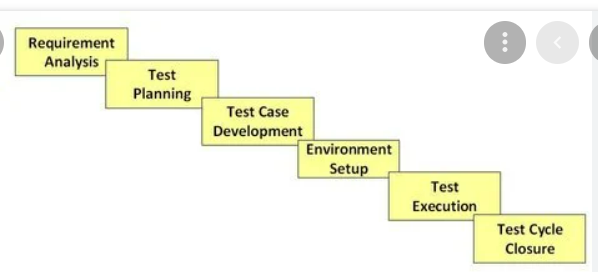
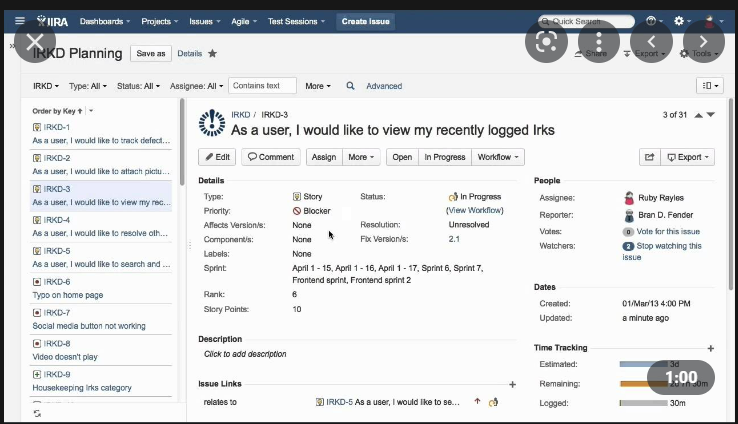
**How to write a sample test case (excel).**

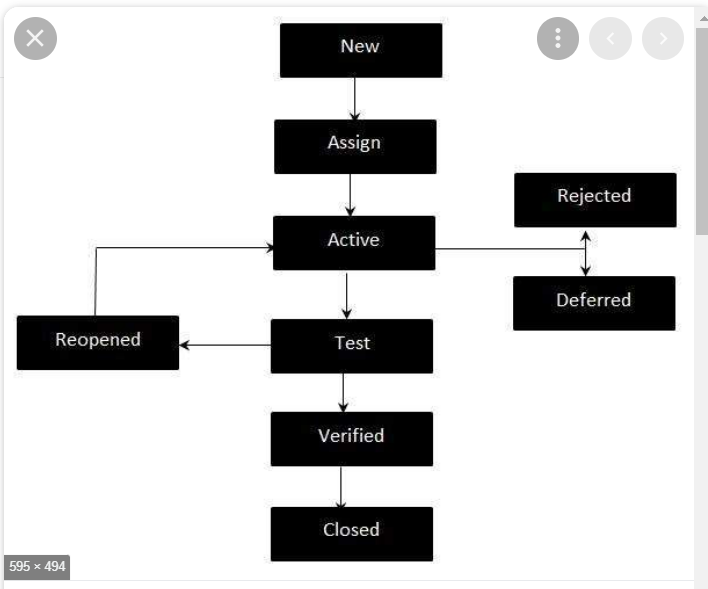
**How to do execution of that test case and take screenshots(evidence).** 



Whenever a test case is failed, we need to create a defect for that testcase and assign it to a developer so that he can fix it and assign it back to us for retest. This journey us called **Defect Life Cycle.**

**Defect Life Cycle: (**Defects are created in Jira Web Application**) (Code, configuration defect)**

Defect life cycle also called as Bug Life Cycle is the journey of a defect, which a defect goes through during its lifetime.

**Workflow:**

* **New -**Potential defect that is raised and yet to be validated.
* **Assigned -**Assigned against a development team to address it but not yet resolved.
* **Active -**The Defect is being addressed by the developer and investigation is under progress. At this stage there are two possible outcomes; viz - Deferred or Rejected.
* **Test -**The Defect is fixed and ready for testing.
* **Verified -**The Defect that is retested and the test has been verified by QA.
* **Closed -**The final state of the defect that can be closed after the QA retesting or can be closed if the defect is duplicate or considered as NOT a defect.
* **Reopened -**When the defect is NOT fixed, QA reopens/reactivates the defect.
* **Deferred -**When a defect cannot be addressed in that particular cycle it is deferred to future release.
* **Rejected -**A defect can be rejected for any of the 3 reasons; viz - duplicate defect, NOT a Defect, Non Reproducible.

While closing the defect give the comment as below:

**Code** defect. Closing the defect after retest is successful.

**Agile Methodology**

Code 1- Sprint1 – found a defect (creates a defect in jira and will assign It to developer)

Code 2 - Sprint2 -

Sprint3 -

**Defect reporting:**

Summary

Description

Steps/Steps to reproduce defect

Expected result

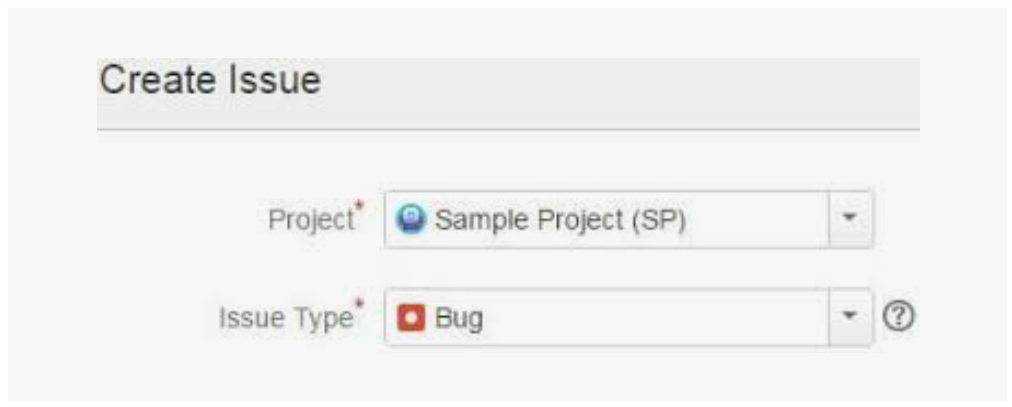
Actual result

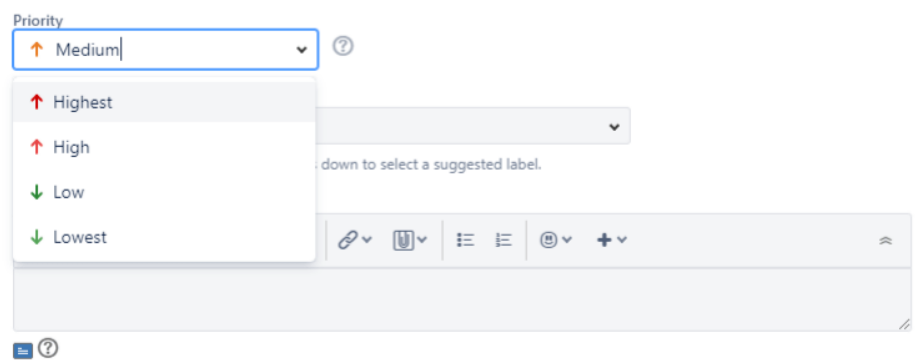
Severity

Priority

Assign it to dev.

Attachment





Severity and priority:

**Severity** is the impact of the bug/defect on the customer business workflow.

**Priority:** How soon you want the defect to be fixed or the importance given to the defect.

**Severity types:**

**Blocker** - Unable to login with correct credentials into net banking

**Critical** – Major functionality is not working

Major – send a mail but not sent popup is displayed

Minor – look and feel of the application.

**Priority types:**

**P0 – High –** dev team to fix the defect immediately

**P1 – Medium –** This defect can be fixed in coming builds

**P2 – Low –** this defect can be fixed in coming releases.

**Date: 18-08-2022**

**QA – Quality Assurance:**

* Defines the process
* Requirements, Test Planning, Test strategies etc.,
* Project Manager, product manager, Team leads
* Prevention of Defects

**QC – Quality Control:**

* Following the process defined by QC’s
* Test design, Test execution, defect reporting
* Testers
* Detection of defects

**QE- Quality Engineer/engineering:**

* Automation testers

**Responsibilities in System Testing:**

* Understanding the requirements
* Writing the cases (testcase design)
* Execution of testcases
* Finding defects and reporting them to dev. Team
* Participate in different review meetings

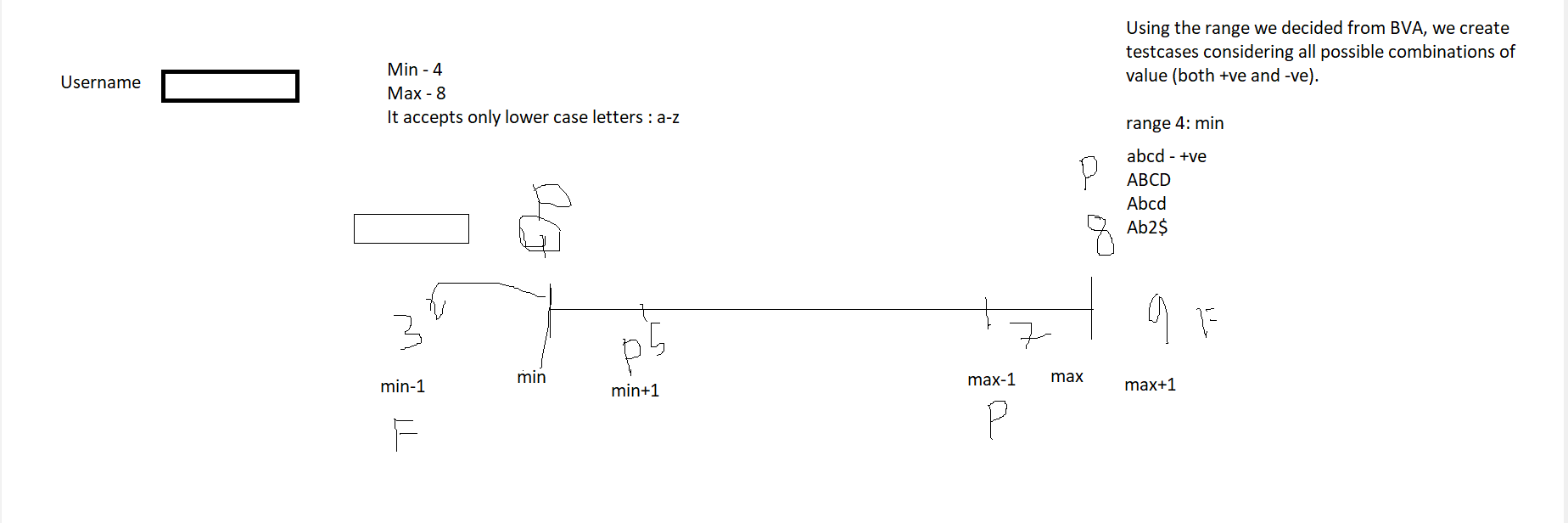
1. Writes test Automation scripts
2. Executes the automated test scripts(Smoke, regression)
3. Create reports
4. Status reports to the Management

**Test Design Techniques:**

S/W Testing techniques helps you design better test cases. Since Exhaustion testing is not possible. Testing techniques helps reduces the number of test cases to be executed while increasing test coverage. They help identify test conditions that are otherwise difficult to recognize.

* Boundary value analysis
* Equivalence Class Partitioning
* Decision Tale Based Testing
* State Transition
* Error Guessing

**Boundary value analysis:** (Range check)

****

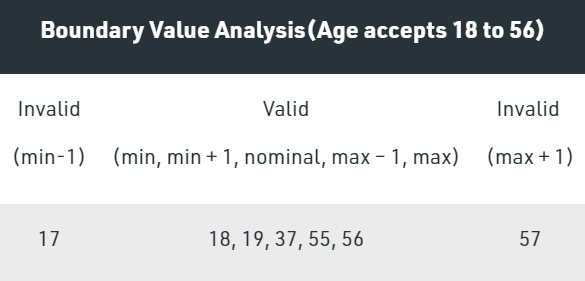
[Boundary Value Analysis](https://www.geeksforgeeks.org/boundary-value-analysis-triangle-problem/) is based on testing the boundary values of valid and invalid partitions. The behavior at the edge of the equivalence partition is more likely to be incorrect than the behavior within the partition, so boundaries are an area where testing is likely to yield defects.

It checks for the input values near the boundary that have a higher chance of error. Every partition has its maximum and minimum values and these maximum and minimum values are the boundary values of a partition.

**Note:**

* A boundary value for a valid partition is a valid boundary value.
* A boundary value for an invalid partition is an invalid boundary value.
* For each variable we check-
  + Minimum value.
  + Just above the minimum.
  + Nominal Value.
  + Just below Max value.
  + Max value.

**Example:**Consider a system that accepts ages from 18 to 56.

****

**Valid Test cases:**Valid test cases for the above can be any value entered greater than 17 and less than 57.

* Enter the value- 18.
* Enter the value- 19.
* Enter the value- 37.
* Enter the value- 55.
* Enter the value- 56.

**Invalid Testcases:**When any value less than 18 and greater than 56 is entered.

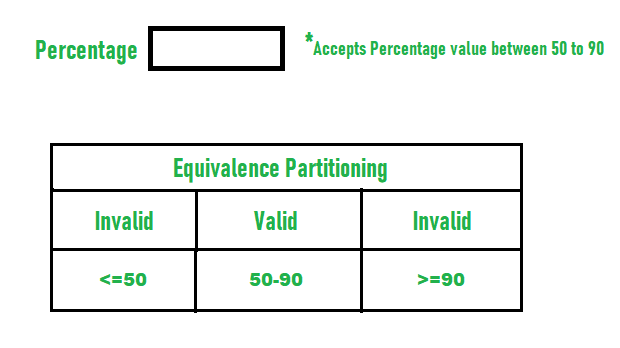
* Enter the value- 17.
* Enter the value- 57.

**Equivalence Partitioning Method** is also known as Equivalence class partitioning (ECP). It is a [software testing](https://www.geeksforgeeks.org/software-testing-basics/) technique or [black-box testing](https://www.geeksforgeeks.org/software-engineering-black-box-testing/) that divides input domain into classes of data, and with the help of these classes of data, test cases can be derived.

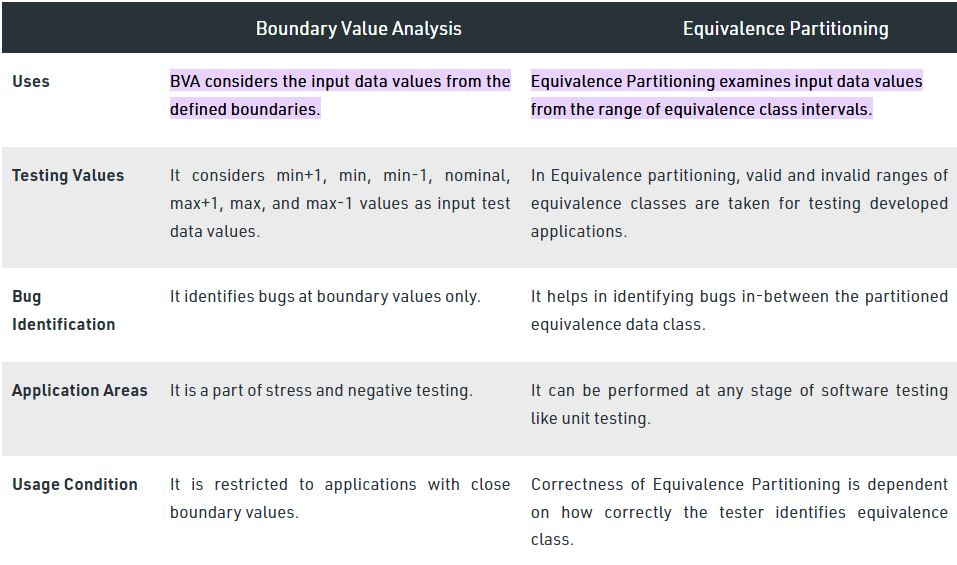
In equivalence partitioning, equivalence classes are evaluated for given input conditions. Whenever any input is given, then type of input condition is checked, then for this input conditions, Equivalence class represents or describes set of valid or invalid states.

**Guidelines for Equivalence Partitioning :**

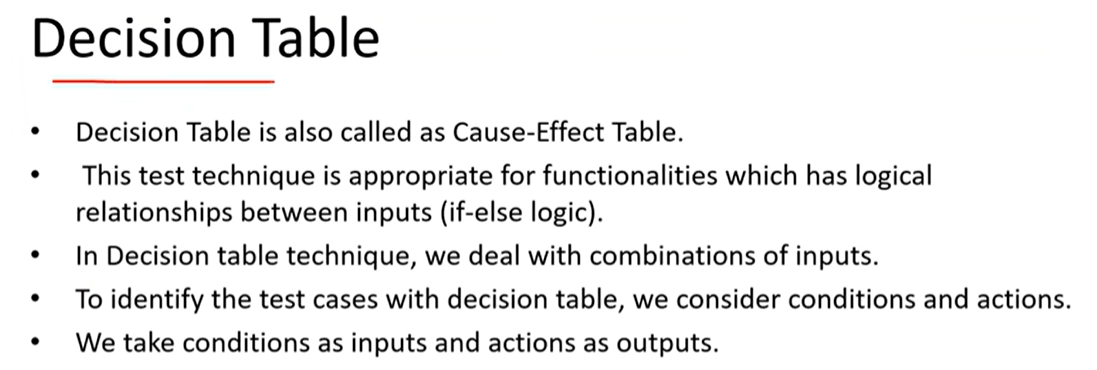
* If the range condition is given as an input, then one valid and two invalid equivalence classes are defined.
* If a specific value is given as input, then one valid and two invalid equivalence classes are defined.
* If a member of set is given as an input, then one valid and one invalid equivalence class is defined.
* If Boolean no. is given as an input condition, then one valid and one invalid equivalence class is defined.

**Example-1:**   


**Differences between BVA and Equivalence Partitioning.**

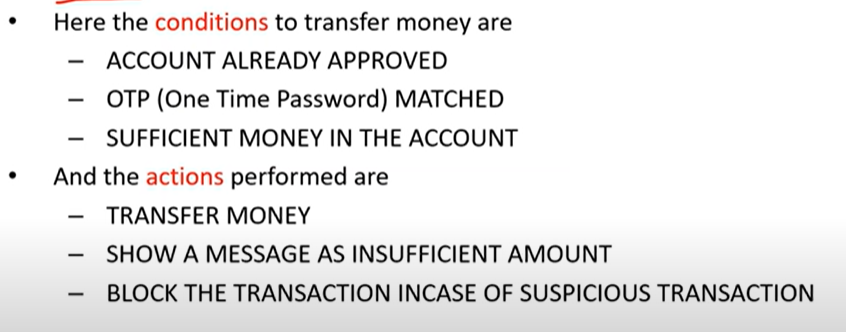


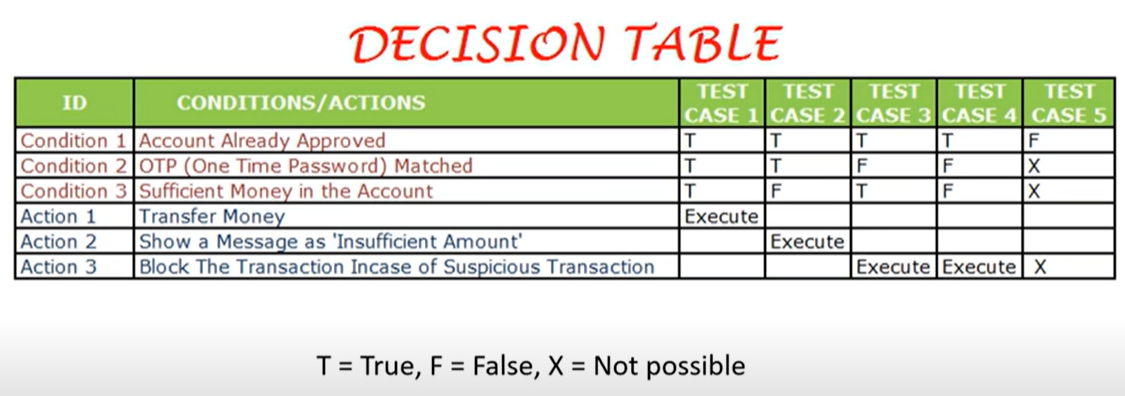
**Decision Table:**



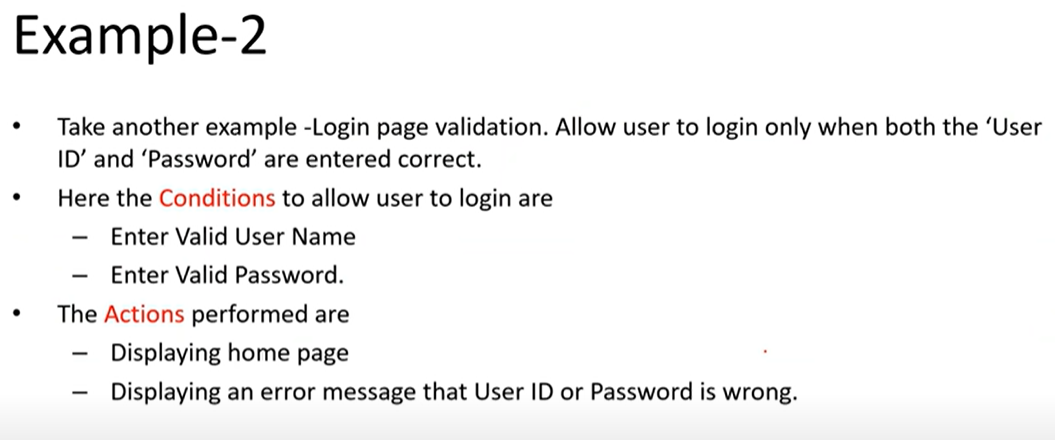
**Example : 1**

**Online transfer of money from an account which is already added & approved**

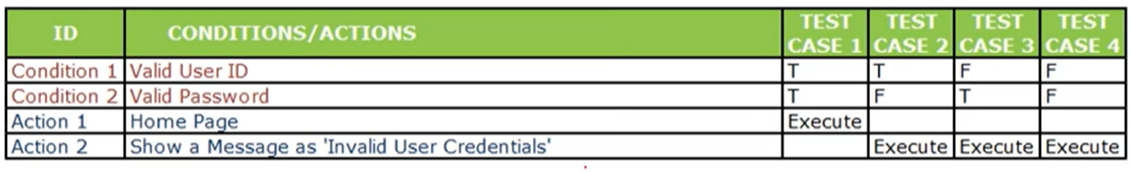




**Example:2**



**Decision Table:**

****

**State Transition Testing** is a type of software testing which is performed to check the change in the state of the application under varying input. The condition of input passed is changed and the change in state is observed.

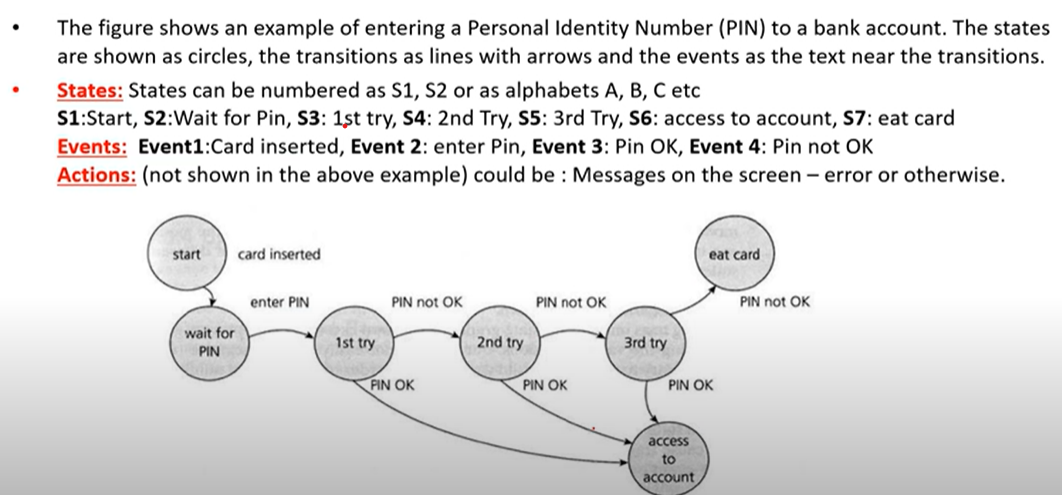
State Transition Testing is basically a black box testing technique that is carried out to observe the behavior of the system or application for different input conditions passed in a sequence. In this type of testing, both positive and negative input values are provided and the behavior of the system is observed.

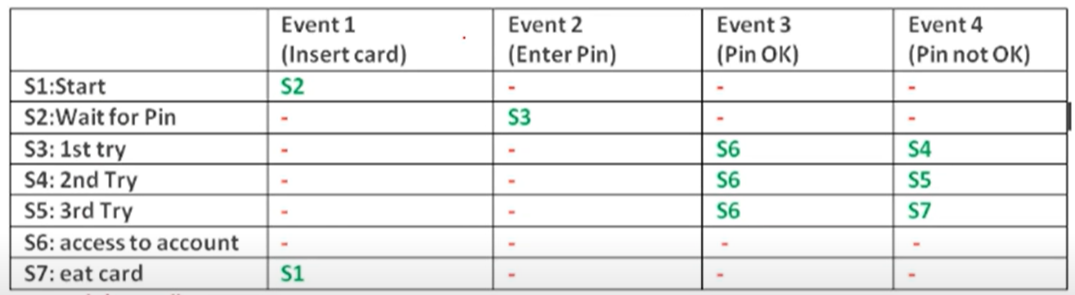
State Transition Testing is basically used where different system transitions are needed to be tested.

**Objectives of State Transition Testing:**  
The objective of State Transition testing is:

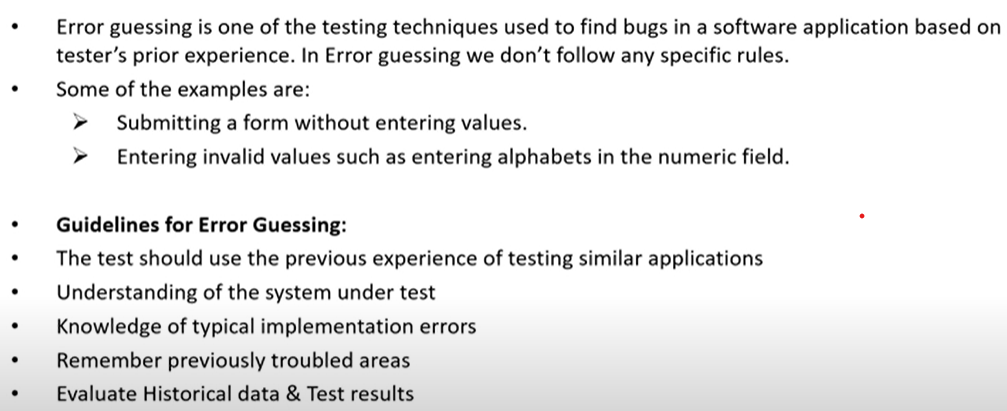
* To test the behaviour of the system under varying input.
* To test the dependency on the values in the past.
* To test the change in transition state of the application.
* To test the performance of the system.

**Example:**



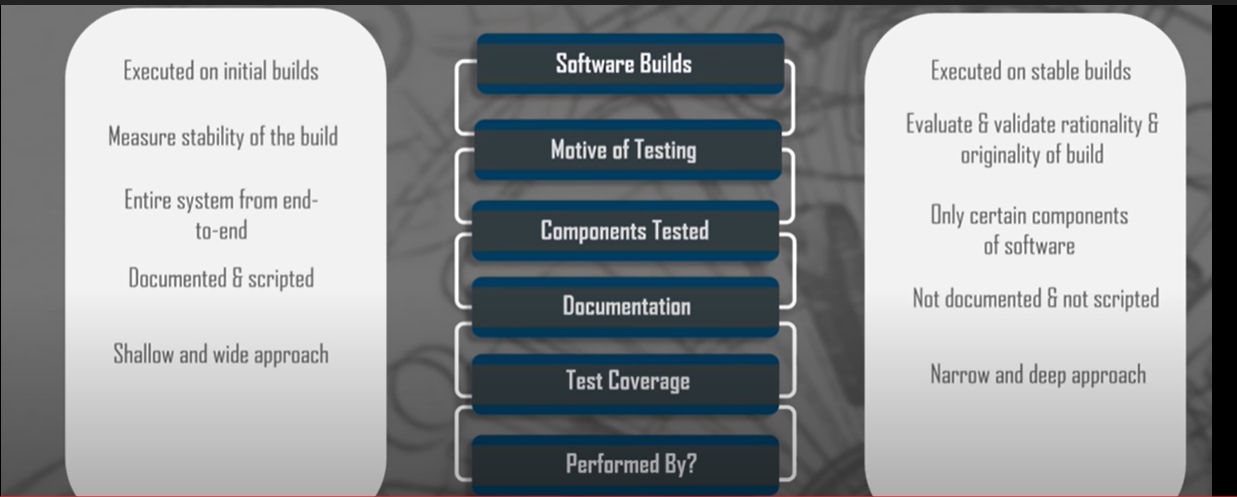


**Error Guessing**



**Smoke & Sanity Testing**

Smoke testing is done to assure that the acute functionalities of program is working fine. Sanity testing is done to check the bugs have been fixed after the build. Smoke testing is also called subset of acceptance testing. Sanity testing is also called subset of regression testing. Smoke testing is documented.



**Stages of Testing:**

Before release of any application, it undergoes a thorough testing process to ensure that the app is working in the manner in which it was intended. There are four main stages of testing that need to be completed before a program can be cleared for use: **unit testing, integration testing, system testing, and acceptance testing**. Why not include [Regression testing](https://www.seguetech.com/regression-testing-quality-assurance/)? Regression Testing is not a separate level of testing; it is just a type of testing that can be performed during any of the four main software testing stages.

## **Unit Testing**

During this first round of testing, the program is submitted to assessments that focus on specific units or components of the software to determine whether each one is fully functional. The main aim of this endeavor is to determine whether the application functions as designed. In this phase, **a unit can refer to a function, individual program or even a procedure**, and a [White-box Testing](http://stackoverflow.com/questions/402161/black-box-vs-white-box-testing) method is usually used to get the job done. One of the biggest benefits of this testing phase is that it can be run every time a piece of code is changed, allowing issues to be resolved as quickly as possible. It’s quite common for software developers to perform unit tests before delivering software to testers for formal testing.

## **Integration Testing**

Integration testing allows individuals the opportunity to combine all of the units within a program and test them as a group. This testing level is designed to **find interface defects between the modules/functions**. This is particularly beneficial because it determines how efficiently the units are running together. Keep in mind that no matter how efficiently each unit is running, if they aren’t properly integrated, it will affect the functionality of the software program. In order to run these types of tests, individuals can make use of various testing methods, but the specific method that will be used to get the job done will depend greatly on the way in which the units are defined.

## **System Testing**

System testing is the first level in which **the complete application is tested as a whole**. The goal at this level is to evaluate whether the system has complied with all of the outlined requirements and to see that it meets Quality Standards. System testing is undertaken by independent testers who haven’t played a role in developing the program. This testing is performed in an environment that closely mirrors production. System Testing is very important because it verifies that the application meets the technical, functional, and business requirements that were set by the customer.

## **Acceptance Testing**

The final level, Acceptance testing (or User Acceptance Testing), is conducted to **determine whether the system is ready for release**. During the Software development life cycle, requirements changes can sometimes be misinterpreted in a fashion that does not meet the intended needs of the users. During this final phase, the user will test the system to find out whether the application meets their business’ needs. Once this process has been completed and the software has passed, the program will then be delivered to production.

## **Software Build**

Before we delve deeper into how these three testing methods differ, we must explain what a software build is.

Can you think of the primary component of building software? Well, yes, we are talking about the code. But we know that it is not just one code that builds the entire software. There are thousands of source code files depending on the complexity of the software. Needless to mention, these source codes need to be compiled into a single executable file, which can be shared with the release team to be deployed. The process of compiling these source code files into a single file is called a software build. It is pretty literal here because it is the process where the software is built to implement it.

#### **Smoke Testing**

Smoke testing is carried out in the initial stages of the software development life cycle (SDLC). It ensures that all the core functionalities of the program are working seamlessly and cohesively. Smoke testing is executed before any functional tests are done.

The intent of smoke testing is not deep testing. It just rules out any errors in the core of the software. It largely detects and provides testing teams insights into poor builds at the primary stages to avoid bigger challenges in the later stages of the SDLC.

Also known as the build verification test, smoke testing tests for issues in critical areas of the software build than the complete application. It is carried out by both the testers and developers because of its ease and less time commitment. It is part of the rigorous testing process and uses test cases to check all important components of the build.

Smoke testing is performed when the developers provide a fresh build to the Quality Assurance teams. However, there is no limitation of it being performed only at a start of a new project. In case there are new module additions to existing functionality, smoke testing will still do its job!

***Key points:***

* They are performed to cover critical functionalities of the application.
* The objective is to verify "stability" of the application to continue with further testing.
* They are a subset of regression tests.
* Failing these tests results in instant rejection.

#### **Automation & Smoke Testing:**

Smoke testing is carried out much before automation test cases can be run on a software build. In case an organization has continuous testing built-in, smoke testing is as good as checking the successful installation of the build for the execution of the first test case. If you haven’t yet, please consider automating smoke tests since this will help run the tests quicker resulting in faster feedback look so you can work on them immediately.

#### **Sanity Testing**

Sanity testing is executed to determine if new module additions to an existing software build are stable enough to pass to the next level of testing. Also known as Surface Level Testing, this must be done to quickly evaluate the quality of regressions made to the software. In case of minor changes to the code, sanity tests further clarify whether the end-to-end testing of the build can be carried out seamlessly.

Sanity tests also ensure that any changes made do not impact other functionalities of the software build. In QA, sanity testing is part of regression testing. It is a stage after the smoke test is carried out.

***Key points:***

* They are performed to check more detailed functionalities of the application.
* The objective is to verify "rationality" of the application to continue with further testing.
* They are also a subset of regression tests.

#### **Automation & Sanity Testing:**

 Sanity testing too can be automated since it is a subset of regression testing.

#### **Regression Testing**

A superset of smoke and sanity testing, regression testing is a process that confirms whether an application functions as expected after any improvements, updates or code changes. It is responsible for the overall stability of the existing features of the build. When a modification is made, a regression test will ensure if it will remain sustainable under continuous improvements.

Regression testing is carried out after running sanity tests of any changed functionality, leading to Quality Assurance and related functionalities. It is only done by the QA team. Regression testing is the final step in the testing cycle and tests the product behavior as a whole.

When regression testing, always ensure that the code under test is under a configuration management tool and should be changed when the test is ongoing. Additionally, the QA team should only use an isolated database.

**Key points:**

* They are performed to check detailed functionalities of the application.
* The objective is to verify every existing feature of the application.
* Regression test cases should be well documented.
* They are the superset of smoke and sanity testing.

#### **Automation & Regression Testing:**

Regression test cases are ideal for automation. When taking the automation route, organizations usually automate regression testing first.

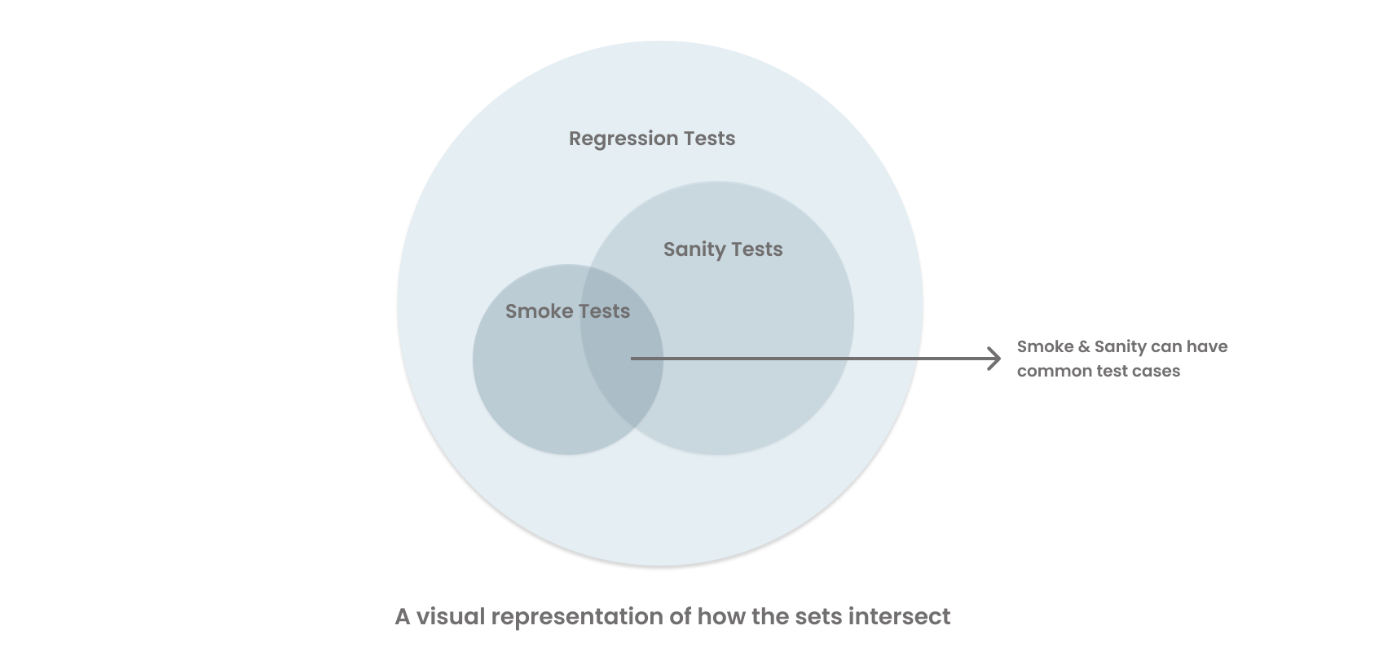
#### **In a nutshell**

Smoke and Sanity testing are less time-consuming and check the core functionalities of the code, along with checking eligibility for further tests. Regression tests on the other hand check the overall quality after a modification is made to an existing code and ensuring the change does not impact other related areas.

|  |  |  |
| --- | --- | --- |
| **Smoke Testing** | **Sanity Testing** | **Regression Testing** |
| Performed on initial builds | Performed on stable builds | Performed on stable builds |
| Tests the stability of a new build | Tests the stability of a new functionality or code changes in the existing build | Tests the functionality of all affected areas after new functionality /code changes in the existing build |
| Covers end-to-end basic functionalities | Covers certain modules, in which code changes have been made | Covers detailed testing targeting all the affected areas after new functionalities are added |
| Executed by testers & sometimes also by developers | Executed by testers | Executed by testers, mostly via automation |
| A part of basic testing | A part of regression testing | Regression Testing is a superset of Smoke and Sanity Testing |

Smoke testing can be carried out by both the developers and testers; however, sanity and regression tests are done only by the QA team and while both can be automated, regression testing is recommended to be automated.

Smoke testing is done in the initial stages of the SDLC and sanity and regression tests are usually run in the final stages. Based on time availability and requirement, the QA team must always start with smoke testing, followed by sanity and then regression tests.



In practice, all QA teams need to do smoke, sanity, and regression testing. However, with the increasing number of tests, it becomes challenging to maintain them. All these tests get executed multiple times by the QA teams and automating these tests are standard practice in a modern development process.  When opting for automation, get a tool that has the right ROI right from the start.

Selenium cannot pass data to html tags directly(sendkeys of selenium cannot interact with html codes)

But JS Script executor can do that.

1. We have to cast/covert driver object to js object
2. Js object has methods to pass calendar from internally to html code

Testing :

Functional- manual and automation

Nft

## What is Regression Testing?

**Regression Testing** is defined as a type of software testing to confirm that a recent program or code change has not adversely affected existing features. Regression Testing is nothing but a full or partial selection of already executed test cases which are re-executed to ensure existing functionalities work fine.

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

In this tutorial, we will learn

* [Need of Regression Testing](https://www.guru99.com/regression-testing.html#1)
* [How to do Regression Testing](https://www.guru99.com/regression-testing.html#2)
* [Selecting test cases for regression testing](https://www.guru99.com/regression-testing.html#3)
* [Regression Testing Tools](https://www.guru99.com/regression-testing.html#4)
* [Regression Testing and Configuration Management](https://www.guru99.com/regression-testing.html#5)
* [Difference between Re-Testing and Regression Testing](https://www.guru99.com/regression-testing.html#6)
* [Challenges in Regression Testing](https://www.guru99.com/regression-testing.html#7)
* [Practical Application of Regression Testing Example with a Video](https://www.guru99.com/regression-testing.html#8)

## Need of Regression Testing

The **Need of Regression Testing** mainly arises whenever there is requirement to change the code and we need to test whether the modified code affects the other part of software application or not. Moreover, regression testing is needed, when a new feature is added to the software application and for defect fixing as well as performance issue fixing.

## How to do Regression Testing

In order **to do Regression Testing** process, we need to first debug the code to identify the bugs. Once the bugs are identified, required changes are made to fix it, then the regression testing is done by selecting relevant test cases from the test suite that covers both modified and affected parts of the code.

Software maintenance is an activity which includes enhancements, error corrections, optimization and deletion of existing features. These modifications may cause the system to work incorrectly. Therefore, Regression Testing becomes necessary. Regression Testing can be carried out using the following techniques:



### Retest All

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

## Regression Test Selection

**Regression Test Selection** is a technique in which some selected test cases from test suite are executed to test whether the modified code affects the software application or not. Test cases are categorized into two parts, reusable test cases which can be used in further regression cycles and obsolete test cases which can not be used in succeeding cycles.

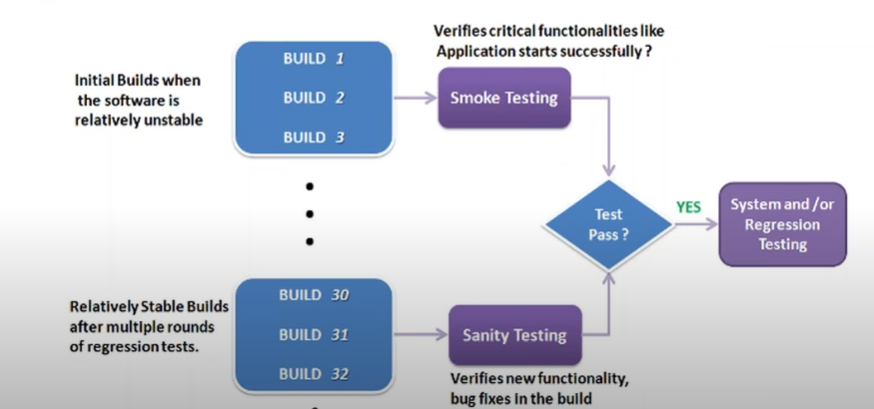
### Prioritization of Test Cases

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

## Selecting test cases for regression testing

It was found from industry data that a good number of the defects reported by customers were due to last minute bug fixes creating side effects and hence selecting the[Test Case](https://www.guru99.com/test-case.html)for regression testing is an art and not that easy.  Effective Regression Tests can be done by selecting the following test cases –

* Test cases which have frequent defects
* Functionalities which are more visible to the users
* Test cases which verify core features of the product
* Test cases of Functionalities which has undergone more and recent changes
* All Integration Test Cases
* All Complex Test Cases
* Boundary value test cases
* A sample of Successful test cases
* A sample of Failure test cases



## What is Agile Methodology?

Agile Methodology meaning a practice that promotes **continuous iteration** of development and testing throughout the software development lifecycle of the project. In the Agile model in software testing, both development and testing activities are concurrent, unlike the Waterfall model.

## Agile Model Vs Waterfall Model

Agile and Waterfall model are two different methods for software development process. Though they are different in their approach, both methods are useful at times, depending on the requirement and the type of the project.

|  |  |
| --- | --- |
| **Agile Model** | **Waterfall Model** |
| * Agile methodology definition: Agile methodologies propose incremental and iterative approach to software design | * Waterfall Model: Development of the software flows sequentially from start point to end point. |
| * The **Agile process** in software engineering is broken into individual models that designers work on | * The design process is not broken into an individual models |
| * The customer has early and frequent opportunities to look at the product and make decision and changes to the project | * The customer can only see the product at the end of the project |
| * Agile model is considered unstructured compared to the waterfall model | * Waterfall model are more secure because they are so plan oriented |
| * Small projects can be implemented very quickly. For large projects, it is difficult to estimate the development time. | * All sorts of project can be estimated and completed. |
| * Error can be fixed in the middle of the project. | * Only at the end, the whole product is tested. If the requirement error is found or any changes have to be made, the project has to start from the beginning |
| * Development process is iterative, and the project is executed in short (2-4) weeks iterations. Planning is very less. | * The development process is phased, and the phase is much bigger than iteration. Every phase ends with the detailed description of the next phase. |
| * Documentation attends less priority than software development | * Documentation is a top priority and can even use for training staff and upgrade the software with another team |
| * Every iteration has its own testing phase. It allows implementing regression testing every time new functions or logic are released. | * Only after the development phase, the testing phase is executed because separate parts are not fully functional. |
| * In agile testing when an iteration end, shippable features of the product is delivered to the customer. New features are usable right after shipment. It is useful when you have good contact with customers. | * All features developed are delivered at once after the long implementation phase. |
| * Testers and developers work together | * Testers work separately from developers |
| * At the end of every sprint, user acceptance is performed | * User acceptance is **performed** at the end of the project. |
| * It requires close communication with developers and together analyze requirements and planning | * Developer does not involve in requirement and planning process. Usually, time delays between tests and coding |

**Agile Model:**

* Client
* Business Analysts
* Development
* Testers

Sprint:

Sprint duration 2 weeks

1. **Sprint Planning Session:**

Attendees: Client, Business Analysts, Development, Testers

**BA’s** will divide the req. into stories and puts in **backlog**.

**Client**  will prioritise them.

**DEV & QA**  will go through the storis and gives the **story points**.

**Story point** will be provided in jira board.

**Acceptance Criteria:**

Usually sprint duration will be 2 weeks.

Dev, QA – If it is possible to do things for current sprint.

1. **Scrum meeting: (Scrum Master)**

Attendees: Development, Testers, Client, Business Analysts (sometimes)

What they are working on.

Blocking issues

Their progress

1. **Retrospective call:**

Attendees: Client, Business Analysts, Development, Testers

How good was the last sprint

Any changes to the last sprint if needed (improvements)

Time is sufficient or not.

Any feedback.

## **What are Agile meetings?**

To understand the ins and outs of Agile meetings, you first need to understand what Agile is.

Although [Agile](https://www.wrike.com/project-management-guide/faq/what-is-agile-methodology-in-project-management/) began as an approach for better software development, it has since gained popularity across a number of different teams and industries.

It’s often misunderstood as a specific process or framework, but that’s not entirely correct. Instead, Agile is a governing set of ideals and principles that are set forth in the [Agile manifesto](http://agilemanifesto.org/principles.html).

The full list of principles in the manifesto is worth a read, but here’s the rub: Agile involves working in shorter time frames, which offers regular intervals to reflect, adjust, and become more effective. It’s all about iterative development, rather than tackling something in one swing and realizing way too late that you’re on the wrong track.

While [Agile](https://www.wrike.com/project-management-guide/faq/how-to-run-an-agile-project/) itself isn’t a framework, there are a number of different frameworks that fall under the Agile umbrella. The two most popular are:

* [**Scrum**](https://www.wrike.com/project-management-guide/faq/what-is-scrum-in-project-management/)**:** Uses set intervals of work (called “sprints”) where teams work for a specific amount of time and then evaluate their output and processes at the end of each cycle
* [**Kanban**](https://www.wrike.com/kanban-guide/)**:** Focuses on improving workflows and visualizing a team’s work (on a tool called a [Kanban board](https://www.agilealliance.org/glossary/kanban-board/#q=~(infinite~false~filters~(postType~(~'page~'post~'aa_book~'aa_event_session~'aa_experience_report~'aa_glossary~'aa_research_paper~'aa_video)~tags~(~'kanban*20board))~searchTerm~'~sort~false~sortDirection~'asc~page~1))) continuously

In this article, we’re going to focus on the [Scrum framework](https://www.wrike.com/blog/scrum-vs-kanban-board-project-plan/), as this framework is built around several different types of Agile methodology meetings (which we’ll dig into a little later).

So now, to the big question: What is an Agile meeting?

Top of Form

Remove barriers, find clarity, exceed goals

Anything is possible with the most powerful work management software at your fingertips

Try for free

Bottom of Form

Unlike a traditional meeting where you might discuss a dozen different topics or try [virtual meeting ice breakers](https://www.wrike.com/remote-work-guide/virtual-team-building-games-ice-breakers/), Agile meetings are highly-focused and meant to be as efficient as possible without tangents or extraneous conversations. Every meeting has a clear objective that should be achieved — such as evaluating the previous sprint or [planning](https://www.wrike.com/project-management-guide/faq/what-is-planning-in-project-management/) for the next one.

## **Can Agile meetings work remotely?**

Yes, Agile meetings can be done remotely. However, they do present some hurdles. As [McKinsey and Company explains](https://www.mckinsey.com/business-functions/organization/our-insights/revisiting-agile-teams-after-an-abrupt-shift-to-remote), some challenges of remote Agile meetings include:

* Conversations can quickly become unstructured and inefficient
* Alignment of a large group doesn’t happen as naturally
* Demonstrations can be difficult
* Psychological safety can suffer, which is important for retrospectives

When leading remote Agile meetings, you’ll need to be flexible and find out what works best for your team. Maybe you need to break out into smaller groups for collaborative discussions. Or perhaps you need to add another type of meeting to your schedule (so, for example, your [sprint planning](https://www.wrike.com/scrum-guide/scrum-sprint-planning/) sessions don’t turn into problem-solving sessions).

## **Who attends Agile meetings?**

One of the best ways to ensure you’re set up for an effective and efficient meeting is to evaluate your attendee list. That way, everybody who needs to be there is present — but you aren’t clogging up other people’s calendars if they have nothing to contribute to the conversation.

[](https://web-static.wrike.com/blog/content/uploads/2020/11/A-Guide-To-Managing-Agile-Meetings-2.jpg?av=cf2bf5f52ddc2b142cf64796dbe29975)(Source: You X Ventures via *[Unsplash](https://unsplash.com/photos/vbxyFxlgpjM" \t "_blank)*)

In Agile Scrum meetings, generally, the following people are involved:

* **Development team:** Development team members are the ones who have boots on the ground and are actually doing the work, so they should be present at all of your Agile meetings.
* [**Scrum master**](https://www.wrike.com/scrum-guide/scrum-master/)**:** The [Scrum master](https://www.scrum.org/resources/what-is-a-scrum-master) is a team leader who helps everyone abide by Scrum theories and practices and enables better team interactions.
* [**Product owner**](https://www.wrike.com/product-management-guide/product-owner/)**:** The [product owner](https://www.scrum.org/resources/what-is-a-product-owner) (typically the key [stakeholder](https://www.wrike.com/project-management-guide/faq/what-is-a-stakeholder-in-project-management/)) is accountable for the work of the [Scrum team](https://www.wrike.com/scrum-guide/scrum-team-roles/) and oversees the product or output of the team.
* **Stakeholders:** [Stakeholders](https://www.scrum.org/resources/blog/scrum-who-are-key-stakeholders-should-be-attending-every-sprint-review#:~:text=According%20to%20the%20Scrum%20Glossary,Scrum%20Team%20at%20Sprint%20Review.%22) are any people who have an interest in what the Scrum team is creating, but aren’t directly involved in the process of creating it.

Don’t worry — below, we’ve spelled out which specific meetings these different players should attend.

## **What are the four types of Agile meetings?**

There are several types of Agile meetings (for example, the three amigos meeting in Agile) but here are the four most common. You’ll also hear Agile meetings referred to as “ceremonies” or “Scrum events.” As Scrum Alliance explains, there are four specific types of events in Scrum:

### **1. Sprint planning meeting**

**What it is:** The sprint planning session is when the Scrum team discusses what work they want to tackle in the next sprint and then prioritizes that work accordingly.

**Meeting goals:**

* Decide on objectives for the next sprint (i.e. what features should be developed)
* Divide tasks and responsibilities

**Who should attend:**

* Development team
* Scrum master
* Product owner

**How long it lasts:** It’s recommended that you schedule two hours of meeting time for every week of your sprint. So, if your team works in two-week sprints, your [sprint planning meeting](https://www.wrike.com/scrum-guide/scrum-sprint-planning/) should be four hours. Cap meetings at eight hours, though. Anything longer than that is too cumbersome.

### **2. Daily standup meeting**

**What it is:** The Agile [standup meeting](https://www.wrike.com/blog/stand-up-meetings-best-practices/" \t "_blank) happens every day of the sprint. It’s a quick check-in on what each team member is working on, how the process is going for them, and what stands in their way. As [Scrum Alliance explains](https://www.scrumalliance.org/about-scrum/events), this isn’t a glorified status update — it’s a chance to improve the process (and ultimately, the end product) daily before too much more work is invested.

**Meeting goals:**

* Discuss progress on work for that sprint
* Identify roadblocks and impediments to work

**Who should attend:**

* Development team

**How long it lasts:** No more than 15 minutes.

### **3. Sprint review meeting**

**What it is:** Sprint reviews are easily confused with [sprint retrospectives](https://www.wrike.com/blog/ultimate-guide-sprint-retrospectives/), but the two are different. During the sprint review, the development team presents the work that was done during the sprint (often with a demonstration), with the aim of collecting as much feedback as possible.

**Meeting goals:**

* Collect feedback

**Who should attend:**

* Development team
* Scrum master
* Product owner
* All relevant stakeholders

**How long it lasts:** It’s recommended to allocate one hour for every week of the sprint. So, if your sprint was two weeks, your sprint review should be two hours. Your sprint review should not go over four hours.

### **4. Sprint retrospective meeting**

**What it is:** During a [sprint retrospective](https://www.wrike.com/scrum-guide/sprint-retrospective/), the Scrum team focuses specifically on their work together — and not necessarily the product or output. What went well? What didn’t go well? What should they do differently in the next sprint? Each sprint retrospective should conclude with action items the team will implement to improve their collaboration.

**Meeting goals:**

* Evaluate what worked and what didn’t in the previous sprint
* Establish action items to improve the next sprint

**Who should attend:**

* Development team
* Scrum master

**How long it lasts:** It’s recommended that you allow 45 minutes for each week of your sprint. Sticking with our two-week sprint example, your sprint retrospective would be an hour and a half. Sprint retrospectives shouldn’t go longer than three hours.

**Test Closure Document:**

A test closure report is **a report that describes the testing activities performed by a QA team**. It is created once the testing phase is completed successfully or until the product being tested meets the exit criteria. Test closure reports are created by lead QA engineers and are reviewed by all other stakeholders.

Jira Tool:

**Scrum Master:**

* Creating a project
* Creating Users
* Creating Version
* Epic and Story Creation
* Sprint Creation - Stories
* Adding Story points

**Test Team/Testers:**

* **Creating Tasks (For every story)**
* **He will assign tasks**
* **Add burndown hours in jira for that task**
* **Sprint Life cycle**
* **Create defects**
* **Retesting**
* **Regression**
* **Smoke Tests**

**Tasks:**

1. **Requirement Analysis**
2. **Test Plan/Design**
3. **Test Execution**
4. **Defect Tracking**
5. **Regression**
6. **Retesting**

**Dev tasks:**

**1. Dev req. analysis**

**2.Create code**

**3. deployment**