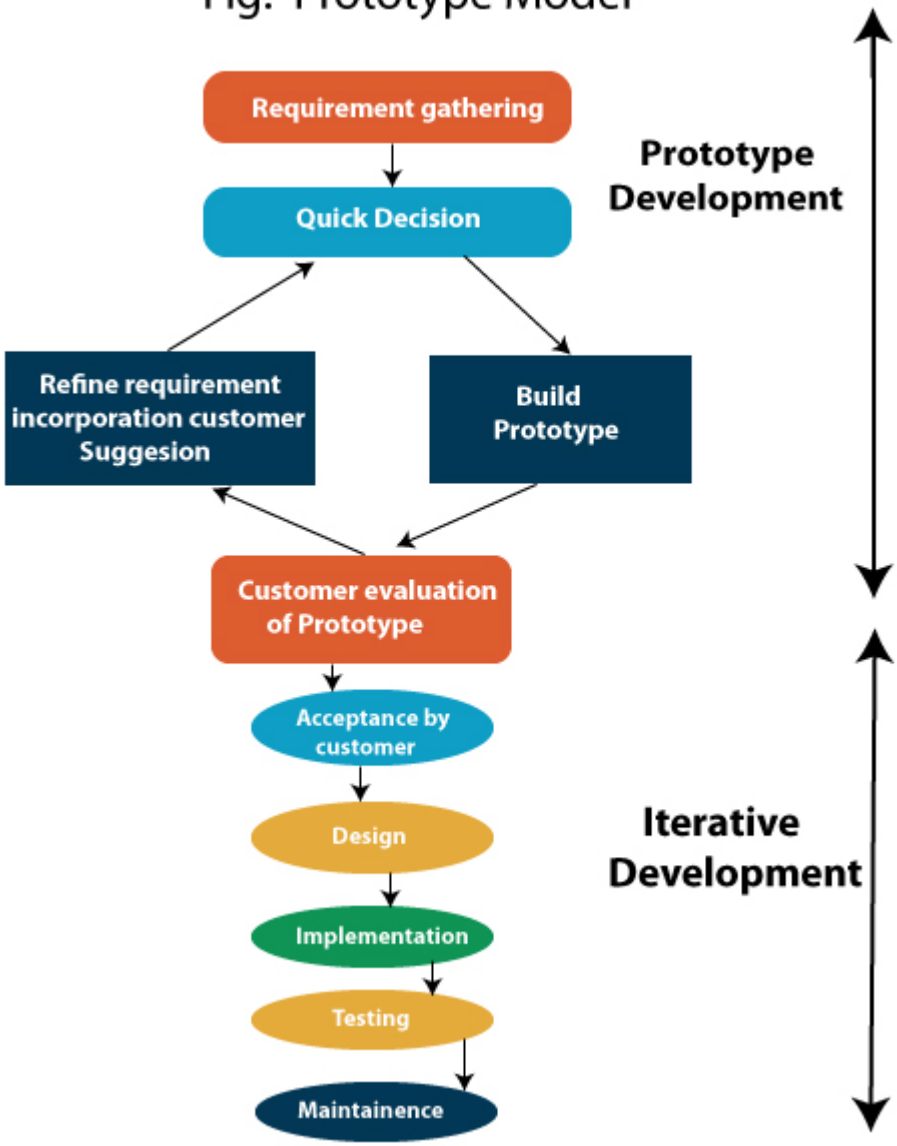
Disadvantage (Cons) of Big Bang Model:

1. There are high risk and uncertainty.
2. Not acceptable for a large project.
3. If requirements are not clear that can cause very expensive.

Prototype Model

The prototype model requires that before carrying out the development of actual software, a working prototype of the system should be built. A prototype is a toy implementation of the system. A prototype usually turns out to be a very crude version of the actual system, possible exhibiting limited functional capabilities, low reliability, and inefficient performance as compared to actual software. In many instances, the client only has a general view of what is expected from the software product. In such a scenario where there is an absence of detailed information regarding the input to the system, the processing needs, and the output requirement, the prototyping model may be employed.

Fig: Prototype Model



Steps of Prototype Model

1. Requirement Gathering and Analyst
2. Quick Decision
3. Build a Prototype
4. Assessment or User Evaluation

5. Prototype Refinement
6. Engineer Product

Advantage of Prototype Model

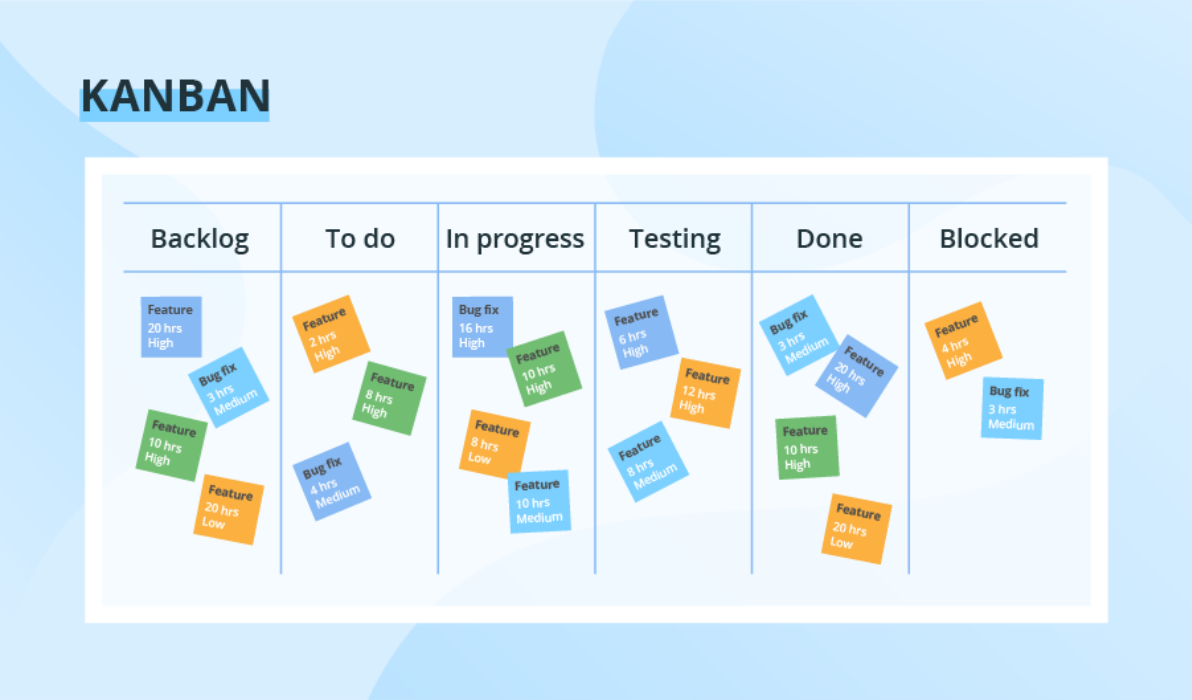
1. Reduce the risk of incorrect user requirement
2. Good where requirement are changing/uncommitted
3. Regular visible process aids management
4. Support early product marketing
5. Reduce Maintenance cost.
6. Errors can be detected much earlier as the system is made side by side.

Disadvantage of Prototype Model

1. An unstable/badly implemented prototype often becomes the final product.
2. Require extensive customer collaboration
    - Costs customer money
    - Needs committed customer
    - Difficult to finish if customer withdraw
    - May be too customer specific, no broad market
3. Difficult to know how long the project will last.
4. Easy to fall back into the code and fix without proper requirement analysis, design, customer evaluation, and feedback.
5. Prototyping tools are expensive.

6. Special tools & techniques are required to build a prototype.
7. It is a time-consuming process.

Kanban



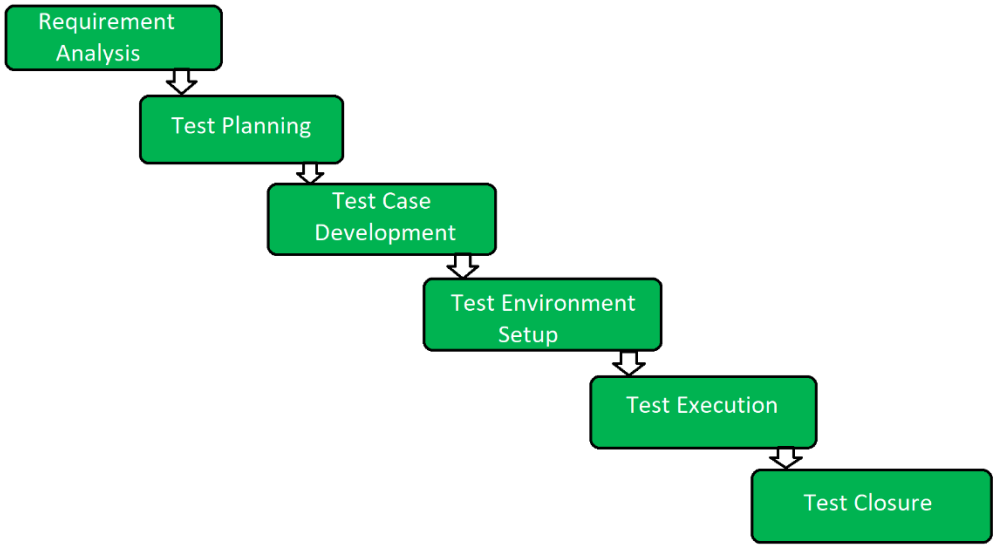
As for **Kanban,** its key distinguishing feature is the absence of pronounced iterations. If used, they are kept extremely short (‘daily sprints’). Instead, the emphasis is placed on plan visualization. The team uses the **Kanban Board** tool that provides a clear representation of all project activities, their number, responsible persons, and progress. Such increased transparency helps to estimate the most urgent tasks more accurately. Also, the model has no separate planning stage, so a new change request can be introduced at any time. Communication with the customer is ongoing, they can check the work results whenever they like, and the meetings with the project team can happen even daily. Due to its nature, the model is frequently used in *projects on software support and evolution*.

What is Software Testing Life Cycle (STLC)?

Software Testing Life Cycle (STLC) is a sequence of specific activities conducted during the testing process to ensure software quality goals are met. STLC involves both verification and validation activities. Contrary to popular belief, Software Testing is not just a single/isolate activity, i.e. testing. It consists of a series of activities carried out methodologically to help certify your software product. STLC stands for Software Testing Life Cycle.

STLC Phases

There are following six major phases in every Software Testing Life Cycle Model (STLC Model):



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

You have Entry and Exit Criteria for all levels in the Software Testing Life Cycle (STLC) 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. So for this tutorial, we will focus on activities and deliverables for the different stages in STLC life cycle. Let's look into them in detail.

1. **Requirement Analysis:**

   Requirement Analysis is the first step of Software Testing Life Cycle (STLC). In this phase quality assurance team understands the requirements like what is to be tested. If anything is missing or not understandable then quality assurance team meets with the stakeholders to better understand the detail knowledge of requirement.

2. **Test Planning:**

   Test Planning is most efficient phase of software testing life cycle where all testing plans are defined. In this phase manager of the testing team calculates estimated effort and cost for the testing work. This phase gets started once the requirement gathering phase is completed.

3. **Test Case Development:**

   The test case development phase gets started once the test planning phase is completed. In this phase testing team note down the detailed test cases. Testing team also prepare the required test data for the testing. When the test cases are prepared then they are reviewed by quality assurance team.

4. **Test Environment Setup:**

   Test environment setup is the vital part of the STLC. Basically test environment decides the conditions on which software is tested. This is independent activity and can be started along with test case development. In this process the testing team is not involved. Either the developer or the customer creates the testing environment.

5. **Test Execution:**

   After the test case development and test environment setup test execution phase gets started. In this phase testing team start executing test cases based on prepared test cases in the earlier step.

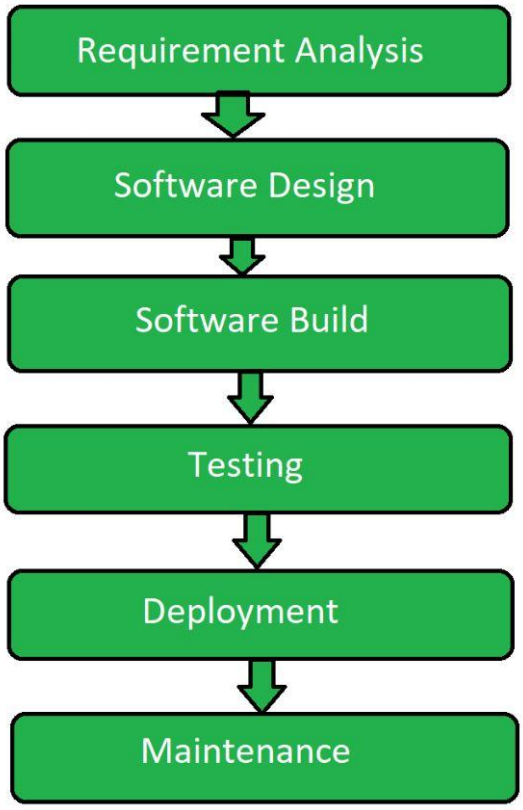
6. **Test Closure:**

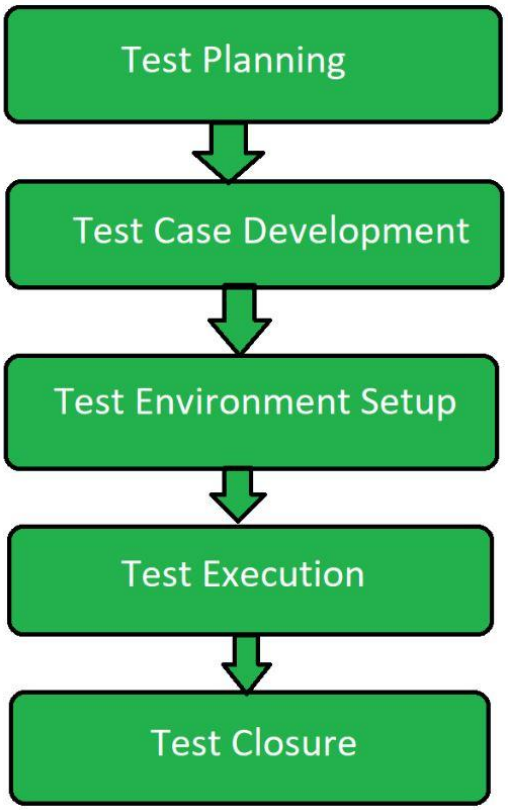
   This is the last stage of STLC in which the process of testing is analyzed.

Difference between SDLC and STLC

Software Development Life Cycle (SDLC) is a sequence of different activities performed during the software development process.

Software Testing Life Cycle (STLC) is a sequence of different activities performed during the software testing process.





Difference between SDLC and STLC:

STLC	SDLC
STLC is mainly related to software testing.	SDLC is mainly related to software development.
It focuses only on testing the software.	Besides development other phases like testing is also included.
STLC involves only five phases or steps.	SDLC involves total six phases or steps.
In STLC, less number of members (testers) are needed.	In SDLC, more number of members (developers) are required for the whole process.
In STLC, testing team(Test Lead or Test Architect) makes the plans and designs.	In SDLC, development team makes the plans and designs based on the requirements.

STLC	SDLC
Goal of STLC is to complete successful testing of software.	Goal of SDLC is to complete successful development of software.
It helps in making the software defects free.	It helps in developing good quality software.
STLC phases are performed after SDLC phases.	SDLC phases are completed before the STLC phases.
Regression tests are run by QA team to check deployed maintenance code and maintains test cases and automated scripts.	Post deployment support , enhancement , and update are to be included if necessary.
A tested software system is the end result of STLC.	Creation of reusable software systems is the end result of SDLC.

What is Static Testing?

Static Testing is a type of software testing in which software application is tested without code execution. Manual or automated reviews of code, requirement documents and document design are done in order to find the errors. The main objective of static testing is to improve the quality of software applications by finding errors in early stages of software development process.

What is Dynamic Testing?

Under **Dynamic Testing**, a code is executed. It checks for functional behavior of software system, memory/cpu usage and overall performance of the system. Hence the name “Dynamic”

The main objective of this testing is to confirm that the software product works in conformance with the business requirements. This testing is also called an Execution technique or validation testing.

Dynamic testing executes the software and validates the output with the expected outcome. Dynamic testing is performed at all levels of testing and it can be either black or white box testing.

What is Traceability Matrix (TM)?

A Traceability Matrix is a document that co-relates any two-baseline documents that require a many-to-many relationship to check the completeness of the relationship.

It is used to track the requirements and to check the current project requirements are met.

What is Requirement Traceability Matrix?

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

Why RTM is Important?

The main agenda of every tester should be to understand the client's requirement and make sure that the output product should be defect-free. To achieve this goal, every QA should understand the requirement thoroughly and create positive and negative test cases.

This would mean that the software requirements provided by the client have to be further split into different scenarios and further to test cases. Each of this case has to be executed individually.

A question arises here on how to make sure that the requirement is tested considering all possible scenarios/cases? How to ensure that any requirement is not left out of the testing cycle?

A simple way is to trace the requirement with its corresponding test scenarios and test cases. This merely is termed as 'Requirement Traceability Matrix.'

The traceability matrix is typically a worksheet that contains the requirements with its all possible test scenarios and cases and their current state, i.e. if they have been passed or failed. This would help the testing team to understand the level of testing activities done for the specific product.

Which Parameters to include in Requirement Traceability Matrix?

- Requirement ID
- Requirement Type and Description
- Test Cases with Status

Req No	Req Desc	Testcase ID	Status
123	Login to the application	TC01,TC02,TC03	TC01-Pass TC02-Pass
345	Ticket Creation	TC04,TC05,TC06, TC07,TC08,TC09 TC010	TC04-Pass TC05-Pass TC06-Pass TC06-Fail TC07-No Run
456	Search Ticket	TC011,TC012, TC013,TC014	TC011-Pass TC012-Fail TC013-Pass TC014-No Run

Above is a sample requirement traceability matrix.

But in a typical software testing project, the traceability matrix would have more than these parameters.

Sno	Req ID	Req Desc	TC ID	TC Desc	Test Design	Test Designer	UAT Test Req?	Test Execution			Defects?	Defect ID	Defect Status	Req Coverage Status
								Test Env	UAT Env	Prod Env				
1	Req01	Login to the Application	TC01	Login with Invalid Username and valid password	Completed	XYZ	No	Passed	No Run	No Run	None	None	N/A	Partial
2			TC02	Login with Valid Username and invalid password	Completed	YZA	No	Passed	No Run	No Run	None	None	N/A	Partial
3			TC03	Login with valid credentials	Completed	XYZ	Yes	Passed	Passed	No Run	Yes	DFCT001	Test OK	Partial

Sheet1 Sheet2 Sheet3

As illustrated above, a requirement traceability matrix can:

- Show the requirement coverage in the number of test cases
- Design status as well as execution status for the specific test case
- If there is any User Acceptance test to be done by the users, then UAT status can also be captured in the same matrix.
- The related defects and the current state can also be mentioned in the same matrix.

This kind of matrix would be providing **One Stop Shop** for all the testing activities.

Apart from maintaining an excel separately. A testing team can also opt for requirements tracing available Test Management Tools.

Difference between Defect, Error, Bug, Failure and Fault!

Testing is the process of identifying defects, where a defect is any variance between actual and expected results. “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.”

DEFECT:

It can be simply defined as a variance between expected and actual. The defect is an error found AFTER the application goes into production. It commonly refers to several troubles with the software products, with their external behavior or with its internal features. In other words, a Defect is a difference between expected and actual results in the context of testing. It is the deviation of the customer requirement.

Defect can be categorized into the following:

Wrong:

When requirements are implemented not in the right way. This defect is a variance from the given specification. It is Wrong!

Missing:

A requirement of the customer that was not fulfilled. This is a variance from the specifications, an indication that a specification was not implemented, or a requirement of the customer was not noted correctly.

Extra:

A requirement incorporated into the product that was not given by the end customer. This is always a variance from the specification, but maybe an attribute desired by the user of the product. However, it is considered a defect because it's a variance from the existing requirements.

ERROR:

An error is a mistake, misconception, or misunderstanding on the part of a software developer. In the category of the developer, we include software engineers, programmers, analysts, and testers. For example, a developer may misunderstand a de-sign notation, or a programmer might type a variable name incorrectly – leads to an Error. It is the one that is generated because of the wrong login, loop or syntax. The error normally arises in software; it leads to a change in the functionality of the program.

BUG:

A bug is the result of a coding error. An Error found in the development environment before the product is shipped to the customer. A programming error that causes a program to work poorly, produce incorrect results or crash. An error in software or hardware that causes a program to malfunction. A bug is the terminology of Tester.

FAILURE:

A failure is the inability of a software system or component to perform its required functions within specified performance requirements. When a defect reaches the end customer it is called a Failure. During development, Failures are usually observed by testers.

FAULT:

An incorrect step, process or data definition in a computer program that causes the program to perform in an unintended or unanticipated manner. A fault is introduced into the software as the result of an error. It is an anomaly in the software that may cause it to behave incorrectly, and not according to its specification. It is the result of the error.

The software industry can still not agree on the definitions for all the above. In essence, if you use the term to mean one specific thing, it may not be understood to be that thing by your audience.

What is Defect Life Cycle?

Defect Life Cycle or Bug Life Cycle in software testing is the specific set of states that defect or bug goes through in its entire life. The purpose of Defect life cycle is to easily coordinate and communicate current status of defect which changes to various assignees and make the defect fixing process systematic and efficient.

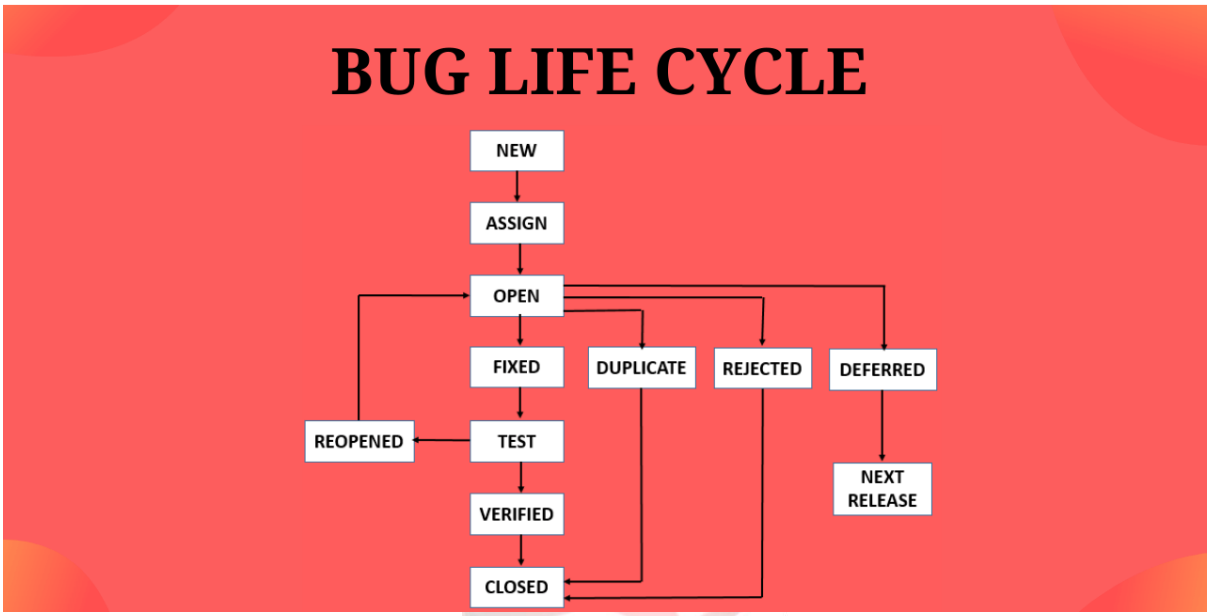
Defect Status

Defect Status or Bug Status in defect life cycle is the present state from which the defect or a bug is currently undergoing. The goal of defect status is to precisely convey the current state or progress of a defect or bug in order to better track and understand the actual progress of the defect life cycle.

The number of states that a defect goes through varies from project to project. Below lifecycle diagram, covers all possible states

- **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.”

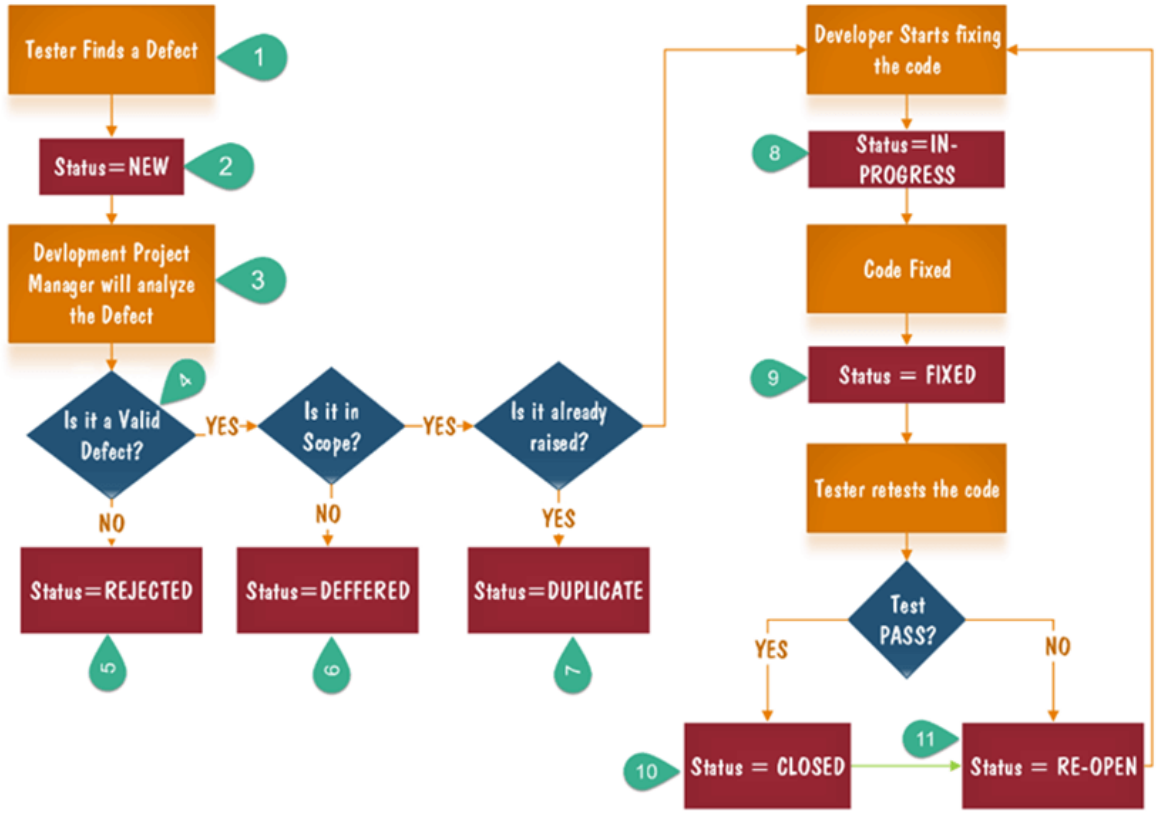
- **Retest**: Tester does the retesting of the code at this stage to check whether the defect is fixed by the developer or not and changes the status to “Re-test.”



- **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 Life Cycle Explained



1. Tester finds the defect
2. Status assigned to defect- New
3. A defect is forwarded to Project Manager for analyze

4. Project Manager decides whether a defect is valid
5. Here the defect is not valid- a status is given “Rejected.”
6. So, project manager assigns a status **rejected**. If the defect is not rejected then the next step is to check whether it is in scope. Suppose we have another function- email functionality for the same application, and you find a problem with that. But it is not a part of the current release when such defects are assigned as a **postponed or deferred** status.
7. Next, the manager verifies whether a similar defect was raised earlier. If yes defect is assigned a status **duplicate**.
8. If no the defect is assigned to the developer who starts fixing the code. During this stage, the defect is assigned a status **in- progress.**
9. Once the code is fixed. A defect is assigned a status **fixed**
10. Next, the tester will re-test the code. In case, the Test Case passes the defect is **closed.** If the test cases fail again, the defect is **re-opened** and assigned to the developer.
11. Consider a situation where during the 1st release of Flight Reservation a defect was found in Fax order that was fixed and assigned a status closed. During the second upgrade release the same defect again re-surfaced. In such cases, a closed defect will be **re-opened.**

What are the different methods of testing?

There are three methods of software testing and they are as follows:

- **Black-box testing:** It is a testing strategy based solely on requirements and specifications. In this strategy, it requires no knowledge of internal paths, structures, or implementation of the software being tested.
- **White box testing:** It is a testing strategy based on internal paths, code structures, and implementation of the software being tested. White box testing generally requires detailed programming skills.
- **Gray box testing:** It is a strategy for software debugging in which the tester has limited knowledge of the internal details of the program.

Explain what is testing type and what are the commonly used testing type?

To get an expected test outcome, a standard procedure is followed which is referred to as Testing Type.

Commonly used testing types are

- ***Unit Testing***: Test the smallest code of an application
- ***API Testing***: Testing API created for the application
- ***Integration Testing***: Individual software modules are combined and tested
- ***System Testing***: Complete testing of the system

- ***Install/UnInstall Testing***: Testing done from the point of client/customer view
- ***Agile Testing***: Testing through Agile technique

Mention what bottom-up testing is?

Bottom-up testing is an approach to integration testing, where the lowest level components are tested first, then used to facilitate the testing of higher level components. The process is repeated until the component at the top of the hierarchy is tested.

Mention what the purpose behind doing end-to-end testing is?

End-to-end testing is done after functional testing. The purpose behind doing end-to-end testing is that

- To validate the software requirements and integration with external interfaces
- Testing application in real-world environment scenario
- Testing of interaction between application and database

In white box testing, what do you verify?

In white box testing following steps are verified.

1. Verify the security holes in the code
2. Verify the incomplete or broken paths in the code
3. Verify the flow of structure according to the document specification

4. Verify the expected outputs
5. Verify all conditional loops in the code to check the complete functionality of the application
6. Verify the line by line coding and cover 100% testing

What is black box testing? What are the different black box testing techniques?

Black box testing is the software testing method which is used to test the software without knowing the internal structure of code or program. This testing is usually done to check the functionality of an application. The different black box testing techniques are

1. Equivalence Partitioning
2. Boundary value analysis
3. Cause-effect graphing

What is Verification and Validation in Software Testing?

***Verification*:** It is a static analysis technique. Here, testing is done without executing the code. Examples include – Reviews, Inspection, and walkthrough.

***Validation*:** It is a dynamic analysis technique where testing is done by executing the code. Examples include functional and non-functional testing techniques.

In the V model, the development and QA activities are done simultaneously. There is no discrete phase called Testing, rather testing starts right from the requirement phase. The verification and validation activities go hand in hand.

Explain Load Testing on websites?

To access a website, a user sends a "request" to that website's server, and the server sends back a response in the form of the website you want to access. To load test a website, quality assurance engineers and automation engineers just need to multiply the number of responses sent to simulate different traffic loads. The web server's response to the influx of virtual users can then be measured. This is used to determine performance issues and server capacity.

Performance Testing vs Load Testing vs Stress Testing (Difference)

Difference between Performance Testing, Load Testing, and Stress Testing – With Examples

Our previous tutorial in this series will be the best **Performance Testing Guide** for any beginner.

In the software testing field, we come across terms like performance testing, load testing, stress testing, etc. These terms are often misunderstood and interpreted as the same concepts.

However, there is a significant difference between these three testing types and it is important for a tester to understand the

Difference between Performance Testing, Load Testing, and Stress Testing

Performance Testing

What is Performance Testing?

Performance testing is the testing that is performed to ascertain how the components of a system are performing under a certain given situation.

Resource usage, scalability, and reliability of the product are also validated under this testing. This testing is a subset of performance engineering, which is focused on addressing performance issues in the design and architecture of a software product.

The above image clearly explains to us that ***Performance Testing is the superset for both load & stress testing.*** Other types of testing included in performance testing are Spike testing, Volume testing, Endurance testing, and Scalability testing. Thus, performance testing is basically a very wide term.

Performance Testing Goal:

The primary goal of performance testing includes establishing the benchmark behavior of the system. There are a number of industry-defined benchmarks that should be met during performance testing.

Performance testing does not aim to find defects in the application. It also does not pass or fail the test. Rather, it addresses the critical task of setting the benchmark and standard for an application. Performance testing should be done very accurately. Close monitoring of application/system performance is the primary characteristic of performance testing.

The benchmark and standard of the application should be set in terms of attributes like speed, response time, throughput, resource usage, and stability. All these attributes are tested in a performance test.

For Example,

For instance, you can test the application's network performance through the "Connection Speed vs. Latency' chart. Latency is the time difference between the data to reach from the source to the destination.

A 70kb page would not take more than 15 seconds to load for the worst connection of a 28.8kbps modem (latency=1000 milliseconds), while a page of the same size would appear within 5 seconds for the average connection of 256kbps DSL (latency=100 milliseconds).

A 1.5mbps T1 connection (latency=50 milliseconds) would have the performance benchmark set as 1 second to achieve this target.

Another **example** would be that of a Request-response model. We can set a benchmark that the time difference between the generation of requests and acknowledgment of responses should be in the range of x ms (milliseconds) and y ms, where x and y are the standard digits.

A successful performance test should project most of the performance issues, which could be related to database, network, software, hardware, etc.

Load Testing

Load testing is meant to test the system by constantly and steadily increasing the load on the system until it reaches the threshold limit. This is a subset of performance testing.

Load testing can be easily done by employing any of the suitable automation tools available in the market. WAPT and LoadRunner are two such famous tools that aid in load testing. Load testing is also famous by names like **Volume testing** and **Endurance testing**.

However, Volume testing mainly focuses on databases. Endurance testing tests the system by keeping it under a significant load for a sustained time period.

The sole purpose of load testing is to assign the system the largest job it can possibly handle to test the endurance of the system and monitor the results. An interesting fact here is that sometimes the system is fed with an empty task to determine the behavior of the system in a zero-load situation.

The attributes which are monitored in the load test include peak performance, server throughput, response time under various load levels (below the threshold of break), adequacy of H/W environment, and how many user applications it can handle without affecting the performance.

Load Testing Goal:

The goals of load testing include:

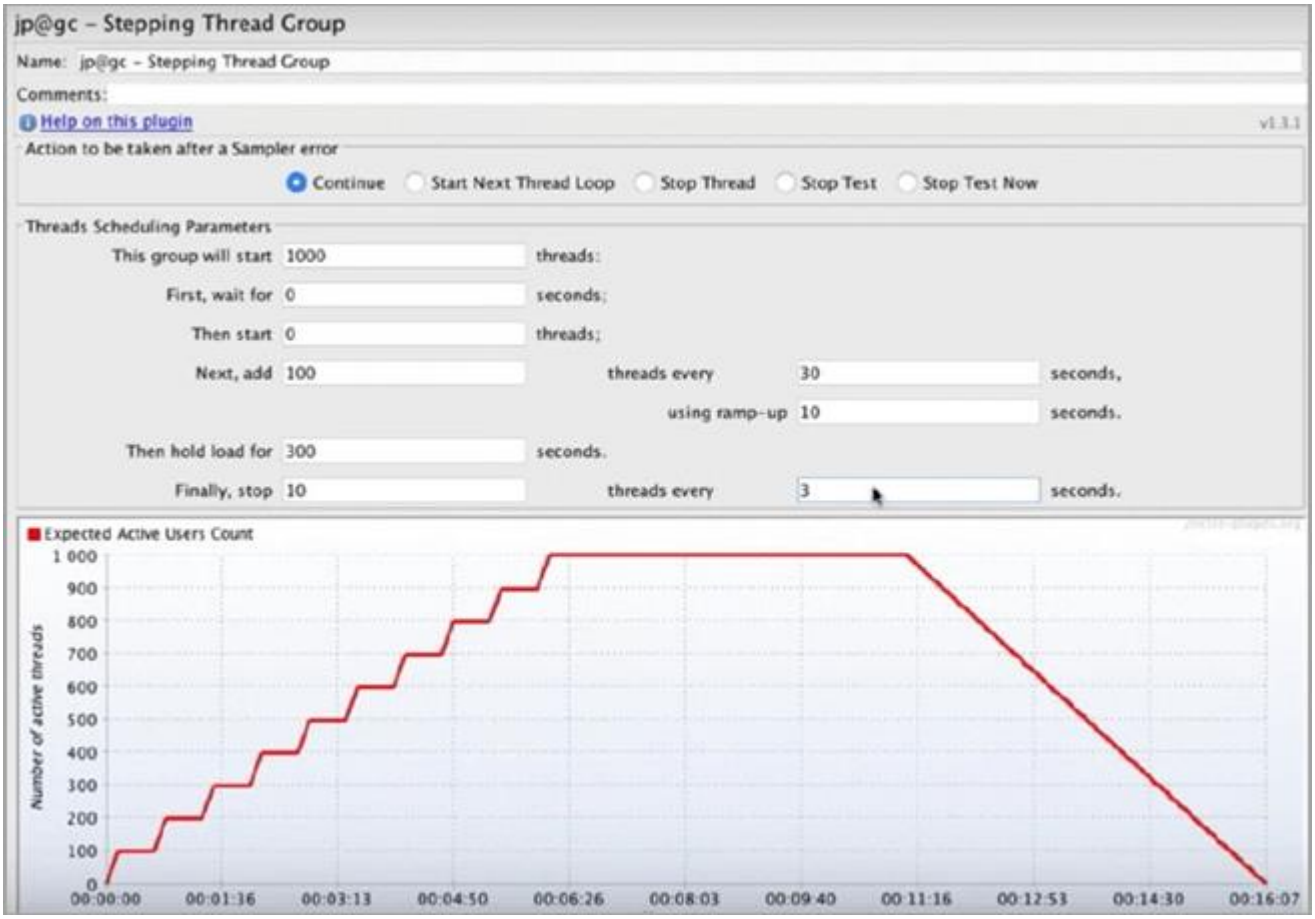
- Exposing defects in an application related to buffer overflow, memory leaks and mismanagement of memory. The issues that will eventually come out as a result of load testing may include load balancing problems, bandwidth issues, the capacity of the existing system, etc.
- To determine the upper limit of all the components of an application like a database, hardware, network, etc. so that the application can manage the anticipated load in the future.
- To set the SLAs for the application.

For Example,

Let's consider to check the email functionality of an application that could be flooded with up to 1000 users at a time. Now, 1000 users can fire the email transactions (read, send, delete, forward, reply) in many different ways.

If we take one transaction per user per hour, then it would be 1000 transactions per hour. By simulating 10 transactions/users, we can load test the email server by occupying it with 10000 transactions/hour.

Another Example of a load test is shown in the image below:



The above image depicts a load test done in the tool called JMeter. This test is done to identify how many users a system can handle. In this test, 100 users are added after every 30 seconds until the load reaches 1000 users. Each step takes 30 seconds to complete and JMeter waits for 30 seconds before kicking off the next step.

Once the load reaches 1000 threads, all of them will continue running for 300 seconds (5 mins) together and then finally stops 10 thread every 3 seconds.

Stress Testing

Under stress testing, various activities to overload the existing resources with excess jobs are carried out in an attempt to break the system down. **Negative testing**, which includes removal of components from the system, is also done as part of stress testing.

Also known as **fatigue testing**, this testing should capture the stability of an application by testing it beyond its bandwidth capacity.

Thus, stress testing evaluates the behavior of an application beyond peak load and normal conditions.

The purpose of stress testing is to ascertain the failure of the system and to monitor how the system recovers back gracefully. The challenge here is to set up a controlled environment before launching the test so that you can precisely capture the behavior of the system repeatedly under the most unpredictable scenarios.

Issues that will eventually come out as a result of stress testing may include synchronization issues, memory leaks, race conditions, etc. If the stress test is checking how the system behaves in the situation of a sudden ramp-up in the number of users, then it is called a spike test.

If the stress test is to check the system's sustainability over a period of time through a slow ramp-up in the number of users, it is called a soak test.

Stress Testing Goal:

The goal of stress testing is to analyze post-crash reports to define the behavior of the application after failure.

The biggest challenge is to ensure that the system does not compromise the security of sensitive data after the failure. In successful stress testing, the system will return to normality along with all its components even after the most terrible breakdown.

For Example,

A word processor like Writer1.1.0 by OpenOffice.org is utilized in the development of letters, presentations, spreadsheets, etc. The purpose of our stress testing is to load it with excess characters.

To do this, we will repeatedly paste a line of data, until it reaches its threshold limit of handling a large volume of text. As soon as the character size reaches 65,535 characters, it would simply refuse to accept more data.

The result of stress testing on Writer 1.1.0 produces a result that does not crash under stress and it handles the situation gracefully which ensures that the application is working correctly even under rigorous stress conditions.

Explain the difference between Authorization and Authentication in Web testing.

	Authentication	Authorization
1	Authentication is the process with which the system identifies who the user is?	Authorization is the process with which system identifies what user is authorized to do?
2	Authentication determines the identity of the user.	Authorization decides the privileges given to the user i.e. whether the user can access or manipulate features of certain program.
3	There are different types of authentications, like password based, device based, etc.	There are two types of authorizations, like read only and read write both.
4	For example: Within an organization, each and every employee can login into an intranet application.	For example: Only account manager or person in accounts department can access account section.

What is Positive Testing?

Answer: It is the form of testing which is conducted on the application to determine if the system works properly or not. Basically, it is known as the “test to pass” approach.

What is Negative Testing?

Answer: Testing software with a negative approach to check if the system is not “showing error when not supposed to” and “not showing error when supposed to” is termed as Negative Testing.

What is an End-to-End Testing?

Answer: Testing the overall functionality of the system including the data integration among all the modules is called End-to-End Testing.

What is Exploratory Testing?

Answer: Exploring the application, understanding its functionalities, adding (or) modifying the existing test cases for better testing is called Exploratory

What is Monkey Testing?

Answer: Testing conducted on an application without any plan and carried out randomly with the tests to find any system crash with the intention of finding tricky defects is called Monkey Testing.

What is Non-Functional Testing?

Answer: Validating various non-functional aspects of the system such as user interfaces, user-friendliness, security, compatibility, Load, Stress, and Performance, etc., is called Non-Functional testing.

What is Usability Testing?

Answer: Checking how easily the end-users are able to understand and operate the application is called Usability Testing.

What is Security Testing?

Answer: Validating whether all security conditions are properly implemented in the software (or) not is called Security testing.

What is the full form of CMMI?

Answer: Capability Maturity Model Integration

What is Integration Level Testing?

Answer: Testing of related programs, modules (or) unit of code.
(or)
Partitions of the system which are ready for testing with other partitions of the system are termed as Integration level testing.

What is System Level Testing?

Answer: Testing of the entire computer system across all the modules is termed as System-level testing. This kind of testing can include Functional as well as Structural Testing.

What is Alpha Testing?

Answer: Testing of a whole computer system before rolling out to the UAT is termed as Alpha testing.

What is Re-Testing?

Answer: Re-testing the application means verifying whether the defects have been fixed or not.

What is Regression Testing?

Answer: Verifying an existing functional and non-functional area after making changes to the part of a software or addition of new features is termed as Regression Testing.

What is Globalization Testing?

Answer: It is the process of verifying whether the software can be run independently of its geographical and cultural environment. Verifying if the application has the feature to set and change language, date, format, and currency or if it is designed for global users.

What is Localization Testing?

Answer: Verifying globalized application for a particular locality of users, under cultural and geographical conditions is termed as Localization Testing.

What is the difference between QA, QC, and Testing?

Answer:

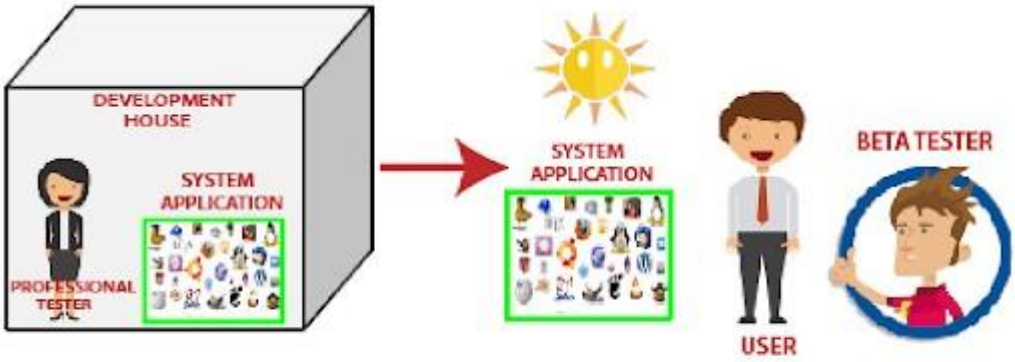
- **QA:** It is processbug-oriented and its aim is to prevent the defects in an application.
- **QC:** QC is product-oriented and it is a set of activities used to evaluate a developed work product.
- **Testing:** Executing and verifying an application with the intention of finding defects.

What is Alpha Testing?

Alpha Testing is a type of Acceptance Testing performed by the testers who are part of the organization, in other words: internal employees. It is the final stage of testing and it is usually done to verify that an application is free of errors / bugs before being launched on the market.

This test uses black box and white box techniques and it is performed near the end of the development of the software and before Beta Testing.

What is Beta Testing?



BETA TESTING OF THE PRODUCT IN REAL WORLD ENVIRONMENT

Beta Testing is performed by real users and it is unstructured. It can be considered as a form of external User Acceptance Testing.

Users can freely use the application and then they are encouraged to give feedback about their experience. This test is more focused on performance and scalability.

Beta Testing helps reduce product failures and provides higher product quality through customer validation that resulted from their experience with the application.

Alpha Testing vs Beta Testing

To clarify the distinction between the two, let's look at the comparison table:

Alpha Testing	**Beta Testing**
It is done by internal testers of the organization.	It is done by real users. (customers)
It is an internal test, performed within the organization.	It is an external test, carried out in the user's environment.
Alpha Testing uses both black box and white box testing techniques	Beta Testing only uses the black box testing technique.
Identifies possible errors.	Checks the quality of the product.

Developers start fixing bugs as soon as they are identified.	Errors are found by users and feedback is necessary.
Long execution cycles.	It only takes a few weeks.
It can be easily implemented as it is done before the near end of development.	It will be implemented in the future version of the product.
It is performed before Beta Testing.	It is the final test before launching the product on the market.
It answers the question: Does the product work?	It answers the question: Do customers like the product?

Functionality and usability are tested.	Usability, functionality, security and reliability are tested with the same depth.

Key Differences

- Alpha Testing is performed by internal employees of the organization and Beta Testing is done by users.
- Alpha Testing is done within the organization, while Beta Testing is done in the user's environment.
- During Alpha Testing only functionality and usability are tested, while during Beta Testing usability, functionality, security, and reliability are tested to the same depth.
- Long execution cycles may be needed for Alpha Testing while just a few weeks of execution make Beta Testing possible.

Key Similarities

- Alpha and Beta Testing are forms of Acceptance Testing and they both try to determine how an application and its features behave.
- When the goal is to obtain the most comprehensive tests, both the Alpha and Beta Tests are typically run. That way, every application of the product can be tested and gather feedback internally and externally.
- Both of them ensure a good quality product.

What is mutation testing?

Mutation testing is a technique to identify if a set of test data or test case is useful by intentionally introducing various code changes (bugs) and retesting with original test data/ cases to determine if the bugs are detected

Original Program	**Mutant Program**
If (x>y)	If(x<y)
Print “Hello”	Print “Hello”
Else	Else
Print “Hi” Else If(x=y)	Print “Hi”

What do you mean by regression and confirmation testing?

Regression Testing: It is defined as a type of software testing to confirm that a recent code change has not adversely affected existing features.

Confirmation Testing: When a test fails because of the defect, the defect is reported. Then a new version of the software is submitted whose defect is fixed. This is called as **confirmation testing** or re-**testing**.

What do you mean by boundary value analysis?

Boundary Value Analysis (BVA) is a ***black box test*** design technique which is applied to see if there are any ***bugs*** at the boundary of the input domain.

What is Random testing?

Usually, in Random testing, data is generated randomly often using a tool. For example, the following figure shows how randomly-generated data is sent to the system.

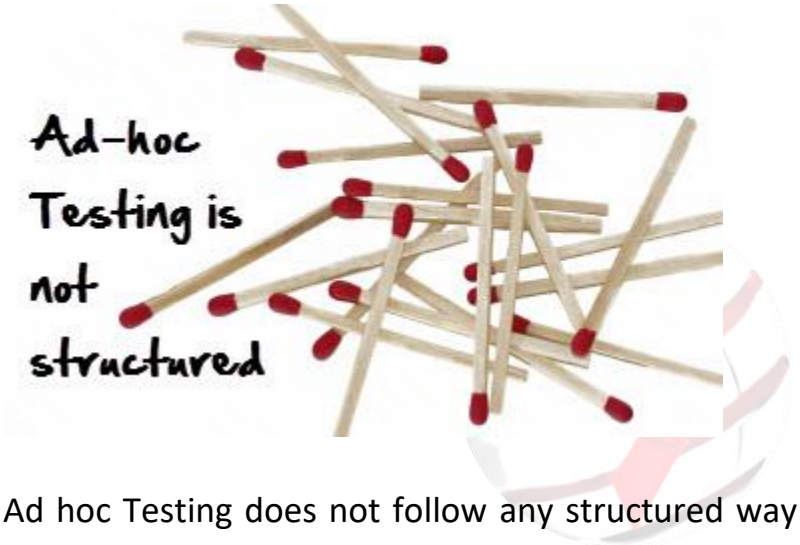
What is the difference between UAT (User Acceptance Testing) and System testing?

System Testing: System testing is finding defects when the system undergoes testing as a whole; it is also known as end-to-end testing. In such type of testing, the application suffers from beginning till the end.

UAT: User Acceptance Testing (UAT) involves running a product through a series of specific tests which determines whether the product will meet the needs of its users.

Ad hoc Testing

Ad hoc Testing is an informal or unstructured software testing type that aims to break the testing process in order to find possible defects or errors at an early possible stage. Ad hoc testing is done randomly and it is usually an unplanned activity which does not follow any documentation and test design techniques to create test cases.



Ad hoc 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 Software 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.

This testing requires no documentation/ planning /process to be followed. Since this testing aims at finding defects through random approach, without any documentation, defects will not be mapped to test cases. This means that, sometimes, it is very difficult to reproduce the defects as there are no test steps or requirements mapped to it.

7 Principles of Software Testing

1. Testing shows presence of defects
2. Exhaustive testing is not possible
3. Early testing
4. Defect clustering
5. Pesticide paradox
6. Testing is context dependent
7. Absence of errors fallacy

1) Exhaustive testing is not possible

Yes! Exhaustive testing is not possible. Instead, we need the optimal amount of testing based on the risk assessment of the application.

So if you were testing this Operating system, you would realize that defects are likely to be found in multi-tasking activity and need to be tested thoroughly which brings us to our next principle Defect Clustering

2) Defect Clustering

Defect Clustering which states that a small number of modules contain most of the defects detected. This is the application of the Pareto Principle to software testing: approximately 80% of the problems are found in 20% of the modules.

By experience, you can identify such risky modules. But this approach has its own problems

If the same tests are repeated over and over again, eventually the same test cases will no longer find new bugs.

3) Pesticide Paradox

Repetitive use of the same pesticide mix to eradicate insects during farming will over time lead to the insects developing resistance to the pesticide Thereby ineffective of pesticides on insects. The same applies to software testing. If the same set of repetitive tests are conducted, the method will be useless for discovering new defects.

To overcome this, the test cases need to be regularly reviewed & revised, adding new & different test cases to help find more defects.

Testers cannot simply depend on existing test techniques. He must look out continually to improve the existing methods to make testing more effective. But even after all this sweat & hard work in testing, you can never claim your product is bug-free. To drive home this point, let's see this video of the public launch of Windows 98

You think a company like MICROSOFT would not have tested their OS thoroughly & would risk their reputation just to see their OS crashing during its public launch!

4) Testing shows a presence of defects

Hence, testing principle states that – Testing talks about the presence of defects and don't talk about the absence of defects. i.e. Software 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.

But what if, you work extra hard, taking all precautions & make your software product 99% bug-free. And the software does not meet the needs & requirements of the clients.

This leads us to our next principle, which states that- Absence of Error

5) Absence of Error – fallacy

It is possible that software which is 99% bug-free is still unusable. This can be the case if the system is tested thoroughly for the wrong requirement. Software testing is not mere finding defects, but also to check that software addresses the business needs. The absence of Error is a Fallacy i.e. finding and fixing defects does not help if the system build is unusable and does not fulfil the user's needs & requirements.

To solve this problem, the next principle of testing states that early Testing

6) Early Testing

Early Testing – Testing should start as early as possible in the Software Development Life Cycle. So that any defects in the requirements or design phase are captured in early stages. It is much cheaper to fix a Defect in the early stages of testing. But how early one should start testing? It is recommended that you start finding the bug the moment the requirements are defined. More on this principle in a later training tutorial.

7) Testing is context dependent

Testing is context dependent which basically means that the way you test an e-commerce site will be different from the way you test a commercial off the shelf application. All the developed software's are not identical. You might use a

different approach, methodologies, techniques, and types of testing depending upon the application type. For instance testing, any POS system at a retail store will be different than testing an ATM machine.

What is Smoke Testing

Smoke Testing is a software testing process that determines whether the deployed software build is stable or not. Smoke testing is a confirmation for QA team to proceed with further software testing. It consists of a minimal set of tests run on each build to test software functionalities. Smoke testing is also known as “Build Verification Testing” or “Confidence Testing.”

In simple terms, smoke tests means verifying the important features are working and there are no showstoppers in the build that is under testing. It is a mini and rapid regression test of major functionality. It is a simple test that shows the product is ready for testing. This helps determine if the build is flawed as to make any further testing a waste of time and resources.

Advantages of Smoke testing

Here are few advantages listed for Smoke Testing.

- Easy to perform testing
- Defects will be identified in early stages.
- Improves the quality of the system
- Reduces the risk
- Progress is easier to access.
- Saves test effort and time

- Easy to detect critical errors and correction of errors.
- It runs quickly
- Minimises integration risks

What is Sanity Testing?

Sanity testing is a kind of Software Testing performed after receiving a software build, with minor changes in code, or functionality, 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.

Key Difference between Sanity and Smoke Test

- Smoke Testing has a goal to verify “stability” whereas Sanity Testing has a goal to verify “rationality”.
- Smoke Testing is done by both developers and testers whereas Sanity Testing is done by testers.
- Smoke Testing verifies the critical functionalities of the system whereas Sanity Testing verifies the new functionality like bug fixes.
- Smoke testing is a subset of acceptance testing whereas Sanity testing is a subset of Regression Testing.
- Smoke testing is documented or scripted whereas Sanity testing isn’t.

- Smoke testing verifies the entire system from end to end whereas Sanity Testing verifies only a particular component. Difference between Smoke Testing and Sanity Testing

Following is the difference between Sanity and Smoke testing:

Smoke Testing	Sanity Testing
Smoke Testing is performed to ascertain that the critical functionalities of the program is working fine	Sanity Testing is done to check the new functionality/bugs have been fixed
The objective of this testing is to verify the "stability" of the system in order to proceed with more rigorous testing	The objective of the testing is to verify the "rationality" of the system in order to proceed with more rigorous testing
This testing is performed by the developers or testers	Sanity testing in software testing is usually performed by testers
Smoke testing is usually documented or scripted	Sanity testing is usually not documented and is unscripted
Smoke testing is a subset of Acceptance testing	Sanity testing is a subset of Regression Testing

Smoke testing exercises the entire system from end to end	Sanity testing exercises only the particular component of the entire system
Smoke testing is like General Health Check Up	Sanity Testing is like specialized health check up

Points to note about Smoke and Sanity Tests

- Both Sanity and Smoke testing are ways to avoid wasting time and effort by quickly determining whether an application is too flawed to merit any rigorous testing.
- Smoke Testing is also called tester acceptance testing.
- Smoke testing performed on a particular build is also known as a build verification test.
- One of the best industry practice in software engineering, is to conduct a Daily build and smoke test in software projects.
- Both smoke and sanity tests can be executed manually or using an automation tool. When automated tools are used, the tests are often initiated by the same process that generates the build itself.
- As per the needs of testing, you may have to execute both Sanity and Smoke Tests in the software build. In such cases, you will first execute Smoke tests and then go ahead with Sanity Testing. In industry, test cases for Sanity Testing are commonly combined with that for smoke tests, to speed up test

execution. Hence, it's a common that the terms are often confused and used interchangeably

- **Difference between Smoke Testing and Regression Testing:**

Smoke Testing	Regression Testing
Smoke Testing is the Surface Level Testing to verify stability of system.	Regression Testing is the Deep Level Testing to verify the rationality of system.
Smoke Test is always followed by Regression Test.	Regression Test is always carried out throughout the testing phase.
Test Cases of Smoke Test is a part of Regression Testing and covers only the core functionalities.	Regression testing is obtained from functional specification or Software Requirement Specification (SRS).
Smoke tests are performed by the developers.	Regression tests are performed by the professional testers.
Smoke tests are performed quickly to confirm whether to accept or reject the build.	Regression tests are not accountable for accepting or rejecting a software build for further testing procedures.
This type of testing is performed on when the new software is developed.	This type of testing is performed to check when the change made by tester, code affects the existing features or not.
Cost of Smoke testing is low.	Cost of Regression testing is little bit high.

This testing is documented or scripted.	This testing is not documented or scripted.
It is also known as Build Verification Testing.	It is not known by any other name.
Requirement of time and manpower is less than Regression Testing.	Requirement of time and manpower is more than Smoke Testing.
Smoke testing can be performed by using both automation tools and manually.	Regression testing can be performed by either automation tools or manually.

What is BI Testing?

Business Intelligence (BI) is the process of gathering, cleansing, analyzing, integrating and sharing data to derive actionable insights that drive business growth. Business Intelligence Testing or BI testing verifies the staging data, ETL process, BI reports and ensures the implementation is correct. BI Testing ensures data credibility and accuracy of insights derives from the BI process.

BI Testing Test Cases & Scenarios

Following are generic test cases that need to be validated for any BI Testing Project

ETL verification Test Scenarios

Sample Test Cases

- Verify data is mapped correctly from source to target system
- Verify all tables and their fields are copied from source to target
- Verify keys configured to be auto-generated are created properly in target system
- Verify that null fields are not populated
- Verify data is neither garbled nor truncated
- Verify data type and format in target system is as expected
- Verify there is no duplicity of data in the target system
- Verify transformations are applied correctly
- Verify that the precision of data in numeric fields is accurate

- Verify exception handling is robust

Staging Data Test Scenarios

Sample Test Cases

- Reconciliation check- record count between the STG (staging) tables and target tables are same after applying filter rules
- Insert a record which is not loaded into target table for given key combination
- Copy records, sending same records that are already loaded into target tables-should not be loaded
- Update a record for a key when value columns changed on day_02 loads
- Delete the records logically in the target tables
- Values loaded by process tables
- Values loaded by reference tables

Data Loading in BI Test Scenarios

Sample Test Cases

- Check if the target and source data base are connected well and there are no access issues.
- For a full load, check the truncate option and ensure its working fine.
- While loading the data, check for the performance of the session
- Check for non-fatal errors.

- Verify you can fail the calling parent task if the child task fails.
- Verify that the logs are updated
- Verify mapping and workflow parameters are configured accurately
- Verify the number of tables in source and target systems is the same
- Compare the attributes from stage tables to that of the target tables. They should be matched.

BI Reports Test Scenarios

Sample Test Cases

- Display date and time
- Decimal precision for key figures
- In a given page display the number of rows and columns
- Free characteristics in the report
- How are blank values/data displayed for both characteristics and key figures in the report
- Whether search for characteristics is based on key or key &text as applicable
- Does search option on text is case sensitive- Upper, Lower or both

What is a test case?

A test case is nothing but a set of conditions or variables under which a tester will determine whether a system under test satisfies requirements or works correctly.

What Test Plans consists of?

Test design, scope, test strategies, approach are various details that Test plan document consists of.

1. Test case identifier
2. Scope
3. Features to be tested
4. Features not to be tested
5. Test strategy & Test approach
6. Test deliverables
7. Responsibilities
8. Staffing and training
9. Risk and Contingencies

What is the difference between test scenarios, test cases, and test script?

Difference between test scenarios and test cases is that

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

Test Cases: It is a document that contains the steps that have to be executed; it has been planned earlier.

Test Script: It is written in a programming language and it's a short program used to test part of the functionality of the software system. In other words a written set of steps that should be performed manually.

Explain what Test Plan is? What is the information that should be covered in Test Plan?

A test plan can be defined as a document describing the scope, approach, resources, and schedule of testing activities and a test plan should cover the following details.

- Test Strategy
- Test Objective
- Exit/Suspension Criteria
- Resource Planning
- Test Deliverables

Which test cases are written first: white boxes or black boxes?

Usually, black box test cases are written first and white box test cases later. To write black box test cases we need the requirement document and, design or project plan. These documents are easily available at the initial start of the project. White box test cases cannot be started in the initial phase of the project because they need more architecture clarity which is not available at the start of the project. So normally white box test cases are written after black box test cases are written.

Testing Techniques

- **Boundary Value Analysis (BVA):** As the name suggests it's the technique that defines the testing of boundaries for a specified range of values.
- **Equivalence Partition (EP):** This technique partitions the range into equal parts/groups that tend to have the same behaviour.
- **State Transition Technique**: This method is used when software behaviour changes from one state to another following particular action.
- **Error Guessing Technique:** This is guessing/anticipating the error that may arise while doing manual testing. This is not a formal method and takes advantages of a tester's experience with the application

However, every test case can be broken down into 8 basic steps.

1. Step 1: Test Case ID. ...
2. Step 2: Test Description. ...
3. Step 3: Assumptions and Pre-Conditions. ...
4. Step 4: Test Data. ...
5. Step 5: Steps to be Executed. ...
6. Step 6: Expected Result. ...
7. Step 7: Actual Result and Post-Conditions. ...
8. Step 8: Pass/Fail.

What is a Test Case?

A Test Case is a set of actions executed to verify a particular feature or functionality of your software application. A Test Case contains test steps, test data, precondition, post condition developed for specific test scenario to verify any requirement. The test case includes specific variables or conditions, using which a testing engineer can compare expected and actual results to determine whether a software product is functioning as per the requirements of the customer.

Test scenarios are rather vague and cover a wide range of possibilities. Testing is all about being very specific.

For a Test Scenario: Check Login Functionality there many possible test cases are:

- Test Case 1: Check results on entering valid User Id & Password
- Test Case 2: Check results on entering Invalid User ID & Password
- Test Case 3: Check response when a User ID is Empty & Login Button is pressed, and many more

The format of Standard Test Cases

Below is a format of a standard login Test cases example.

Test Case ID	Test Case Description	Test Steps	Expected Results	Actual Results	Pass/Fail
TC_001	Check Customer Login with valid Data	Go to site http://sumagotest.in/demo Enter UserId Enter Password Click on Submit button	Userid = sumagoinfo86 Password = pass99 User should Login into an application	As Expected	Pass
TC_002	Check Customer Login with invalid Data	Go to site http://sumagotest.in/demo Enter UserId Enter Password Click Submit	Userid = sumagoinfo86 Password = glass99 User should not Login into an application	As Expected	Pass

This entire table may be created in Word, Excel or any other Test management tool. That's all to Test Case Design

This entire table may be created in Word, Excel or any other Test management tool. That's all to Test Case Design

Step 1) A simple test case to explain the scenario would be

Test Case ID	Test Case Description
TC_001	Check response when valid email and password is entered

Step 2) Test the Data.

In order to execute the test case, you would need Test Data. Adding it below

Test Case ID	Test Case Description	Test Data
TC_001	Check response when valid email and password is entered	Email: sumagoinfo86@email.com Password: Sum@g9In40

Identifying test data can be time-consuming and may sometimes require creating test data afresh. The reason it needs to be documented.

Step 3) Perform actions.

In order to execute a test case, a tester needs to perform a specific set of actions on the AUT. This is documented as below:

Test Case ID	Test Case Description	Test Steps	Test Data
TC_001	Check response when valid email and password is entered	1) Enter Email Address 2) Enter Password 3) Click Sign in	Email: sumagoinfo86@email.com Password: Sum@g9In40

Many times the Test Steps are not simple as above, hence they need documentation. Also, the author of the test case may leave the organization or go on a vacation or is sick and off duty or is very busy with other critical tasks. A recently hire may be asked to execute the test case. Documented steps will help him and also facilitate reviews by other stakeholders.

Step 4) Check behaviour of the AUT.

The goal of test cases in software testing is to check behaviour of the AUT for an expected result. This needs to be documented as below

Test Case ID	Test Case Description	Test Steps	Expected Results
TC_001	Check response when valid	Email: sumagoinfo86@email.com Password: Sum@g9In40	Login should be successful

	email and password is entered		

During test execution time, the tester will check expected results against actual results and assign a pass or fail status

Test Case ID	Test Case Description	Test Steps	Expected Results	Actual Results	Pass/Fail
TC_001	Check response when valid email and password is entered	Email: sumagoinfo86@email.com Password: Sum@g9In40	Login should be successful	Login was successful	Pass

Step 5) That apart your test case -may have a field like,

Pre – Condition which specifies things that must be in place before the test can run. For our test case, a pre-condition would be to have a browser installed to have access to the site under test. A test case may also include Post – Conditions which specifies anything that applies after the test case completes. For our test case, a post condition would be time & date of login is stored in the database

Test Case Best Practice

1. Test Cases need to be simple and transparent:

Create test cases that are as simple as possible. They must be clear and concise as the author of the test case may not execute them.

Use assertive language like go to the home page, enter data, and click on this and so on. This makes the understanding the test steps easy and tests execution faster.

2. Create Test Case with End User in Mind

The ultimate goal of any software project is to create test cases that meet customer requirements and is easy to use and operate. A tester must create test cases keeping in mind the end user perspective

3. Avoid test case repetition.

Do not repeat test cases. If a test case is needed for executing some other test case, call the test case by its test case id in the pre-condition column

4. Do not Assume

Do not assume functionality and features of your software application while preparing test case. Stick to the Specification Documents.

5. Ensure 100% Coverage

Make sure you write test cases to check all software requirements mentioned in the specification document. Use Traceability Matrix to ensure no functions/conditions is left untested.

6. Test Cases must be identifiable.

Name the test case id such that they are identified easily while tracking defects or identifying a software requirement at a later stage.

7. Implement Testing Techniques

It's not possible to check every possible condition in your software application. Software Testing techniques help you select a few test cases with the maximum possibility of finding a defect.

- **Boundary Value Analysis (BVA):**
- **Equivalence Partition (EP):**
- **State Transition Technique**:
- **Error Guessing Technique:**

8. Self-cleaning

The test case you create must return the Test Environment to the pre-test state and should not render the test environment unusable. This is especially true for configuration testing.

9. Repeatable and self-standing

The test case should generate the same results every time no matter who tests it

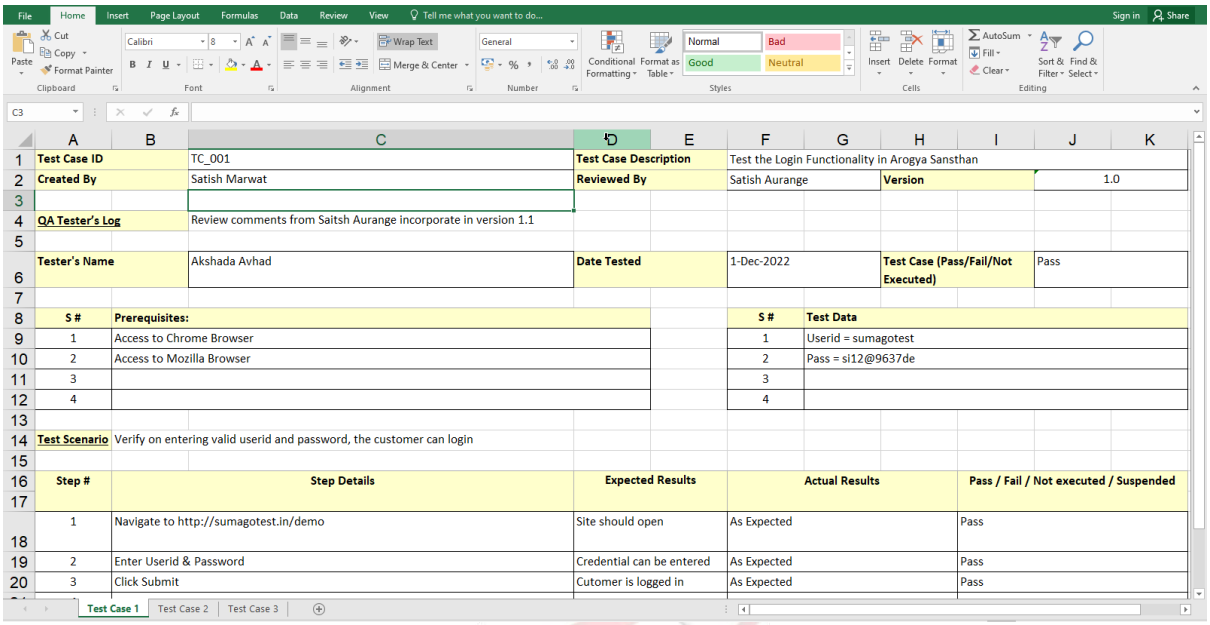
10. Peer Review.

After creating test cases, get them reviewed by your colleagues. Your peers can uncover defects in your test case design, which you may easily miss.

While drafting a test case to include the following information

- The description of what requirement is being tested
- The explanation of how the system will be tested
- The test setup like a version of an application under test, software, data files, operating system, hardware, security access, physical or logical date, time of day, prerequisites such as other tests and any other setup information pertinent to the requirements being tested
- Inputs and outputs or actions and expected results
- Any proofs or attachments
- Use active case language
- Test Case should not be more than 15 steps
- An automated test script is commented with inputs, purpose and expected results
- The setup offers an alternative to pre-requisite tests

Sample Draft



1	**Test Case ID**		TC_001	**Test Case Description**		Test the Login Functionality in Arogya Sansthan				
2	**Created By**		Satish Marwat	**Reviewed By**		Satish Aurange	**Version**		1.0	
3										
4	**QA Tester's Log**		Review comments from Saitsh Aurange incorporate in version 1.1							
5										
6	**Tester's Name**		Akshada Avhad	**Date Tested**		1-Dec-2022	**Test Case (Pass/Fail/Not Executed)**		Pass	
7										
8	**S #**	**Prerequisites:**				**S #**	**Test Data**			
9	1	Access to Chrome Browser				1	Userid = sumagotest			
10	2	Access to Mozilla Browser				2	Pass = si12@9637de			
11	3					3				
12	4					4				
13										
14	**Test Scenario**	Verify on entering valid userid and password, the customer can login								
15										
16	**Step #**	**Step Details**		**Expected Results**		**Actual Results**		**Pass / Fail / Not executed / Suspended**		
17										
18	1	Navigate to http://sumagotest.in/demo		Site should open		As Expected		Pass		
19	2	Enter Userid & Password		Credential can be entered		As Expected		Pass		
20	3	Click Submit		Cutomer is logged in		As Expected		Pass		

How to Write Test Cases For a Login Page (Sample Scenarios)

Whenever you will be asked to write the test cases for the *'Form with some controls',* you need to follow the list of rules for writing test cases as mentioned below:

- Write a test case on each form object.
- Written test cases should be a combination of both negative and positive test cases.
- Also, test cases should always be a combination of functional, performance, UI, usability, and compatibility test cases.
- *When you will be asked in the interview to write the test cases for a login page, firstly you need to think about, how many maximum controls can be available on a login page?*
- Because you don't have a login page in front of you and neither have you had requirements document for this login page. But the login page is such a common thing of which we can easily imagine the controls.
- There can be a username, password, 'Sign In' button, Cancel Button, and Forgot Password link. There can be one more control which is a checkbox named 'Remember me' to remember the login details on a particular machine.

Test Cases – Login Page

Following is the possible list of functional and non-functional test cases for a login page:

Functional Test Cases:

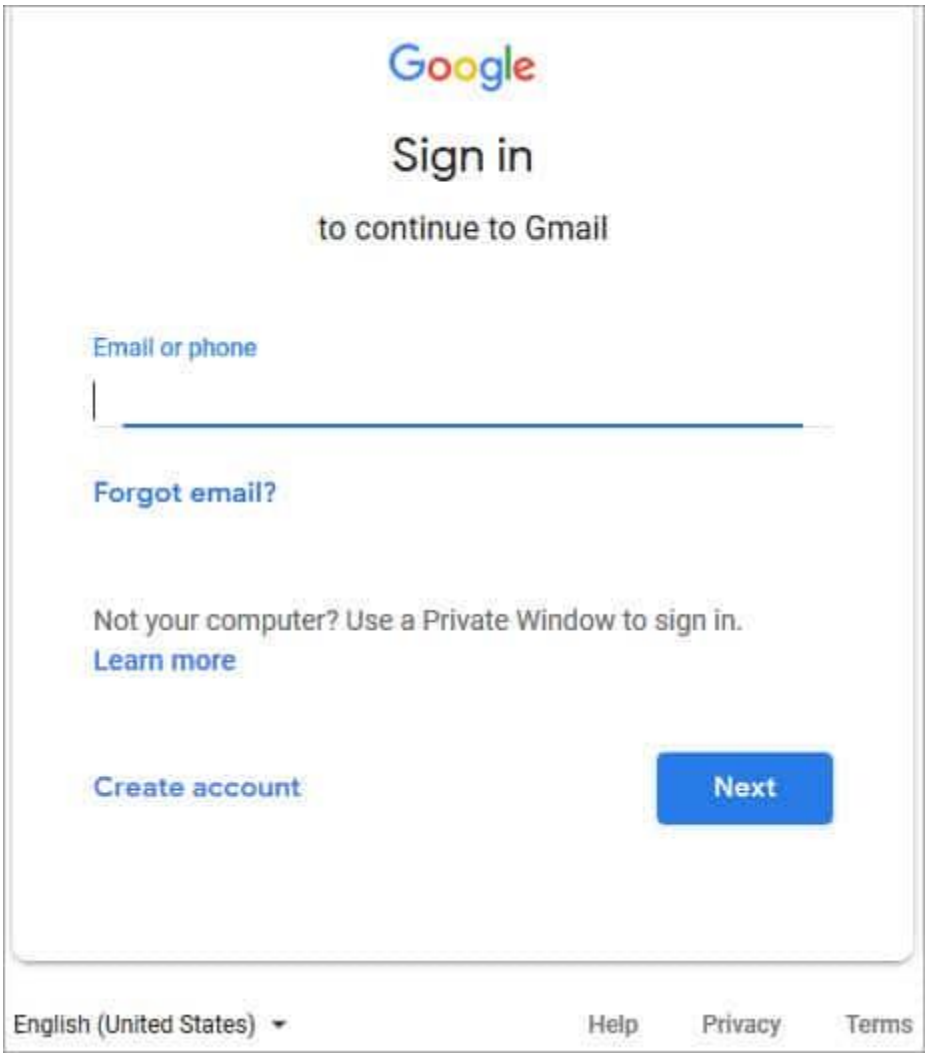
Sr. No.	Functional Test Cases	Type- Negative/ Positive Test Case
1	Verify if a user will be able to login with a valid username and valid password.	Positive
2	Verify if a user cannot login with a valid username and an invalid password.	Negative
3	Verify the login page for both, when the field is blank and Submit button is clicked.	Negative
4	Verify the 'Forgot Password' functionality.	Positive
5	Verify the messages for invalid login.	Positive
6	Verify the 'Remember Me' functionality.	Positive
7	Verify if the data in password field is either visible as asterisk or bullet signs.	Positive

8	Verify if a user is able to login with a new password only after he/she has changed the password.	Positive
9	Verify if the login page allows to log in simultaneously with different credentials in a different browser.	Positive
10	Verify if the 'Enter' key of the keyboard is working correctly on the login page.	Positive
	Other Test Cases	
11	Verify the time taken to log in with a valid username and password.	Performance & Positive Testing
12	Verify if the font, text color, and color coding of the Login page is as per the standard.	UI Testing & Positive Testing
13	Verify if there is a 'Cancel' button available to erase the entered text.	Usability Testing
14	Verify the login page and all its controls in different browsers	Browser Compatibility & Positive Testing.

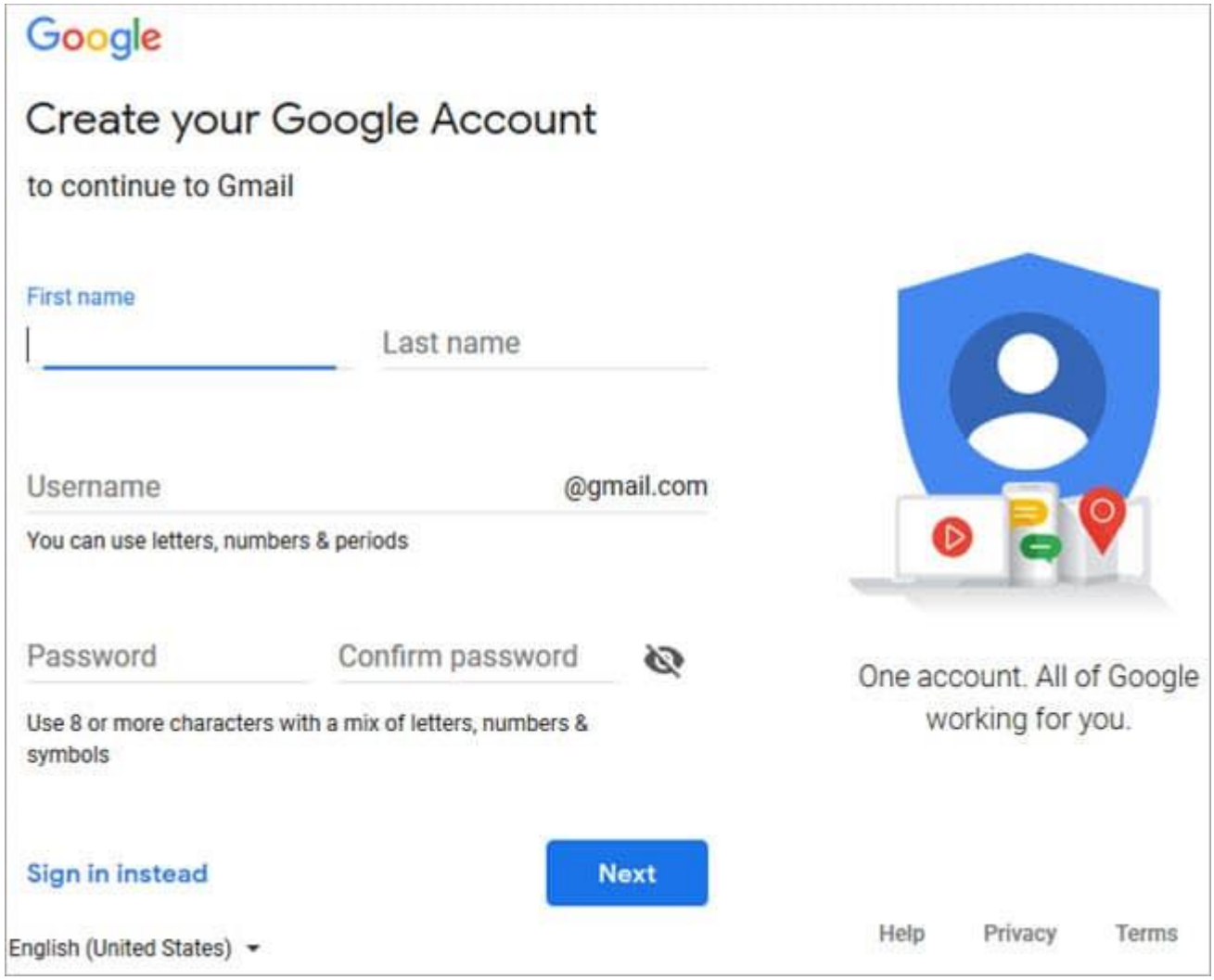
Non-functional Security Test Cases:

Sr. No.	Security test cases	Type- Negative/ Positive Test Case
1	Verify if a user cannot enter the characters more than the specified range in each field (Username and Password).	Negative
2	Verify if a user cannot enter the characters more than the specified range in each field (Username and Password).	Positive
3	Verify the login page by pressing 'Back button' of the browser. It should not allow you to enter into the system once you log out.	Negative
4	Verify the timeout functionality of the login session.	Positive
5	Verify if a user should not be allowed to log in with different credentials from the same browser at the same time.	Negative
6	Verify if a user should be able to login with the same credentials in different browsers at the same time.	Positive
7	Verify the Login page against SQL injection attack.	Negative
8	Verify the implementation of SSL certificate.	Positive

We can take an **Example** of Gmail Login page. Here is the image of it.


Google
Sign in
to continue to Gmail
Email or phone
Forgot email?
Not your computer? Use a Private Window to sign in.
Learn more
Create account
Next
English (United States)
Help
Privacy
Terms

Test Cases for Gmail Login page



Sr. No.	Test Scenarios
1	Enter the valid email address & click next. Verify if the user gets an option to enter the password.

2	Don't enter an email address or phone number & just click the Next button. Verify if the user will get the correct message or if the blank field will get highlighted.
3	Enter the invalid email address & click the Next button. Verify if the user will get the correct message.
4	Enter an invalid phone number & click the Next button. Verify if the user will get the correct message.
5	Verify if a user can log in with a valid email address and password.
6	Verify if a user can log in with a valid phone number and password.
7	Verify if a user cannot log in with a valid phone number and an invalid password.
8	Verify if a user cannot log in with a valid email address and a wrong password.
9	Verify the 'Forgot email' functionality.
10	Verify the 'Forgot password' functionality.

Test Scenarios for the Sign-up page

#1) Verify the messages for each mandatory field.

#2) Verify if the user cannot proceed without filling all the mandatory fields.

#3) Verify the age of the user when the DOB is selected.

#4) Verify if the numbers and special characters are not allowed in the First and Last name.

#5) Verify if a user can sign-up successfully with all the mandatory details.

#6) Verify if a user can log in with the valid details.

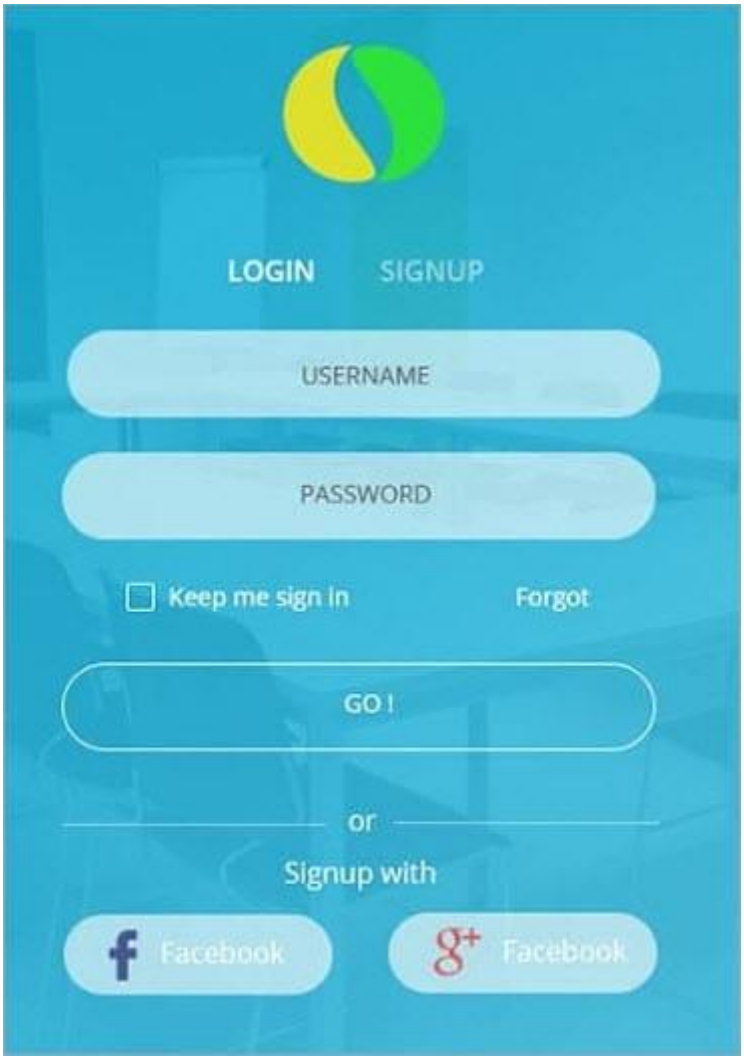
#7) Verify if the Password and Confirm Password fields are accepting similar strings only.

#8) Verify if the Password field will prompt you for the weak passwords.

#9) Verify if duplicate email address will not get assigned.

#10) Verify that hints are provided for each field on the form, for the ease of use.

Test Scenarios for the Login page of Mobile Application



#1) Verify if a user can log in with a valid username and password.

#2) Verify if a user cannot log in with an invalid username or password. Check permutation and combinations of this.

#3) Verify the 'Keep me Sign In' option. If this check box is selected, then the user should not get logged out even after exiting the app.

#4) Verify if this check box is not selected by default.

#5) If the user has signed up with Facebook or social media, verify that the user can log in with those credentials or not.

#6) Verify the Forgot password functionality.

#7) Verify if the login page fits the mobile screen. The user should not have to scroll the screen.

Conclusion

While writing test cases for login or sign-up page write the test cases for all the fields. There should be a combination of both positive and negative test cases. Try to cover the performance, security, and functional scenarios.

The login page is the page with fewer controls, so even though it is looking simple for testing, it should not be considered as an easy task.

Also many a time it is the first impression of an application, so it should be perfect for user interface and usability.

BUG Tools

1) Monday

Monday is a bug tracking tool that that enables you to analyse your performance and manage your team in one place. It provides flexible dashboard for easy visualization of data.

Integrations: Outlook, Microsoft Teams, Dropbox, Slack, Google Calendar, Google Drive etc.

Customer Support: Contact Form

Free Trial: Life Time Free Basic Plan

Features:

- You can collaborate with other people.
- It can automate your daily work.
- You can track your work progress.
- Capture a screenshot or video feedback
- Powerful team collaboration
- It enables you to work remotely.
- It provides customer support via Contact Form
- Seamlessly integrates with Outlook, Microsoft Teams, Dropbox, Slack, Google Calendar, Google Drive, Excel, Gmail, LinkedIn, OneDrive, Zapier, and Adobe Creative Cloud
- You can export your file in PDF, PNG, JPEG, SVG, and CSV formats
- Offers instant Email, and Slack alerts
- Set Scans to run hourly, daily, and weekly

- Provide an option for width, notifications, and design customization
- Supports programming languages like PHP, Python, and Javascript
- Offers Automations, Gantt, Kanban, Time tracking, Security, HIPPA, authentication, and SCIM
- **Supported Platforms:** Windows, Mac, iOS, Android, and Linux
- **Price:** Plans start at $10 a month. 18% Discount on Yearly Payment.
- **Free Trial:** Life Time Free Basic Plan

Pros	Cons
No time restrictions and free of charge	The minimum team size for paid plans is three people
The dashboard view is great	The basic plan lacks sharing functionality
Tracking features that are great	Addition of subtasks is a process that can be challenging
Reliable project organization and management	The response time of the support team is too slow
Dashboard that is modern, intuitive, and visually appealing	
Offers a Kanban-style visual layout	

2) SpiraTeam

SpiraTeam is a turn-key Application Lifecycle Management (ALM) solution with completely integrated bug tracking functionality. SpiraTeam lets you manage your entire testing process from requirements to tests, bugs, and issues, with end-to-end traceability built in. SpiraTeam's come with the following features out of the box:

Integrations: Visual Studio, Android Studio, Eclipse, TFS, HelixCore, VSS, and Mercurial

Customer Support: Contact Form, Phone, and Email

Free Trial: 30 Days Free Trial (No Credit Card Required)

Features:

- Automatic creation of new incidents during the test script execution.
- Fully customizable incident fields including statuses, priorities, defect types, and severities.
- Ability to link incidents (bugs) to other artifacts and incidents.
- Capture a screenshot or video feedback
- Powerful team collaboration
- Robust reporting, searching, and sorting, plus an Audit log tracking changes.
- Email notifications triggered by the customized workflow status changes.
- Ability to report issues and bugs via email.
- It provides customer support via Contact Form, Phone, and Email
- Seamlessly integrates with Visual Studio, Android Studio, Eclipse, TFS, HelixCore, VSS, and Mercurial

- You can export your file in Excel, PDF, Perl TAP, and XML formats
- Offers instant Email alerts
- Set Scans to run hourly, daily, weekly, and monthly
- Provide an option for reporting, and workflows customization
- Supports programming languages like Python, Java, C#, and Javascript
- Offers Planning Board, Project Management, Source Code Management, Task Management, Resource Management, Build Management, Bug Tracking, Automated Testing, Exploratory Testing, and IDE Integration
- **Supported Platforms:** Windows, and Linux
- **Price:** Request a Quote from Sales.
- **Free Trial:** 30 Days Free Trial (No Credit Card Required)

👍 Pros	👎 Cons
It is very easy to maintain roles and access for users	The reporting process could be made a bit easier
Test cycles are easy to maintain	There are some functions that are not always intuitive to use
A good defect tracking system	The process of document baselining can be a bit tricky
Configurable to particular project needs	Customer service should be available 24 hours
Entire collaboration suite for the software development life cycle	
A great product at the best price on the market	

3) BugHerd

BugHerd is the easiest way to track bugs and manage website feedback. Pin bugs and feedback to elements on a website and capture the technical information to help resolve issues. Track feedback tasks to completion with the kanban style task board.

BugHerd.

Integrations: Slack, GitHub, Zapier, WordPress, and Integromat

Customer Support: Contact Form, and Email

Free Trial: 14 Days Free Trial (No Credit Card Required)

Features:

- Simple point-and-click bug and feedback capture
- Technical information like browser, OS, screen resolution is captured for you
- Track and manage all feedback and bugs in one central location with the kanban style Task Board
- Capture a screenshot or video feedback
- Powerful team collaboration
- Save hours during QA and UAT of websites and applications
- Stakeholders love it.
- It provides customer support via Contact Form, and Email
- Seamlessly integrates with Slack, GitHub, Zapier, WordPress, and Integromat
- You can export your file in CSV, XML, and JSON formats
- Offers instant Email, and Slack alerts
- Set Scans to run hourly, daily, and weekly

- Provide an option for Widget customization
- Supports programming languages like HTML, and PHP
- Offers Real-time commenting, Unlimited guests, Unlimited projects, and Version control-sync
- **Supported Platforms:** Windows
- **Price:** Plans start at $39 a month. 16% Discount on Yearly Payment.
- **Free Trial:** 14 Days Free Trial (No Credit Card Required)

Pros	Cons
Easy-to-use, intuitive, and feature-rich interface	Free trial period for a limited period of time
Assign bugs to developers, give them due dates, and prioritize them	Prices are a bit high
A customizable Kanban workflow is available	
Integration with third-party applications	
You can add an unlimited number of projects to your account	

4) SmartSheet

Smartsheet is a spreadsheet-style bug tracking tool for businesses managing multiple projects. It helps organizations to manage workflows and helps them improve team collaboration. Smartsheet allows you to automate actions using simple rules.

Integrations: Microsoft Office 365, Microsoft Teams, Google Workspace, Box, Dropbox etc.

Customer Support: Phone, Contact Form, and Chat

Free Trial: 30 Days Free Trial

Features:

- It offers security, user management, and single sign-on capabilities for team and project management.
- Provides solution building to meet your unique needs.
- It provides customizable templates.
- Capture a screenshot or video feedback
- Powerful team collaboration
- Helps you to simplify budget and planning.
- It offers powerful analytics and reporting.
- Supports automation, input from web forms, proofing, and approvals.
- It provides customer support via Phone, Contact Form, and Chat
- Seamlessly integrates with Microsoft Office 365, Microsoft Teams, Google Workspace, Box, Dropbox, Slack, Brandfolder, PowerBI, Tableau, and Adobe Creative Cloud Extension

- You can export your file in PDF, DOC, and CSV formats
- Offers instant Email alerts
- Set Scans to run hourly, daily, and weekly
- Provide an option for Gantt Chart, Behaviour, messages, placeholders, and Internal Help Screen customization
- Supports programming languages like HTML, and PHP
- Offers WorkApps, Sheets, Reports, Control Center, Dynamic View, Home screen, Connectors, Data Shuttle, DataTable, Bridge, Governance controls, Customer managed encryption keys, Event reporting, Resource Management by Smartsheet, Brandfolder, Calendar App, Pivot App, and DataMesh
- **Supported Platforms:** Windows, and Linux
- **Price:** Plans start at $9 a month. 22% Discount on Yearly Payment.
- **Free Trial:** 30 Days Free Trial

👍 Pros	👎 Cons
Powerful, highly customizable, and very easy to use	A premium support package is available only to Business and Enterprise customers
Tools for powerful analytics and reporting	A limited range of functionality is available
A robust data protection and compliance program	Every keystroke does not update pages in real time
Compared to Microsoft Excel and Google Sheets, it's user-friendly	
An intuitive dashboard system support builder	

5) JIRA

Thousands of software professionals use JIRA as a bug-tracking tool because of its easy to use framework. JIRA is a commercial product and helps to capture and organize the team issues, prioritizing the issue and updating them with the project. It is a tool that directly integrates with the code development environments making it a perfect fit for developers as well. Due to its capability to track any kind of issues it is not just restricted to the software industry. It supports agile projects. It comes with many add-ons that make this tool more powerful than other tools

ATLASSIAN

Integrations: Jira, Markdown Macro, Automation for Jira, Excel Exporter, etc.

Customer Support: Phone, and Contact Form

Free Trial: Life Time Free Basic Plan

Features:

- It provides customer support via Phone, and Contact Form
- Seamlessly integrates with Tempo Timesheets, Jira, Markdown Macro, Automation for Jira, Better Excel Exporter, Power BI Connector, Microsoft 365, Elements Connect, Team Files, Webhook, TFS4JIRA, Yet Another Commit Checker, and Slack
- You can export your file in Word, HTML, PDF, and XML formats
- Offers instant Phone calls, and Email alerts
- Capture a screenshot or video feedback
- Powerful team collaboration
- Set Scans to run weekly, and monthly

- Provide an option for workflows customization
- Supports programming languages like HTML, Java, Javascript, C, Perl, Php, Python, R, Nyan, Ruby, Scala, and SQL
- Offers Up to 10 users, Unlimited spaces and pages, Macros, Structured page tree, Best practice templates, Page versioning, Scrum boards, Roadmaps, Reports, Project flexibility, and insights
- **Supported Platforms:** Windows, Linux, and MacOS
- **Price:** Plans start at $7.50 a month.
- **Free Trial:** Life Time Free Basic Plan

👍 Pros	👎 Cons
Tracking bugs, issues, and project progress is easy with Jira Software	The user interface is confusing
Developers, project managers, engineers, non-technical people can use Jira	Setup can be challenging
Users can create any type of issue in Jira	There are no features in the system that can be used to manage costs or assess risks
Third-party integrations make issue and project tracking easy	
Jira is extremely easy to use	
This is a great tool for small projects	

6) Marker.io

Marker.io is a visual bug reporting tool made for agencies and software development teams. Simply install the website widget and collect feedback with screenshots, annotations & technical meta-data, directly into your favourite project management tool like Jira, Trello, Asana, GitHub, ClickUp and more.

Say goodbye to messy emails, spreadsheets and powerpoint. There is a better way!

Marker.io

Integrations: Trello, Asana, ClickUp, Teamwork, Shortcut, Notion, Wrike, Monday.com etc.

Customer Support: Live Chat

Free Trial: 15 Days Free Trial

Features:

- Website widget
- Capture a screenshot, annotations or video feedback
- Powerful team collaboration
- Technical data recording (browser, OS, URL, console logs)
- 2-way sync with Jira, Trello, Asana, GitHub, ClickUp (and more)
- Browser extension / WordPress plugin / JS snippet code.
- Custom branding
- It provides customer support via Live Chat
- Seamlessly integrates with Trello, Asana, ClickUp, Teamwork, Shortcut, Notion, Wrike, Monday.com, WordPress, Jira, GitHub, GitLab, Linear, Bitbucket, LogRocket, fullstory icon, Fullstory, Browserstack, and Slack
- You can export your file in PDF, and CSV formats

- Offers instant Email alerts
- Set Scans to run hourly
- Provide an option for Jira customization
- Supports programming languages like PHP
- Offers Console recording, Client communication, Guest portal, and Browser extension
- **Supported Platforms:** Windows, and Linux
- **Price:** Plans start at $49 a month. 20% Discount on Yearly Payment.
- **Free Trial:** 15 Days Free Trial

Pros	Cons
This tool does not require any coding skills	It is not suitable for mobile applications
An easy to use application	There can be some bugs with the browser extension
An intuitive and modern user interface	It is sometimes a bit slow on the mobile version
Collaboration made simple and convenient	
Integrated with the most popular project management tools	
It provides data-rich reports to the users	

7) Zoho bug tracker

Zoho bug tracker is a powerful bug tracker that helps you to view issues filtered by priority and severity. It improves the productivity by exactly knowing which bugs are reproducible. It is an online tool that allows you to create projects, bugs, milestone, reports, documents, etc. on a single platform. With Zoho bug tracker automate reminding and notifying team members is possible.

Integrations: Zoho Desk, Zoho Analytics, Zoho People, Zoho Books, Zoho Invoice, Zoho Docs etc.

Customer Support: Chat, Email, and Phone

Free Trial: Life Time Free Basic Plan

Features:

- Bug views and integration
- Issue trackers flexible workflow
- Capture a screenshot or video feedback
- Powerful team collaboration
- Classification of issues into different categories
- It provides customer support via Chat, Email, and Phone
- Seamlessly integrates with Zoho Desk, Zoho Analytics, Zoho People, Zoho Books, Zoho Invoice, Zoho Docs, Google Drive, OneDrive, JIRA, GitHub, Bitbucket, Dropbox, and Box
- You can export your file in PDF, DOC, and CSV formats
- Offers instant Email, and Slack alerts

- Set Scans to run daily, weekly, monthly or yearly
- Provide an option for Logo, Page, and Chart customization
- Supports programming languages like C, C++, Java, PHP, and Deluge
- Offers Automate Bugs, Bugs View, File Sharing, Bugs Dashboard, Forums Discussions, User Administration, Customizations, Time tracking, and SLA Automation
- **Supported Platforms:** Windows, Mac, and Linux
- **Price:** Plans start at $4 a month.
- **Free Trial:** Life Time Free Basic Plan

👍 Pros	👎 Cons
The integration with your code repository is effortless	Does not offer unlimited storage space
The user interface is intuitive and easy to use	Integrations with non-Zoho apps are limited
An extensive set of features in the area of bug tracking and test management is available	Time tracking isn't as smooth as it could be
It also provides handy bug reports and graphs	Limited customizability
Provides comprehensive search and workflow capabilities	
It is easy to access and flexible	

8) ClickUp

ClickUp is a highly customizable bug tracking tool that allows you to create your custom views. This project management tool offers highly comprehensive time management and task management and facilitates collaboration between business units. This application allows you to assign and resolve comments to tasks. It helps you to set priorities for the work with no hassle.

Integrations: Slack, GitHub, GitLab, Webhooks, Everhour, Toggl, Harvest, Google Drive etc.

Customer Support: Chat

Free Trial: Life Time Free Basic Plan

Features:

- You can filter and search for a specific task with ease.
- It offers a wide range of templates.
- Helps you to automatically import documents from other apps.
- Capture a screenshot or video feedback
- Powerful team collaboration
- Allows you to collaborate with other people.
- It allows you to assign multiple tasks with just one mouse click.
- ClickUp enables you to sort tasks by project.
- You can sync with Google Calendar.
- It provides customer support via Chat
- Seamlessly integrates with Slack, GitHub, GitLab, Webhooks, Everhour, Toggl, Harvest, Google Drive, OneDrive, Dropbox, Outlook, and Figma

- You can export your file in HTML, PDF, and CSV formats
- Offers instant Email, and WhatsApp alerts
- Set Scans to run daily, and weekly
- Provide an option for List view, and Collapse Lists customization
- Supports programming languages like HTML, CMS, JS, Laravel, Python, and PHP
- Offers Automations, Whiteboards, Notepad, Slash Commands, Templates, Reminders, Proofing, Notifications, Goals, Mind Maps, Performance & goal tracking, and White Label
- **Supported Platforms:** Windows, and android
- **Price:** Plans start at $5 a month.
- **Free Trial:** Life Time Free Basic Plan

👍 Pros	👎 Cons
The ability to create custom views with saveable layouts	There is no table view in the mobile app
Providing collaborative features for teams to work together	It would be good if there were filters on the goals section
It has a good selection of templates to choose from	Free Plan offers limited storage and goal-setting options
This is a good solution for both teams and individuals	Onboarding takes time with lots of features
The dashboard is easy to use, and users can customize it according to their needs	
It is affordable for all sizes of teams	

9) Userback

Userback is a visual bug reporting and website feedback tool. It's the fastest way for software teams and users to report bugs from any website and application. Easily collect visual and highly contextual bug reports that help you replicate and fix issues faster. Userback automatically captures annotated screenshots, video recordings, console logs, browser info, custom data and more.

USERBACK

Integrations: Slack, Jira, Microsoft Teams, Teamwork, Trello, Basecamp, Asana, ClickUp etc.

Customer Support: Chat

Free Trial: 14 Days Free Trial

Features:

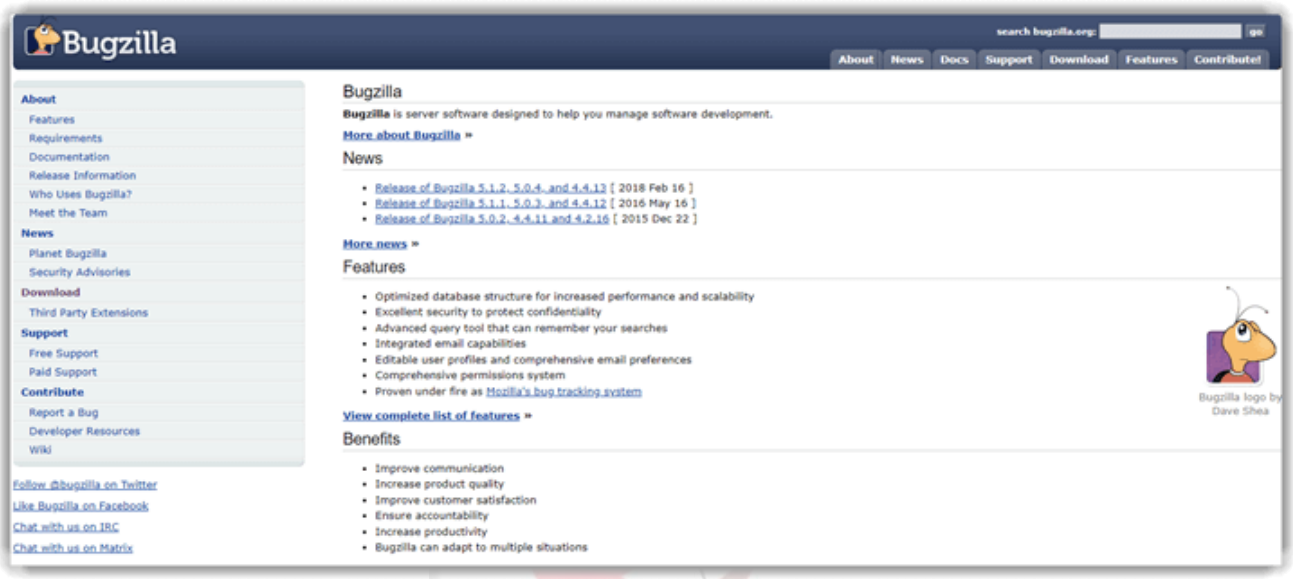
- Visual bug reporting for you and your users
- User-friendly drawing, annotation and video recording tools
- Manage bugs and user feedback in one place
- Kanban-style task board
- Create custom workflows
- Capture a screenshot or video feedback
- Powerful team collaboration
- Collect bug reports your developers will love!
- It provides customer support via Chat
- Seamlessly integrates with Slack, Jira, Microsoft Teams, Teamwork, Trello, Basecamp, Asana, ClickUp, GitHub, GitLab, MondayWrike, Azure DevOps, Monday, Zendesk, Webhook, WordPress, Zapier, and Integrately

- You can export your file in JPG, PNG, GIF, CSV, and PDF formats
- Offers instant Email, Slack, and MS Teams alerts
- Set Scans to run hourly, daily, weekly, and monthly
- Provide an option for Workflow customization
- Supports programming languages like HTML, Javascript, and Python
- Offers Unlimited Reporters, Browser Extension, Session Replay, Default Assignee, Edit Categories, Team-only Comments, Project User Access, Audience Targeting and Custom Fields, and User Identification
- **Supported Platforms:** Windows, Mac, and Linux
- **Price:** Plans start at $19 a month. 25% Discount on Yearly Payment.
- **Free Trial:** 14 Days Free Trial

Pros	Cons
There are excellent integrations with a variety of project management tools	The free trial is only available for a period of 14 days
Feedback can be shared through video and marked screenshots	Inability to capture the full-page screenshots
Any new feedback, comment, or assignment received via email, Slack, etc., will be notified immediately	
Userback's feedback widget is customizable	
Automation tools are powerful and easy to use	
Easy to use and quick to configure	

10) BugZilla

BugZilla is a popular bug tracking tool. These tool is an open source software and provides some great features like



Features:

- E-mail notification for change in code
- Reports and Charts
- Patch Viewers
- Capture a screenshot or video feedback
- Powerful team collaboration
- List of bugs can be generated in different formats
- Capture a screenshot or video feedback
- Schedule daily, monthly and weekly reports
- This bug tracking tool detect duplicate bug automatically
- Setting bug priorities by involving customers
- Predict the time a bug may get fixed

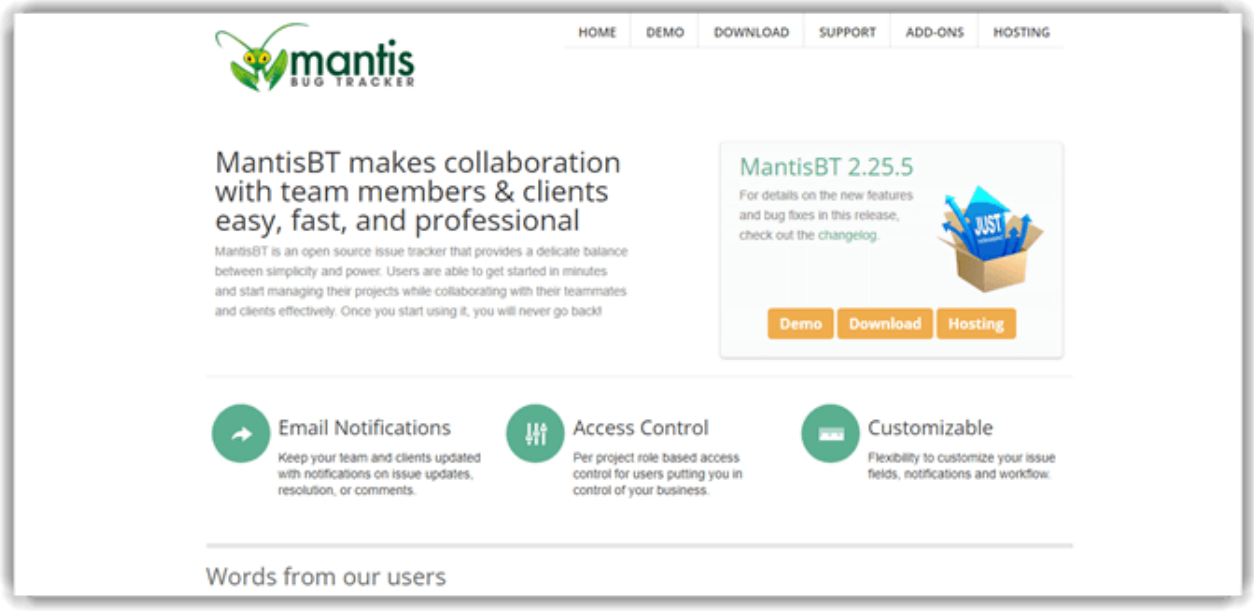
- It provides customer support via Email, and Phone
- Seamlessly integrates with CVS, Email, and ServiceNow
- You can export your file in HTML, CSV, and PDF formats
- Offers instant Email alerts
- Set Scans to run daily, weekly, and hourly
- Provide an option for templates customization
- Supports programming languages like Perl, HTML, and PHP
- Offers Advanced Search Capabilities, Bug Lists in Multiple Formats, Automatic Duplicate Bug Detection, File/Modify Bugs By Email, Time Tracking, Request System, Private Attachments and Comments, Automatic Username Completion, Drop-Down User Lists, Patch Viewer, "Watch" Other Users, Move Bugs Between Installs and Save, and Share Searches
- **Price:** Free Download

👍 Pros	👎 Cons
The most widely used open-source bug tracker	A wide range of customization is available, but it is not easy to customize
It supports localized web user interface	When reporting bugs, it is difficult to attach large files
Customized, user preferences features are available	Dashboards and insights aren't out of the box
A time tracking system is available	It sometimes takes a while to navigate between screens
An integrated email system is available	

This is a great tool for small projects

11) Mantis

If you have used other bug tracking tool, this tool can be easy to use. Mantis not only comes as a web application but also has its own mobile version. It works with multiple databases like MySQL, PostgreSQL, MS SQL and integrated with applications like chat, time tracking, wiki, RSS feeds and many more.



Features:

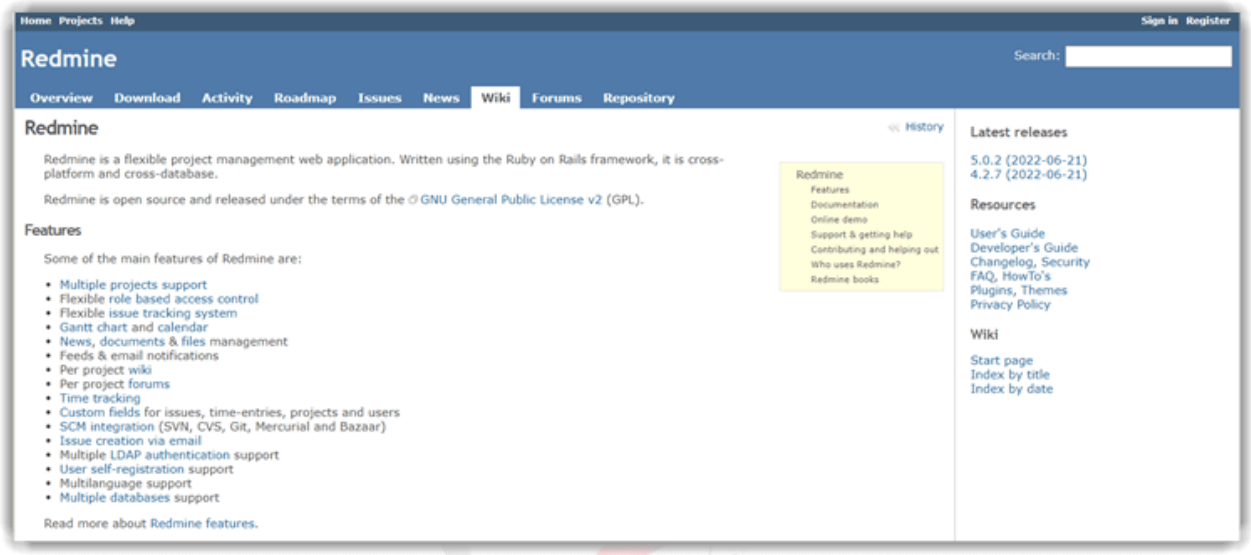
- This is a Open source tool
- Supported reporting with reports and graphs
- Source control integration
- Supports custom fields
- Supports time tracking management
- Capture a screenshot or video feedback

- Powerful team collaboration
- Multiple projects per instance
- Enable to watch the issue change history and roadmap
- Supports unlimited number of users, issues, and projects
- It provides customer support via Email
- Seamlessly integrates with Wiki, Trello, SSO, and obHelpDesk Joomla extension
- You can export your file in Word, CSV, PDF, and XML formats
- Offers instant Email alerts
- Set Scans to run daily, weekly, and monthly
- Provide an option for columns, view page, Issue Fields, and report customization
- Supports programming languages like HTML
- Offers Simple User Experience, Web Based, Available in 68 localizations, Multiple Projects per instance, Support for Projects, Sub-Projects, and Categories, Users can have a different access level per project, and Changelog Support
- **Supported Platforms:** Windows, Linux, and MacOS
- **Price:** Free Download

👍 Pros	👎 Cons
It is very convenient and easy to use the bug tracker	It’s not that easy to use with multiple tickets
It’s free and simple to manage	It is not possible to add multiple screenshots after saving
Mantis can be easily customized according to your needs	
You can control user access at a project level	

12) RedMine

It is an open source bug tracking tool that integrates with SCM (Source Code Management System) too. It supports multiple platforms and multiple data-bases while for reporting purpose, Gantt charts and calendar are used. Redmine is a project management web application developed using Ruby on Rail framework. Some of the RedMine features include



Features

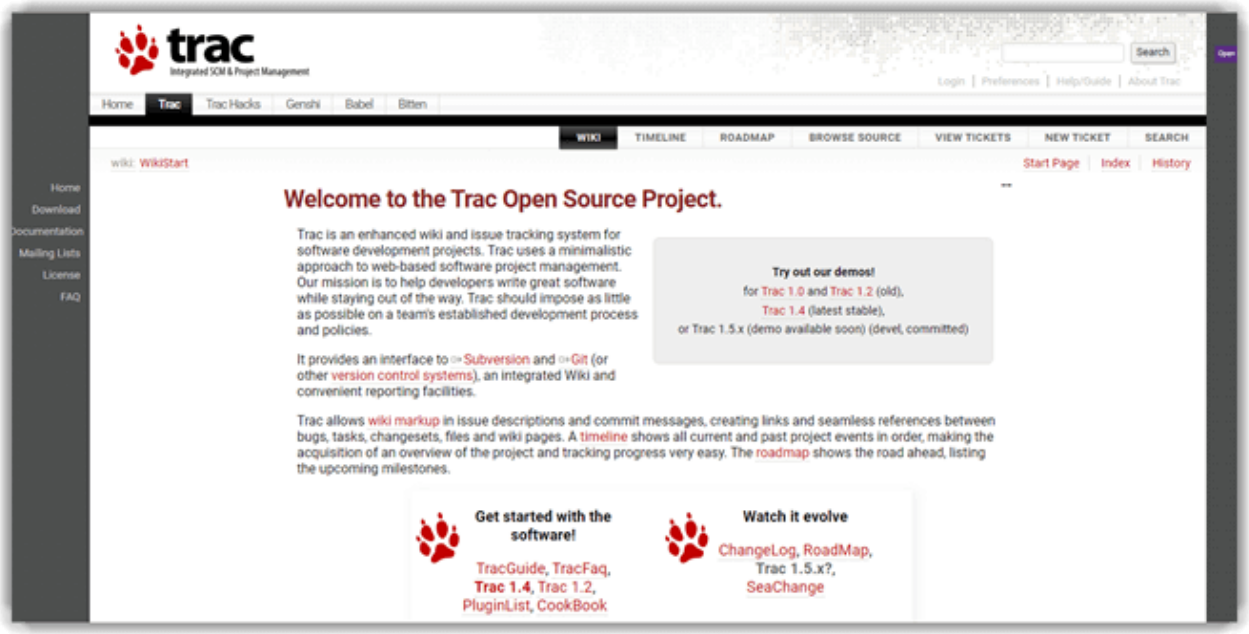
- Gantt chart and calendar
- News, document and files management
- This bug reporting tool provides SCM integration
- Issue creation via e-mail
- Capture a screenshot or video feedback
- Powerful team collaboration
- This bug tracking software supports multiple database.
- Flexible issue tracking system
- Flexible role based access control
- Multilanguage support

- It provides customer support via Chat
- Seamlessly integrates with SVN, CVS, Git, Mercurial, and Bazaar
- You can export your file in XLSX, XLS, PDF, HTML, and CSV formats
- Offers instant Feeds, and Email alerts
- Set Scans to run hourly, daily, and weekly
- Provide an option for issues, time-entries, projects, and users customization
- Supports programming languages like C, C++, C#, CSS, JavaScript, and PHP
- Offers Multiple projects support, Flexible role based access control, Flexible issue tracking system, Gantt chart and calendar, News, documents & files management, Per project wiki, Per project forums, Time tracking, Issue creation via email, and Multiple LDAP authentication support
- **Supported Platforms:** Unix, Linux, macOS, and Windows
- **Price:** Free Download

👍 Pros	👎 Cons
The Redmine task tracking system has a wide range of features	Self-installation and maintenance are required
Project progress and project management are easily manageable	It is limited to community docs for support
A free and open source project	Not suitable for all teams and projects
It includes time estimates, dependencies, Gantt charts, project wikis	The interface of the application is outdated
Provides flexible access based on a user's role	
Customizable task management	

13) Trac

Trac is a web based open source issue tracking system that developed in Python. It is the superior version of wiki and used as the issue tracking tool for software development projects. You can use it to browse through the code, view history, view changes, etc. when you integrate Trac with SCM. It supports multiple platforms like Linux, Unix, Mac OS X, Windows, etc. A time-line shows all current and past project event in order while the roadmap highlights the upcoming milestones.



Features

- It provides customer support via Ticket
- Seamlessly integrates with Perforce, Buildbot, Hudson, Jenkins, Bitten, and TortoiseSVN
- You can export your file in PDF, HTML, CSV, DOC, and XLS formats
- Capture a screenshot or video feedback
- Powerful team collaboration
- Offers instant Email, and Ticket alerts

- Set Scans to run hourly, daily, weekly, and monthly
- Provide an option for email subject, logo, icon, and email content customization
- Supports programming languages like C, C++, C#, Python, Perl, Ruby, PHP, ASP, Java, Javascript, SQL, XML, and Shell
- Offers Project issues, View progress, View your code repository online, and Manage users
- **Supported Platforms:** Unix, Linux, macOS, and Windows
- **Price:** Free Download

Pros	Cons
An intuitive user interface makes it easy to use	The project management was lacking
Live reports on a variety of topics are available	Managing large numbers of tickets is difficult
Provides an overview of where your time is being spent	
Summarized reports and insights are provided	
An all-in-one solution that combines bug tracking with wiki functionality	
Customizable task management	

14) Axosoft

It is a bug tracking system, available for hosted or on-premises software. It is a project management tool for Scrum teams. Project managers and developers can view each task, its requirement, defects and incidents, in the system, on individual filing cards, through the Scrum planning board. With Axosoft, you can manage your user stories, defects, support tickets and a real-time snapshot of your progress.

Features

- It is one of the Best Bug management tool
- Scrum planning board
- Scrum burn down charts
- Requirement Management
- Team wiki
- Data visualization
- SCM integration
- Reporting
- Help desk or incident tracking

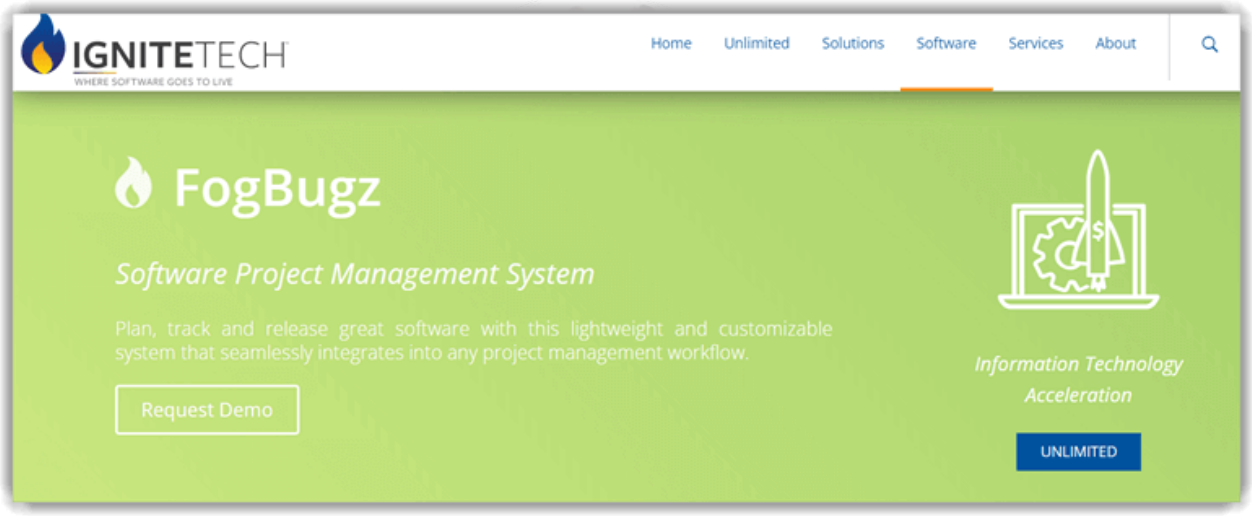
- Capture a screenshot or video feedback
- Powerful team collaboration
- It provides customer support via Ticket, and Email
- Seamlessly integrates with Bitbucket, BugHerd, Bugsnag, Clockify, GitHub, GitLab, Google Calendar, GoToMeeting, Harvest, HipChat, Hubot, Hubspot CRM, Infusionsoft, Office 365, ProjectManager.com, QlikView, Retrace, Slack, SpiraTeam, Sublime Text, TeamCity, Testuff, Timeify, Toggl, Trello, Visual Studio, Wufoo, XP-Dev, Yammer, and Zapier
- You can export your file in PDF, and CSV formats
- Offers instant SMS, and Email alerts
- Set Scans to run daily, and weekly
- Provide an option for appearance for customers, Share options, portal security role permissions, reporting, layout, Text Editor, and default settings customization
- Supports programming languages like JavaScript, PHP, XML, and Typescript
- Offers Track unlimited work items, Epics, User stories, Unlimited workflow templates, Hierarchical project tracking, and Burndown charts
- **Supported Platforms:** Windows, Linux, and MacOS
- **Price:** Free Download
- **Free Trial:** 14 Days Free Trial (No Credit Card Required)

👍 Pros	👎 Cons
The system is easy to set up and easy to use	There is a difficulty in integrating the software with other applications
A Kanban board can be used to visualize the timeline of your project	The import through CSV has some limitations
This is a powerful visual dashboard that allows you to analyze your progress in real time	

The ability to track projects and sprints is helpful

15) FogBugz

FogBugz is a web based bug tracking system that refers to defects as ‘cases’. It prioritize and organize development task in one central place. It includes a powerful search engine that enables you to instantly search the complete content of cases, wiki articles, and customer correspondence. Fogbugz supports almost all ios, Android, Blackberry, tablets, and ipads. It is a commercial product but reasonably priced.



Features

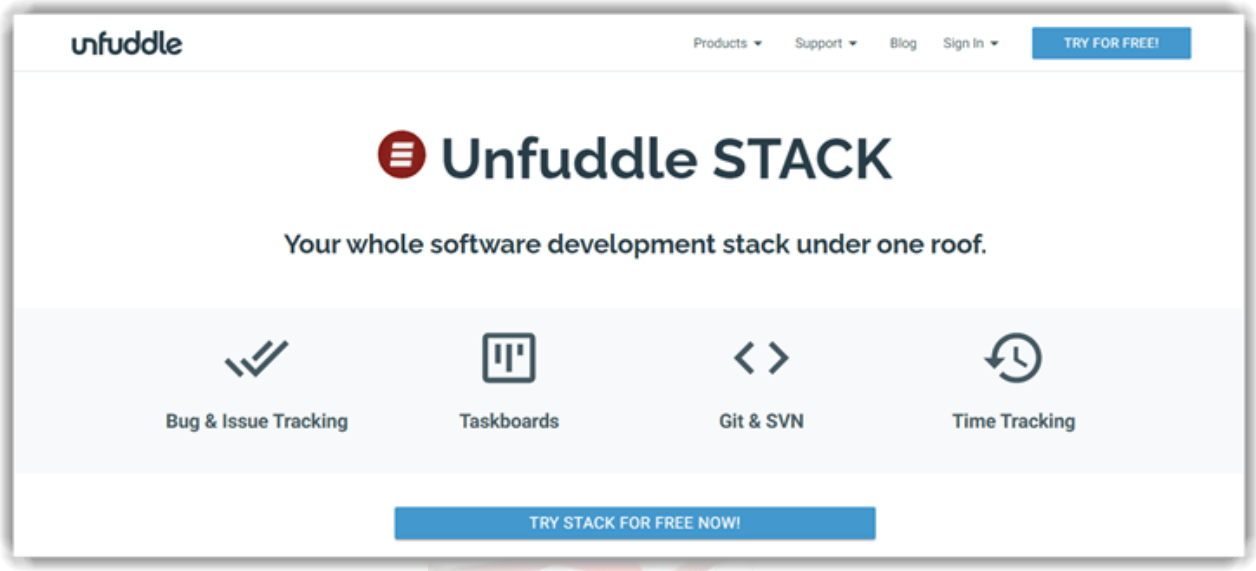
- Issue tracking
- Quick and easy case creation
- Support help desk
- Capture a screenshot or video feedback
- Powerful team collaboration

- Automated error reporting with Bugzscout
- Automatic upgrades and backups
- Project management
- Time tracking
- Integrated wiki
- It provides customer support via Chat, Contact Form, and Email
- Seamlessly integrates with Slack, Twitter, GitHub, Google Drive, Trello, Chameleon, SCCM, Zoom, Microsoft Teams, and Magiq
- You can export your file in Excel, and XML formats
- Offers instant Email alerts
- Set Scans to run daily, and monthly
- Provide an option for Everest windows, columns, formatting, and sequence customization
- Supports programming languages like HTML, Java, XML, python, .NET, and Javascript
- Offers Cloud Cost Optimization, Track all cases in one place, Easily customize case flows, Powerful search, and re-indexing
- **Supported Platforms:** Windows, Linux and MacOS
- **Price:** Request a Quote from Sales

👍 Pros	👎 Cons
Task management made easy with this application	The sorting and filtering functions do not work well
Improve your productivity by editing issues in bulk	Compared to other tools, subscription price is high
Get the latest commits from GitHub	
Creating a public wiki for the community	

16) Unfuddle

With unfuddle developers can commit source code. It can integrate with most critical tools for any software project. Unfuddle gives better security to your data as amazon provides their servers. It helps to track bugs, feature request and manage the tickets.



Features

- Bug and issue management
- Web-based subversion access
- Wiki
- Milestone tracking
- Capture a screenshot
- Powerful team collaboration
- It provides customer support via Chat, and Email
- Seamlessly integrates with Git, SVN, Eclipse, and Timedoctor
- You can export your file in XML, JSON, and CSV formats
- Offers instant Email alerts

- Set Scans to run daily, weekly, and monthly
- Provide an option for CSS, ticket, interface elements, live meetings, Mercurial, and workflows customization
- Supports programming languages like Curl, Ruby, NET, PHP, Python, C# (.NET), and JavaScript
- Offers Unlimited Git/SVN, Task Boards, Time Tracking, File Attachments, Unlimited Storage, Bug & Issue Tracking, and Git & SVN
- **Supported Platforms:** Windows, and Linux
- **Price:** Plans start at $19 a month.
- **Free Trial:** 14 Days Free Trial

👍 Pros	👎 Cons
A simple, easy-to-use program	The layout is a bit confusing
Seamless Git hosting	There is no Gantt chart for scheduling tasks
Use a custom structure to categorize your issues and tasks within your organization	
You can create as many projects as you need	

What is Security Testing?

Security testing is a process to determine whether the system protects data and maintains functionality as intended. Penetration testing or pen testing is also a type of Security testing which is performed to evaluate the security of the system (hardware, software, networks or an information system environment).

We can do security testing using both manual and automated security testing tools and techniques. Security testing reviews the existing system to find vulnerabilities.

Most of the companies perform security testing on newly deployed or developed software, hardware, and network or information system environment. But it's highly recommended by experts to make security testing as a part of information system audit process of an existing information system environment.

Here are some of the Commercial and Open Source Security Testing Tools which are popular among Security Testers.

- #1. Invicti
- #2. Acunetix
- #3. Zed Attack Proxy (ZAP)
- #4. Wfuzz
- #5. Wapiti
- #6. W3af
- #7. Vega
- #8. SQLMap
- #9. SonarQube

- #10. Nogotofail
- #11. Grabber
- #12. Arachni
- #13. Skipfish
- #14. Ratproxy

Penetration testing, also known as "pen testing," is a method of evaluating the security of a computer system, network, or web application by simulating an attack from an external or internal threat actor.

Penetration testing typically involves the use of various tools and techniques to identify vulnerabilities in the target system and attempt to exploit them to gain unauthorized access, steal sensitive information, or cause damage. The goal of the test is to identify weaknesses in the system's defences so that they can be addressed before a real attacker can exploit them.

Penetration testing is often performed by trained security professionals who have expertise in ethical hacking and the ability to identify vulnerabilities in complex systems. The results of the test are typically documented in a report that includes recommendations for improving the security posture of the target system.

SQL injection (SQLi) is a type of cyber-attack in which an attacker uses malicious SQL code to manipulate a database and gain unauthorized access to sensitive data or execute unauthorized actions.

The attack usually involves inserting malicious SQL statements into an application's input fields, which are then executed by the application's database server without proper validation. This can allow an attacker to extract confidential information, modify or delete data, or even take control of the underlying operating system.

SQL injection attacks can be prevented through proper input validation and sanitization of user inputs, and by using prepared statements or parameterized queries. It is important for software developers to be aware of the risks of SQL injection and to incorporate appropriate security measures into their applications to prevent these types of attacks. Additionally, database administrators can implement security measures such as limiting user privileges, monitoring database logs for suspicious activity, and implementing encryption to protect sensitive data.

Flow Graph

Introduction

Flow Graph is defined as a function in a program that can be represented as a control flow graph and the nodes in the flow graph are defined as program statements while the directed edges are the flow of control. A Flow Graph consists of nodes and edges. The two nodes in the Flow Graph can be either unconnected or connected by an edge in either direction or connected by an edge in all directions.

While tracing a path from a source to a sink a back edge is an edge that leads back to a node that has already been visited. The Flow Graph contains one source node and one sink.

A source node is the node that has no incoming edges while a sink node is the node with no outgoing edges. A program's function may contain more than one sink node, but this graph can be converted into a graph with only one sink. There are some languages that allow more than one source. This construct is very rare and not used in Structured Programming.

1. Used as a main tool for test case identification.
2. Represents the relationship between program segments, which is the sequence of statements having the property that if the first member of the sequence is executed then all other statements in that sequence will also be executed.
3. Nodes represent one program segment.

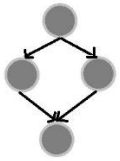
4. The area bounded by edges and nodes are called regions.

Flow Graph Symbols

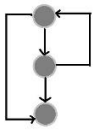
Standard notations used in constructing a flow graph are as in the following.

To indicate a Sequence:

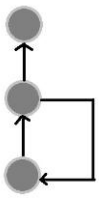
To indicate "IF-THEN-ELSE":



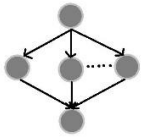
To indicate a "WHILE" Loop:



To indicate a "Repeat-Until" Loop:



To indicate a "CASE" Statement:



On a Flow Graph:

1. Arrows called edges indicates flow of control.
2. Circles called nodes indicates one or more actions.
3. Areas bounded by edges and nodes are called regions.
4. A predicate node is a node containing a condition.

What is Path Coverage Testing?

A short definition of *Path Coverage Testing*

Path coverage testing is systematic and sequential software testing, meaning the tester assesses each line of software code.

Path coverage testing falls under the types of technical testing methods in software testing. It is labour-intensive, so it's usually reserved for critical code sections.

Think of it this way, in car manufacturing, instead of just seeing if the vehicle runs, path coverage testing includes assessing all of its components (e.g., suspension, brakes, lights, etc.). That way, buyers won't have any problems with the car when it goes to market, and the company won't be liable should accidents occur or ensure too many warranties.

Read More about "Path Coverage Testing"

Path coverage testing may be time-consuming but is necessary if a developer wants to guarantee that his or her program will work without hitches.

How Does Path Coverage Testing Work?

Each software is made up of several lines of code. The more complex the program is, the more lines of code it has. Path coverage testing should test for all possible scenarios to see if it works as designed.

Take this straightforward code with two possible outcomes as an example.

input x

if x > 100 then

return true

else return false

This type of software testing should cover the two scenarios the code presents—the “if” and “else” statements—to see if the software works. If the value inputted is greater than 100, the software should return the value “true.” So if you enter “200,” you’ll get the result “true.” And that’s logical since 200 is indeed greater than 100.

If the input is less than 100, the return value should be “false.” If you enter “50,” you should get the result “false.” That’s logical, too, since 50 is less than 100.

You can’t say the software works if it doesn’t give one or both of the possible results (i.e., “true” or “false”). In our examples, 200 can’t be more than 100 or less than 100. The same can be said for 50.

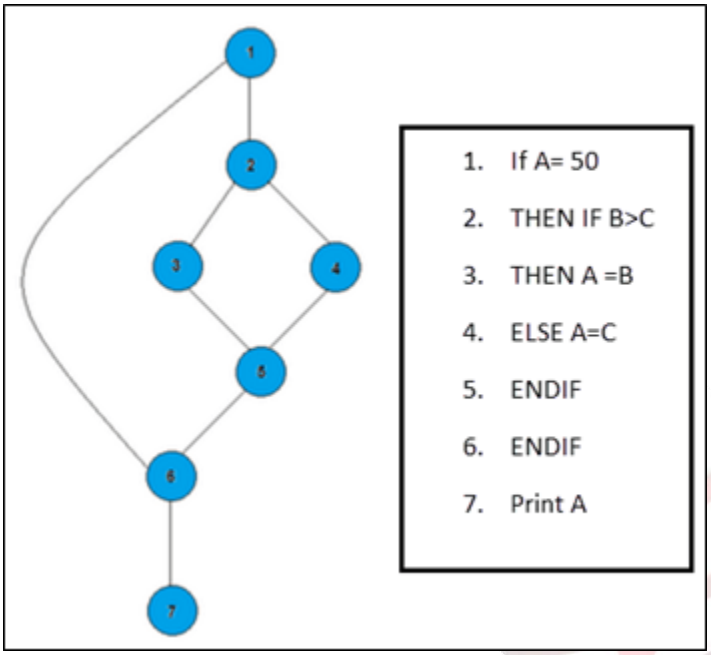
What Are the Steps in Path Coverage Testing?

Path coverage testing follows these four basic steps.

1. Draw a control graph to determine different program paths. A control graph is a graphical representation of a software’s source code.
2. Calculate the program’s cyclomatic complexity or determine the number of independent paths.

3. Find a set of paths to use as a basis.
4. Generate test cases to exercise each path.

Here's an example of a control graph.



For this control graph, you can see three paths that require testing, namely:

1. **Path 1:** Test lines 1, 2, 3, 5, 6, and 7.
2. **Path 2:** Test lines 1, 2, 4, 5, 6, and 7.
3. **Path 3:** Test lines 1, 6, and 7.

What Are the Advantages of Path Coverage Testing?

Since path coverage testing assesses every line of software code, it can:

- **Reduce redundant tests:** Because you have already tested each line of code at least once, you don't have to repeat other tests on the software just to see if it works properly.
- **Focus on program logic:** Testers can ensure the software produces logical results. It shouldn't, for instance, give two results for each input.
- **Execute test cases at least once:** The software will run all possible scenarios since you need to execute every statement.
- **Ensure the best design:** This type of testing facilitates analytical (meaning, you'll get logical results or results that make sense) versus arbitrary (has more to do with functionality than accuracy) case design.

What Are the Different Types of Coverage?

Path coverage testing can assess the following:

- **Test coverage:** Specifically measures the amount of testing performed by a set of tests. It's calculated based on the number of different structural elements in the software.

- **Statement coverage:** The percentage of executable statements a test suite has exercised. It's calculated using the formula: Statement coverage = (Number of statements exercised / Total number of statements) * 100%.

- **Decision coverage:** The percentage of decision outcomes that a test suite has exercised. 100% decision coverage implies both 100% statement coverage and 100% branch coverage. It's calculated using the formula: Decision coverage = (Number of decision outcomes exercised / Total number of decision outcomes) * 100%.

- **Branch coverage:** The percentage of branches that a test suite has exercised. 100% branch coverage implies both 100% statement coverage and 100% decision coverage.

Path Coverage (PC):

Path Coverage ensures covering of all the paths from start to end.

All possible paths are-

1A-2B-E-4F

1A-2B-E-4G-5H

1A-2C-3D-E-4G-5H

1A-2C-3D-E-4F

So path coverage is 4. Thus for the above example SC=1, BC=2 and PC=4.

Memorize these….

100% LCSAJ coverage will imply 100% Branch/Decision coverage

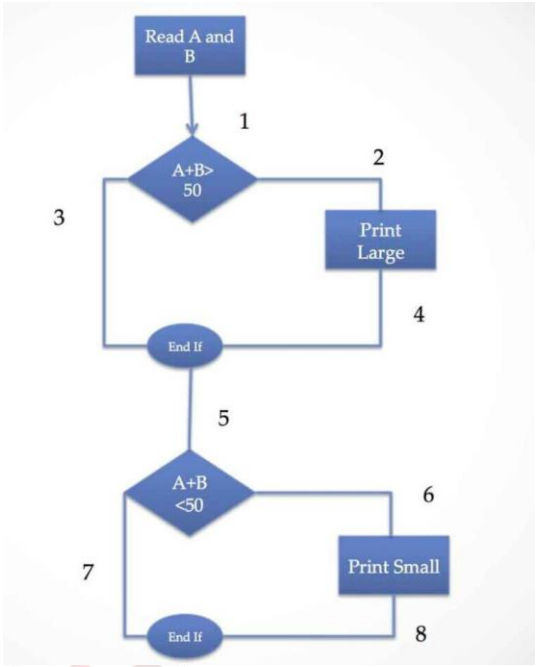
100% Path coverage will imply 100% Statement coverage

100% Branch/Decision coverage will imply 100% Statement coverage

100% Path coverage will imply 100% Branch/Decision coverage

Branch coverage and Decision coverage are same.

*LCSAJ = Linear Code Sequence and Jump



Path Coverage ensures covering of all the paths from start to end.

All possible paths are-

1-3-5-7

1-3-5-6-8

1-2-4-5-6-8

1-2-4-5-7

So Path Coverage is 4.

To calculate Branch Coverage, we have to find minimum number of paths so that all the edges should be covered and there is no single path which will ensure coverage of all the edges at once.

for covering all possible true/false decisions.

(1) 1–2-4-5-6-8

(2)1-3-5-7

So Decision or Branch Coverage is 2.

Statement Coverage

The statement coverage technique is used to design test cases for white box testing which involves the execution of all the statements of the source code at least once. This article focuses on discussing statement coverage in detail.

What is Statement Coverage?

It is one type of white box testing technique that ensures that all the statements of the source code are executed at least once. It covers all the paths, lines, and statements of a source code. It is used to design test box cases where it will find out the total number of executed statements out of the total statements present in the code.

Formula:

Statement coverage = (Number of executed statements / Total number of statements in source code) * 100

Example 1:

Read A

Read B

if A > B

  Print “A is greater than B”

else

  Print “B is greater than A”

endif

Case 1:

If A = 7, B= 3

No of statements Executed= 5

Total statements= 7

Statement coverage= 5 / 7 * 100

= 71.00 %

Case 2:

If A = 4, B= 8

No of statements Executed= 6

Total statements= 7

Statement coverage= 6 / 7 * 100

= 85.20 %

Example 2:

```
print (int a, int b)
{
   int sum = a + b;
   if (sum > 0)
      print (“Result is positive”)
  else
     print (“Result is negative”)
}
```

Case 1:

If A = 4, B= 8

No of statements Executed= 6

Total statements= 8

Statement coverage= 6 / 8 * 100

= 75.00 %

Case 2:

If A = 4, B= -8

No of statements Executed= 7

Total statements= 8

Statement coverage= 7 / 8 * 100

= 87.50 %

In the internal code structure, there are loops, arrays, methods, exceptions, and control statements. Some code would be executed based on input while some may not. Statement coverage will execute all possible paths and statements of the code

Statement coverage covers:

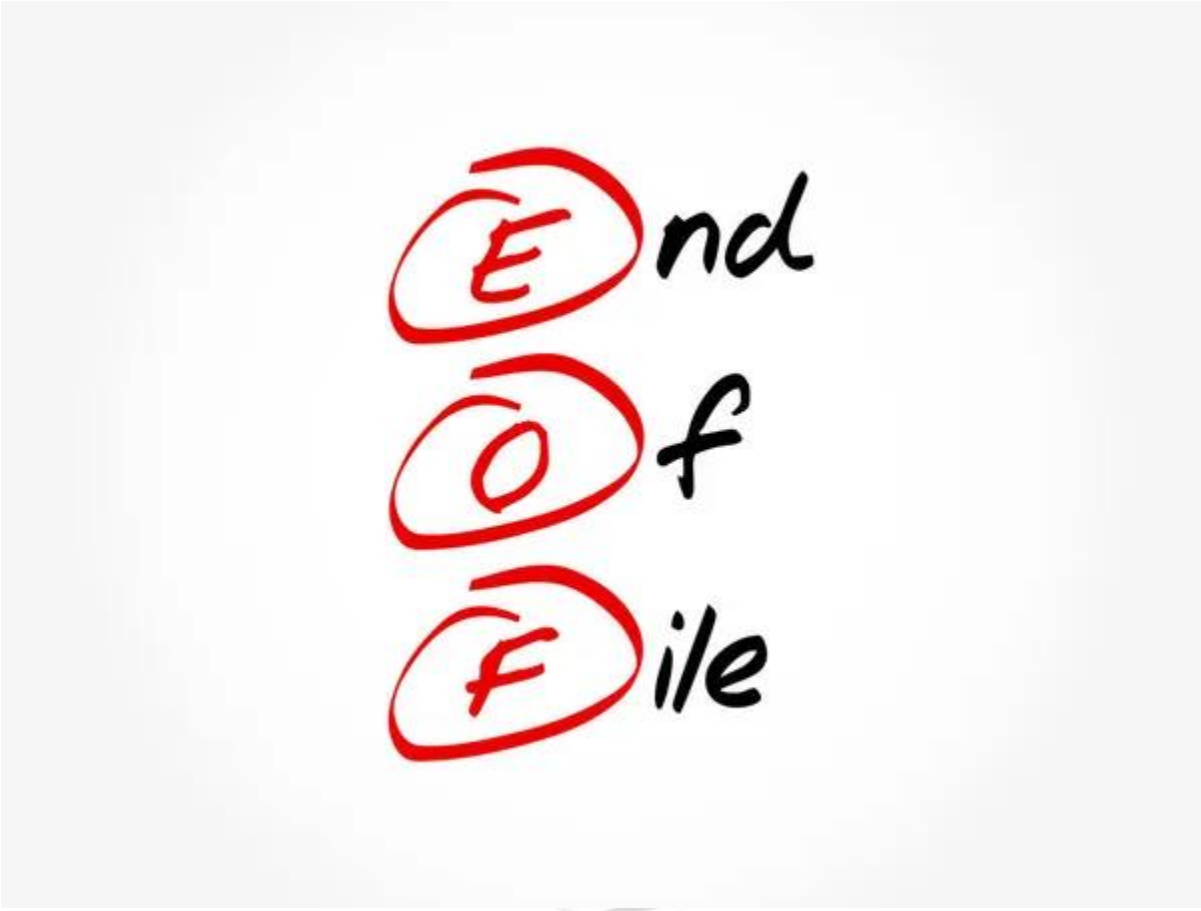
- Dead code.
- Unused statements.
- Unused branches.
- Missing statements.

Why is statement coverage used?

- To check the quality of the code.
- To determine the flow of different paths of the program.
- Check whether the source code expected to perform is valid or not.
- Tests the software's internal coding and infrastructure.

Drawback of Statement Coverage:

- Cannot check the false condition.
- Different input values to check all the conditions.
- More than one test case may be required to cover all the paths with a coverage of 100%.


End
Of
File