***1. Software Quality***

*Software quality refers to the degree to which a software product meets specified requirements, satisfies user needs, and performs reliably and efficiently in its intended environment.*

***1.1 Importance of Software Quality:***

❖ *Software Quality is critically important for several reasons, affecting both the* ***success of a project and the long-term sustainability of a software product.***

❖ *The main purpose of software quality is to ensure that* ***software products are properly developed and maintained to meet the requirements.***

❖ *The three C’s of Software Quality are* ***Consistency, Completeness, and Correctness.***

***1.2 Five Views of Software Quality:***

*•****Transcendent-based:*** *Transcendental view Quality is something that*

***can be recognized but not defined****. For software, this means that certain aspects of quality are appreciated instinctively by users or developers, even if they can't be fully captured or quantified by formal metrics.*

*•****Product-based:*** *It evaluates how well it meets predefined specifications, requirements, and standards.*

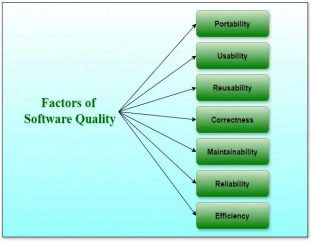
*•****User-based:*** *The user-based perspective emphasizes usability, user satisfaction, and how effectively the software supports the user’s goals.*

*•****Development-based:*** *It focuses on how quality is managed and maintained during the software development process.*

*•****Manufacturer-based:*** *The software quality focuses on the* ***quality management practices, standards, and procedures*** *followed by the software manufacturer (i.e., the company or team that develops, maintains, and distributes the software).*

***•Value-based:*** *It encourages a broader view of quality that balances* ***costs****,* ***benefits****, and* ***risk****, often by* ***aligning software development*** *with* ***business objectives*** *and* ***market demands****.*

***1.3 Factors of Software Quality:***

******

***1. Portability:*** *The software should be* ***adaptable*** *to* ***different environments or platforms*** *(e.g., different operating systems or hardware configurations) without requiring significant changes and maintaining software quality****.***

***2. Usability:*** *The software should be* ***user-friendly, intuitive, and easy to navigate****, ensuring that users can interact with it effectively.* ***3. Reusability:*** *A software has smart reusability if completely different modules of the merchandise will simply be reused to develop new merchandise.*

***4. Correctness:*** *Software is correct if completely different needs as laid out in the* ***SRS document are properly enforced.***

***1.3.1 What is an SRS Document:***

*An SRS* ***(Software Requirements Specification)*** *document is a detailed description of the software system to be developed.*

*It serves as a* ***blueprint*** *that* ***outlines the functional and non-functional requirements for the system****,* ***providing clear and comprehensive guidelines*** *for* ***developers, testers, and stakeholders*** *throughout the s****oftware development process.***

***5. Maintainability:*** *The software should be easy to maintain, update, and fix, which includes having clear, understandable code and a modular design that allows for changes to be made without affecting other parts of the system.*

***6. Reliability:*** *The software should perform* ***consistently and correctly over time, with minimal failures or bugs.*** *This also includes the software's ability to recover from failures.*

***7. Efficiency:*** *The software should make* ***optimal use of system resources*** *(such as memory, CPU, and network bandwidth) and* ***provide good performance.***

***2. Role And Responsibilities Of A Test Lead In Software Testing***

***2.1 Definition of Test Lead:***

*A Test Lead is a vital member of a software development team* ***who oversees the testing process*** *of a software product or application.*

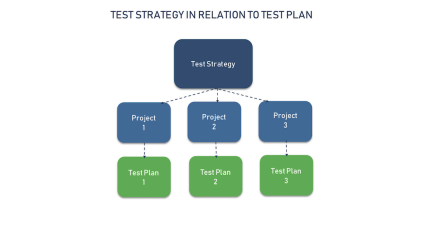
*The test Lead plays a crucial role in ensuring the successful completion of software testing activities. Test Lead responsibilities extend beyond simply overseeing the testing process* ***– also responsible for managing the testing team and guiding them toward achieving the project’s goals.***

***2.2 List of Responsibilities of Test Lead:***

***2.2.1 Planning and Strategy:***

*One of your primary responsibilities as a Test Lead is to develop a comprehensive* ***test plan and test strategy*** *for the project.*

*In software development,* ***a test plan defines your testing team’s test strategy, goals, and scope, which ultimately work together to ensure that all your software components are tested sufficiently before a release****.*

*****2.2.1.1 Testing Strategy:***

*A Test Strategy is a high-level document that outlines the overall* ***approach, objectives, scope, resources, and methods that will be used to conduct testing during a software project****.*

*A* ***Test Strategy*** *involves various activities, including* ***Test Planning, Test Design, Test Execution, Defect Management, and Reporting.*** *To achieve these objectives efficiently, several tools are used throughout the software testing life cycle.*

*These tools help automate repetitive tasks, improve testing accuracy, track defects, and generate reports.*

***2.2.1.2 Test Plan:***

***1. Define the Release Scope:*** *By defining the release scope, teams can manage* ***expectations, prioritize tasks****, and* ***ensure that the release meets business and user requirements.***

***2. Schedule Timelines:*** *A timeline helps teams* ***track progress, allocate resources, and meet deadlines.***

***3. Define Test Objectives:*** *The objectives should align with* ***project goals, product requirements, and user expectations****.*

***4. Determine Test Deliverables:*** *Test deliverables are the products of testing that* ***help track testing progress****.*

***5. Design the Test Strategy:*** *Test strategy helps determine* ***test cost, test effort, and which features will be in-scope (planned to be tested) versus out-of-scope (not planned to be tested)***

***6. Plan test environment and test data:*** *Planning a test environment guarantees precise and robust testing, The test environment includes hardware, software, and network configuration for software testing.*

***2.2. List of highly used tools :***

***Test Management Tools:*** *Jira (with Xray or Zephyr)*

***Test Automation Tools:*** *Selenium, Appium, cucumber*

***Continuous Integration/Continuous Deployment (CI/CD) Tools:*** *Jenkins* ***Performance Testing Tools:*** *Apache JMeter*

***Security Testing Tools:*** *AppScan (by IBM).*

***Test Reporting Tools:*** *Allure Framework, ExtentReports, TestLink.* ***Code Quality and Static Analysis Tools:*** *SonarQube, Checkmarx.*

***2.2.2 Team Management:***

*Another important aspect of the role of a Test Lead is* ***managing the Testing Team.***

*This includes* ***assigning tasks, coordinating resources, and monitoring the team’s progress.*** *Test leads are responsible for ensuring that all team members are adequately trained and equipped to perform their testing activities efficiently.*

***2.2.2.1 Key Categories of Team Management Tools:***

*1.* ***Task and Project Management -*** *Trello, Jira*

*2.* ***Communication and Collaboration -*** *Slack, Zoom, Microsoft Teams, Google Meet*

*3.* ***Time Tracking and Reporting -*** *Clockify, Toggl, Harvest*

*4.* ***Document and File Management -*** *Google Drive, OneDrive, Dropbox 5.* ***Team Performance and Analytics -*** *15five, TeamRetro*

*6.* ***Agile and Scrum Management -*** *Jira (Agile Boards), Targetprocess*

***2.2.3 Test Execution:***

*As a Test Lead, you are responsible for* ***overseeing the actual execution of test cases.***

*This involves coordinating with* ***software developers, business analysts, and other*** *stakeholders to prioritize and schedule testing activities.*

*You must ensure that all test deliverables are completed on time and meet the required quality standards. Additionally, you need to closely monitor the test results and provide timely feedback to the testing team.*

***Functional Testing Tools*** *- Selenium, Appium, Katalon Studio*

***Performance and Load Testing Tools*** *- Apache JMeter, Gatling* ***API Testing Tools*** *- Postman, SoapUI*

***Continuous Integration (CI) and Test Execution Tools*** *- Jenkins, CircleCI* ***Test Reporting and Management Tools*** *- TestRail, Allure Report*

***2.2.4 Defect Management Tools:*** *Defect management tools help track and manage defects found during testing. These tools allow testers and developers to collaborate on defect resolution and ensure issues are fixed before release:* ***Jira, Bugzilla, Redmine***

***2.2.5. Communication and Reporting:***

*Test Leads are responsible for ensuring effective communication between the* ***testing team, project stakeholders, and other teams*** *involved in the software development lifecycle. You need to provide* ***regular updates*** *on the testing progress,* ***highlight any issues or risks****, and ensure that* ***stakeholders*** *are kept informed about the* ***overall testing status.***

*Some software methodologies help in communication and prioritize frequent, open verbal communication over documentation.*

***This Includes:***

***Meetings:*** *Meetings are a key communication channel, and should have a clear agenda and specific time.*

***Daily scrums:*** *These short, 15-minute brainstorming meetings are a quick way to report on progress.*

***Face-to-face conversations:*** *These are considered the most efficient way to share information*

***2.2.6. Continuous Improvement:***

*As a Test Lead, it is essential to promote a culture of* ***continuous improvement*** *within the testing team. This includes encouraging team members* ***to learn new skills, adopting best practices, and staying updated*** *with the latest testing methodologies and technologies. By continuously improving the testing process, you can strive for* ***excellence and deliver high-quality software to clients.***

***2.2.7 Documentation:***

*Documenting your testing activities is essential for* ***maintaining a clear and transparent record.*** *Keep detailed records of* ***test cases, test results, bugs, and any important findings.*** *This documentation will not only help you track progress but also serve as a* ***valuable reference*** *for future projects.*

***Markdown Editors*** *- Typora, Mark Text*

***Documentation Platforms*** *- Confluence, Notion*

***Static Site Generators*** *- MkDocs*

***Visual Documentation Tools*** *- Lucidchart*

***2.2.8 Test Prioritization:***

*In the fast-paced world of software development, it’s crucial to prioritize your testing efforts.* ***Work closely with the project manager and stakeholders to determine which features or functionalities*** *need to be tested first. By prioritizing*

*your tests, you can* ***identify critical issues early on and reduce the risk of impacting the project schedule.***

***Software prioritization matrix tool:*** *Software Test Prioritization Matrix Tool, you can design a tool that allows you to* ***assign scores or weights to different aspects of your test cases*** *(such as risk, business value, and execution time). Then, the tool can* ***automatically prioritize test cases based on these scores****.*

***2.2.8.1 Define Your Prioritization Criteria:***

*We'll use the following criteria to prioritize the tests:*

*●* ***Risk*** *(1-5 scale, where 5 is the highest risk)*

*●* ***Business Value*** *(1-5 scale, where 5 is the highest value)*

*●* ***Test Execution Time*** *(1-5 scale, where 5 is the longest or most complex test)*

*●* ***Test Impact*** *(1-5 scale, where 5 is most critical to the app’s functionality)*

***2.2.9 Test Lead Role in Test Automation:***

*The responsibilities of a Test Lead in Test Automation typically go beyond just* ***managing the team****; they also need to* ***align testing efforts with business goals, maintain a strategic vision, and ensure the automation framework and tests are well-designed and sustainable****.*

***2.2.10 Performance Monitoring:***

*Performance monitoring tools are essential for* ***assessing and maintaining the health of applications, servers, and networks.***

*These tools allow teams to* ***detect performance issues, track resource usage, and analyze the responsiveness of the application*** *under different conditions (e.g., load, stress).*

*They help identify bottlenecks, diagnose slowdowns, and improve overall system performance.*

***Application Performance Monitoring (APM) Tools*** *- New Relic, AppDynamics*

***Network Performance Monitoring Tools*** *- SolarWinds Network Performance Monitor, Wireshark*

***Infrastructure Monitoring Tools*** *- Zabbix, Prometheus*

***End-User Monitoring (EUM) Tools*** *- Google Analytics*

***2.2.11 Risk Assessment:***

*Test leads should be proactive in identifying potential risks and issues that may impact the testing process. Conduct regular risk assessments and work closely with the project manager and stakeholders to develop contingency plans.*

*By anticipating and addressing potential risks early on, you can minimize project delays and ensure a smooth testing process.*

***IT and Cybersecurity Risk Assessment Tools -*** *RiskWatch, RiskLens* ***General Risk Management and Assessment Tools -*** *RiskWatch, LogicManager* ***Project and Operational Risk Assessment Tools -*** *Risk Register (Excel/Custom Tools)*

***Compliance and Regulatory Risk Assessment Tools -*** *RiskVision(formerly RiskWatch)*

***2.2.12 Stay Calm and Flexible:***

*Testing can be challenging and unpredictable, but it’s important to stay calm and flexible in the face of adversity.*

*Be prepared to adjust your plans and strategies as needed and embrace change. Your ability to adapt to new circumstances and remain composed will be crucial in successfully leading your team through challenging testing scenarios.*

***3. Role and Responsibilities of a Technical Lead***

*The software testing life cycle follows these six key testing phases, you can ensure that* ***the final product is free from errors, glitches, and other issues*** *that can hamper its performance.*

***3.1 The 6 phases of testing are***

❖ ***Requirement analysis***

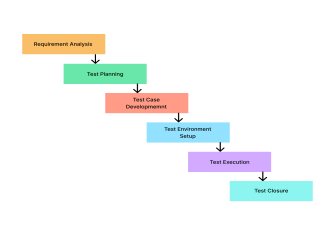
❖ ***Test planning***

❖ ***Test case development***

❖ ***Test environment setup***

❖ ***Test execution***

❖ ***Test reporting***

***3.1.1. Requirement analysis***

➔ *Requirement analysis involves* ***Identifying, Analyzing, and Documenting*** *the requirements of a software system.*

➔ ***During*** *requirement analysis, the* ***software testing team*** *works closely with the* ***Stakeholders*** *to gather information about the* ***System’s Functionality, Performance, and Usability.***

➔ *The requirements document serves as a* ***blueprint*** *for the software development team, guiding them in creating the software system.* ➔ *It also serves as a* ***reference point for the testing team****, helping them design and* ***execute effective test cases*** *to ensure the software meets the requirements.*

***3.1.2 Test planning***

***●*** *During the test planning phase, the team develops a complete plan outlining each testing process step, including* ***Identifying requirements, Determining the target audience, Selecting appropriate testing tools and Methods, Defining roles and responsibilities, and Defining timelines.***

***●*** *This phase aims to ensure that all necessary resources are in place and everyone on the team understands their roles and responsibilities.*

***●*** *A well-designed test plan* ***minimizes risks*** *by ensuring that* ***potential defects are identified early in the development cycle*** *when they are easier to fix.*

***●*** *Also, adhering to the plan throughout the testing process* ***fosters thoroughness and consistency in testing efforts*** *which can save time and cost down the line.*

***3.1.3 Test Case Development***

***●*** *During the test case* ***Development phase, the Team*** *thoroughly tests the software and considers all possible scenarios.*

***●*** *This phase involves multiple steps, including* ***Test Design, Test Case Creation, and Test Case Review.***

***●*** *Test design involves* ***Identifying the Test Scenarios*** *and* ***Defining the steps*** *to be followed during testing.*

***● Writing Test Cases*** *for each identified scenario, including input data, expected output, and the steps to be followed, involves creating test cases.* ***●*** *Test case review involves reviewing the test cases to ensure they are complete and cover all possible scenarios.*

***●*** *Also, this is the phase when the involvement of* ***Test Automation*** *can be started. You can select the test cases for test automation here. And, if automation is already a part of the STLC, and the product is suitable for testing, then the test case automation can be started too.*

***3.1.4 Test environment setup***

***●*** *Test environment setup in software testing life refers to* ***creating an environment*** *that stimulates the* ***production system where the software application is deployed.***

***●*** *A person can ensure* ***efficient and effective*** *testing activities by* ***designing the test environment correctly.***

***The setup includes:***

***■*** *Hardware*

***■*** *Software*

***■*** *Networks*

***■*** *Databases*

***●*** *When setting up test environments, we consider* ***network bandwidth, server capabilities, and storage capacity.***

***●*** *A properly set-up test environment aims to replicate real-world scenarios to* ***identify potential issues before deployment in production systems. ●*** *Testers can perform* ***Functional, Performance, or Load Testing*** *during this phase.*

***3.1.5 Test execution***

***●*** *Test execution refers to the software testing life cycle phase where created test cases are executed on the actual system being tested.*

***●*** *At this stage, testers verify whether the* ***Features, Functions, and Requirements prescribed in earlier phases perform as expected.*** *The test execution also involves the execution of automated test cases.*

***3.1.6 Test closure***

***●*** *Test closure is integral to the STLC and includes completing all planned testing activities. It includes*

***■*** *Reviewing and Analyzing test results*

***■*** *Reporting Defects*

***■*** *Identifying achieved or failed test objectives*

***■*** *Assessing test coverage*

***■*** *Evaluating exit criteria*

***● For example,*** *in* ***Agile Methodology****, test closure occurs at the end of each* ***sprint as a formal review meeting with essential stakeholders.*** *The team aims to validate that they have adequately tested all the functionality before release. They generate closure reports detailing the* ***overall product quality and bug statistics, leading to the project’s successful conclusion.***

***●*** *Effective test closure signifies that the software product is ready for delivery and can move towards launch after addressing all identified defects and weaknesses, ensuring flawless functioning and a positive user experience.*

***Software Development Life Cycle***

*SDLC is a framework that describes the activities performed at each stage of a software development project.*

*SDLC process is used by the software industry to* ***Design, Develop, and Test High-Quality Software.***

*It aims to* ***produce quality software that meets or exceeds customer expectations*** *and* ***reaches completion within time and budget.***

*SDLC, or software development life cycle is a methodology that defines the entire procedure of software development step-by-step.*

***The goal of the SDLC life cycle model is to deliver high-quality, maintainable software that meets the user’s requirements.***

******

***Formation:*** *This phase is the inception of an idea for a solution* ***that improves an existing solution or develops an entirely new one.*** *It helps define the magnitude of the project to plan resources.*

***Requirement/Planning Phase :***

*This phase gathers requirements to formulate a design plan for the software application solution. During the* ***planning phase, it is vital to include documentation to refine requirements and keep a record of the solution’s development.***

*This phase involves the* ***creation of a project charter*** *that defines technical and functional requirements.*

***Design Phase :***

*This phase focuses on the* ***design aspect of the software application solution*** *in terms of the selected technical and functional requirements and the results of the thorough analysis of the software’s viability.*

***Development Phase***

*In this phase, software engineers focus solely on* ***building a prototype*** *of the solution to perform a code review and ultimately create the solution. The team works on* ***transforming software specifications into a working and reliable solution.***

***Testing Phase***

*In the testing phase, software engineers can* ***detect defects, bugs, and errors*** *in the software solution and ultimately* ***have a quality product that meets business expectations.*** *Quality Assurance (QA) specialists perform a series of tests to evaluate the status of the solution.*

*Rigorous QA starts with the first phase and initial deliverables to establish a* ***quality-centered expectation.***

***Release Phase***

*Once the software application is fully* ***developed, tested, and ready, the software goes live and is released to the end user for actual use of the product.*** *The software is fully operational in a live environment where end-users utilize it.*

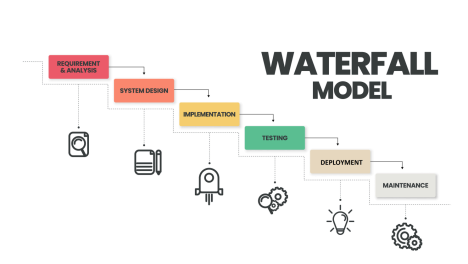
***Maintenance Phase***

*This post-release phase is tasked with* ***keeping the software completely operational, updating it to meet quality standards, and enhancing it throughout its life to ensure it continues to attract and retain users.***

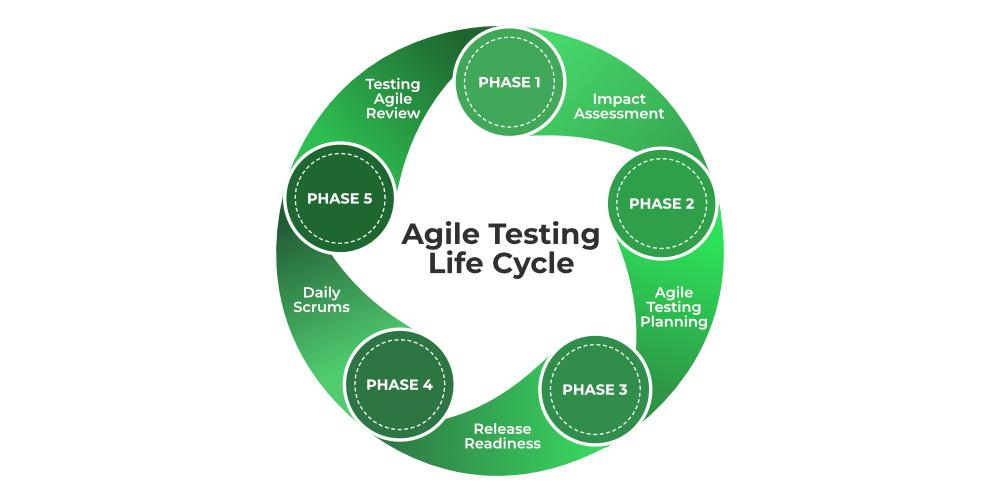
***Software Development Life Cycle Methodologies:***

***Waterfall Model****:*

*It provided a* ***linear sequential approach*** *to managing software projects. Each phase depends on deliverables from the previous one. The sequence includes* ***Requirement, Design, Development, Test, Deploy, and Maintenance.*** *This model dominated for more than 2 decades.*

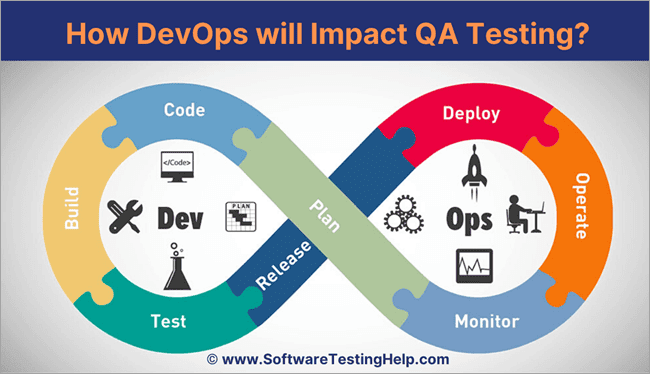
*****Agile:***

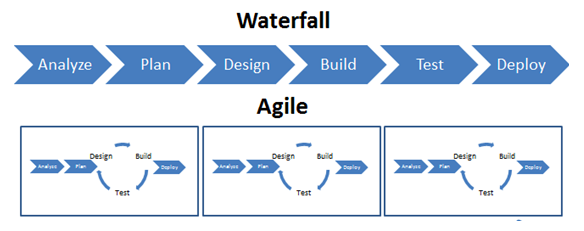
*Agile manifesto was created in 2001 and is now the most popular software development methodology. It is a* ***highly dynamic and iterative approach*** *where you do not need the complete set of requirements to start with. You can develop some features and check customer response before taking the next step.* ***Large Scrum (LeSS) and Scaled Agile Framework (SAFe) are some additional evolutions of agile methodology.***

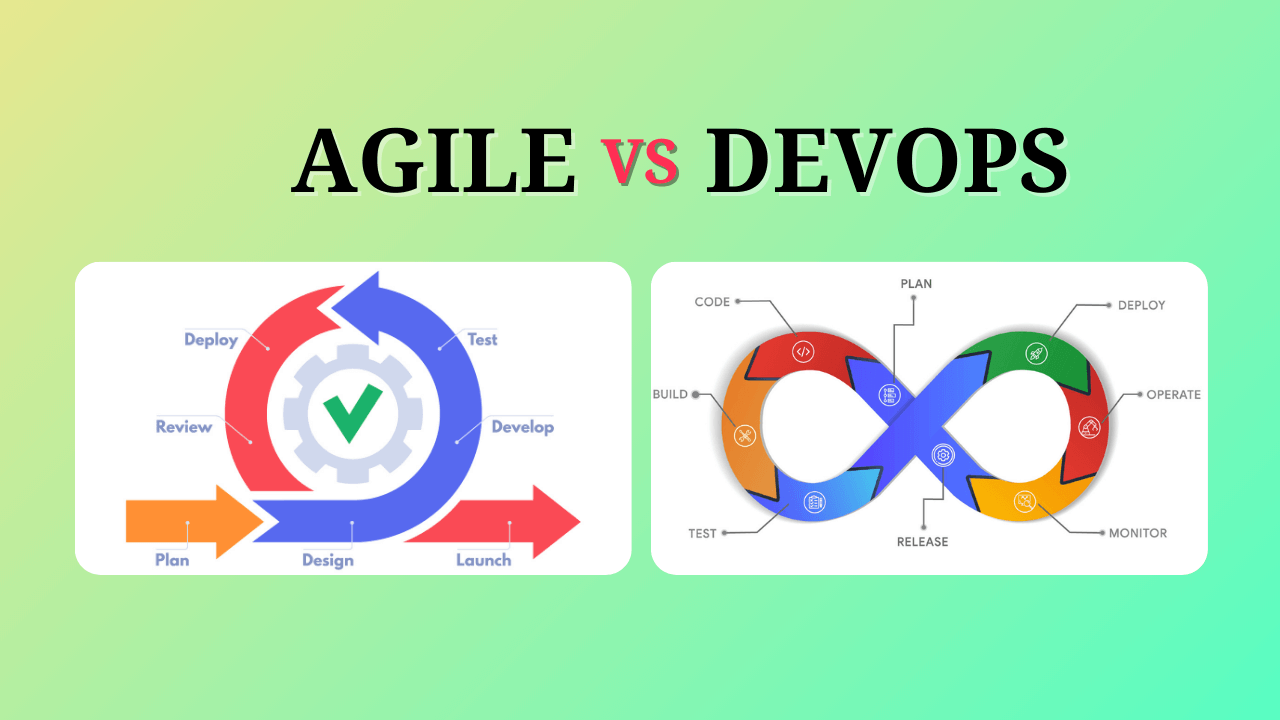
**

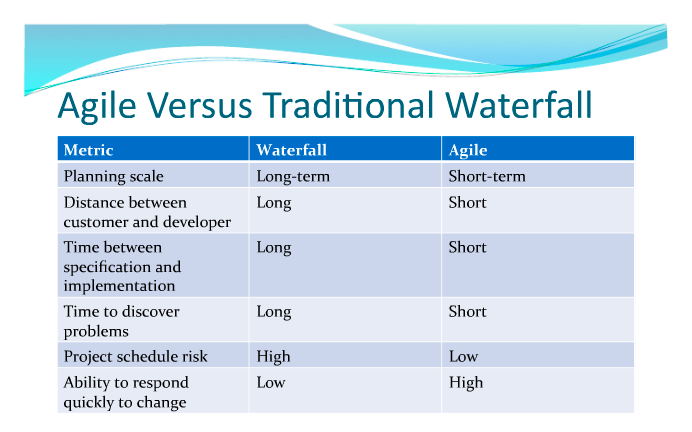
***DevOps:***

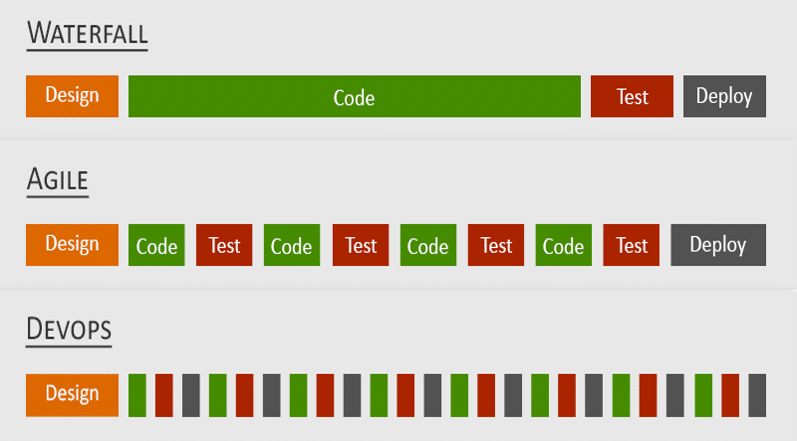
*It is an* ***Agile methodology encompassing Development (Dev) and Operations (Ops).*** *It enables end-to-end lifecycle delivery of features, fixes, and updates at frequent intervals. Agile adoption inherently left the Operations department behind with deployments piling up faster than they could be released. This trend ultimately pushed the rise of DevOps*

**

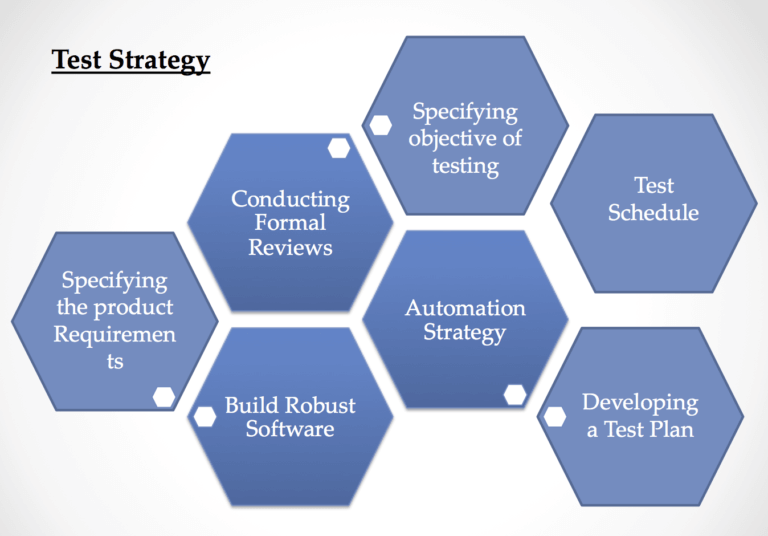
**

**

**

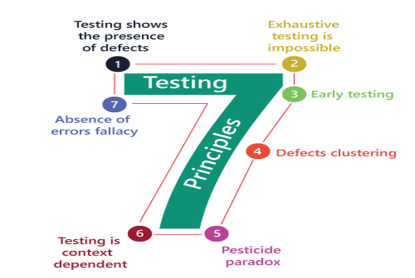
**

*Test strategy*

**

*Test Plan*

**

***Principles in Software Testing Life Cycles:***

***4.1 Testing shows the Presence of Defects***

*● The goal of software testing is to make the* ***software fail****. Software testing* ***reduces the presence of defects****. Software testing* ***talks*** *about the presence of* ***defects*** *and* ***doesn’t talk*** *about the* ***absence of defects****.*

*● Software testing can ensure that defects are present but it can not prove that software is defect-free. Even multiple tests can* ***never ensure*** *that software is* ***100% bug-free****. Testing can* ***reduce the number of defects*** *but not remove all defects.*

***4.1.2 Exhaustive Testing is not Possible***

***●*** *It is the* ***process of testing the functionality of the software*** *in all possible inputs (valid or invalid) and pre-conditions is known as exhaustive testing.* ***●*** *Exhaustive testing refers to testing a software system by trying all possible*

*combinations of inputs, paths, or states to* ***ensure the software behaves correctly in every scenario.*** *While this sounds ideal in theory, it is practically* ***impossible for most real-world software applications****, especially those with* ***complex logic, large datasets, or multiple interacting components.***

***●*** *If the software will test every test case then it will take* ***more cost, effort,*** *etc., which is* ***impractical.***

***4.1.3 Early Testing***

***●*** *To find the defect in the software, early test activity shall be started. The defect detected in the* ***early phases of SDLC will be very less expensive.*** *For better performance of software, software testing will start at the initial phase* ***i.e. testing will perform at the requirement analysis phase.***

***4.1.4 Defect Clustering***

***●*** *In a project, a small number of modules can contain most of the defects. The Pareto Principle for software testing states that* ***80% of software defects come from 20% of modules.***

***4.1.5 Pesticide Paradox***

***●*** *Repeating the same test cases, again and again, will not find new bugs. So it is necessary to* ***review the test cases and add or update test cases to find new bugs.***

***4.1.6 Testing is Context-Dependent***

***●*** *The testing approach depends on the context of the software developed.* ***Different types of software need to perform different types of testing. ● For example,*** *The testing of the* ***e-commerce site is different from the testing of the Android application.***

***4.1.7 Absence of Errors Fallacy***

***●*** *If a built software is* ***99% bug-free but does not follow the user requirement then it is unusable.*** *It is not only necessary that software is 99% bug-free but it is also* ***mandatory to fulfill all the customer requirements.***

***Role and Responsibilities of a Technical Lead***

*The software testing life cycle follows these six key testing phases, you can ensure that* ***the final product is free from errors, glitches, and other issues*** *that can hamper its performance.*

***3.1 The 6 phases of testing are***

❖ ***Requirement analysis***

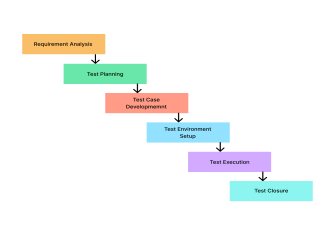
❖ ***Test planning***

❖ ***Test case development***

❖ ***Test environment setup***

❖ ***Test execution***

❖ ***Test reporting***

***3.1.1. Requirement analysis***

➔ *Requirement analysis involves* ***Identifying, Analyzing, and Documenting*** *the requirements of a software system.*

➔ ***During*** *requirement analysis, the* ***software testing team*** *works closely with the* ***Stakeholders*** *to gather information about the* ***System’s Functionality, Performance, and Usability.***

➔ *The requirements document serves as a* ***blueprint*** *for the software development team, guiding them in creating the software system.* ➔ *It also serves as a* ***reference point for the testing team****, helping them design and* ***execute effective test cases*** *to ensure the software meets the requirements.*

***3.1.2 Test planning***

***●*** *During the test planning phase, the team develops a complete plan outlining each testing process step, including* ***Identifying requirements, Determining the target audience, Selecting appropriate testing tools and Methods, Defining roles and responsibilities, and Defining timelines.***

***●*** *This phase aims to ensure that all necessary resources are in place and everyone on the team understands their roles and responsibilities.*

***●*** *A well-designed test plan* ***minimizes risks*** *by ensuring that* ***potential defects are identified early in the development cycle*** *when they are easier to fix.*

***●*** *Also, adhering to the plan throughout the testing process* ***fosters thoroughness and consistency in testing efforts*** *which can save time and cost down the line.*

***3.1.3 Test Case Development***

***●*** *During the test case* ***Development phase, the Team*** *thoroughly tests the software and considers all possible scenarios.*

***●*** *This phase involves multiple steps, including* ***Test Design, Test Case Creation, and Test Case Review.***

***●*** *Test design involves* ***Identifying the Test Scenarios*** *and* ***Defining the steps*** *to be followed during testing.*

***● Writing Test Cases*** *for each identified scenario, including input data, expected output, and the steps to be followed, involves creating test cases.* ***●*** *Test case review involves reviewing the test cases to ensure they are complete and cover all possible scenarios.*

***●*** *Also, this is the phase when the involvement of* ***Test Automation*** *can be started. You can select the test cases for test automation here. And, if automation is already a part of the STLC, and the product is suitable for testing, then the test case automation can be started too.*

***3.1.4 Test environment setup***

***●*** *Test environment setup in software testing life refers to* ***creating an environment*** *that stimulates the* ***production system where the software application is deployed.***

***●*** *A person can ensure* ***efficient and effective*** *testing activities by* ***designing the test environment correctly.***

***The setup includes:***

***■*** *Hardware*

***■*** *Software*

***■*** *Networks*

***■*** *Databases*

***●*** *When setting up test environments, we consider* ***network bandwidth, server capabilities, and storage capacity.***

***●*** *A properly set-up test environment aims to replicate real-world scenarios to* ***identify potential issues before deployment in production systems. ●*** *Testers can perform* ***Functional, Performance, or Load Testing*** *during this phase.*

***3.1.5 Test execution***

***●*** *Test execution refers to the software testing life cycle phase where created test cases are executed on the actual system being tested.*

***●*** *At this stage, testers verify whether the* ***Features, Functions, and Requirements prescribed in earlier phases perform as expected.*** *The test execution also involves the execution of automated test cases.*

***3.1.6 Test closure***

***●*** *Test closure is integral to the STLC and includes completing all planned testing activities. It includes*

***■*** *Reviewing and Analyzing test results*

***■*** *Reporting Defects*

***■*** *Identifying achieved or failed test objectives*

***■*** *Assessing test coverage*

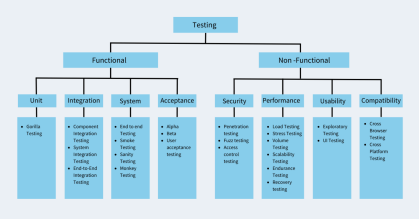
***■*** *Evaluating exit criteria*

***● For example,*** *in* ***Agile Methodology****, test closure occurs at the end of each* ***sprint as a formal review meeting with essential stakeholders.*** *The team aims to validate that they have adequately tested all the functionality before release. They generate closure reports detailing the* ***overall product quality and bug statistics, leading to the project’s successful conclusion.***

***●*** *Effective test closure signifies that the software product is ready for delivery and can move towards launch after addressing all identified defects and weaknesses, ensuring flawless functioning and a positive user experience.*

***Types of Testing***

*There are various software testing types for manual testing and automated testing, and different* ***testing methodologies such as black-box testing, white-box testing, and gray-box testing.*** *During the testing process, testers may use various* ***testing types such as functional testing, performance testing, security testing, and usability testing.***

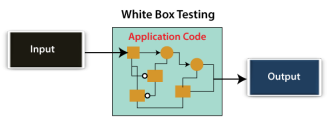
******

***What are the Software Testing Techniques and How are They Different from Testing Types?***

***White Box Testing***

➔ *White box testing techniques* ***analyze the internal structures*** *of the* ***used data structures, internal design, code structure, and the working of the software.***

➔ *It is also called* ***glass box, clear box testing, or structural testing****. White Box Testing is also known as* ***transparent testing or open box testing.*** ➔ *The developer will execute the complete white-box testing for the particular software and* ***send the specific application to the testing team.*** ➔ *The purpose of implementing white box testing is to emphasize the* ***flow of inputs and outputs over the software and enhance the security of an application.***

******

*White box testing is a software testing technique that involves testing the internal structure and workings of a software application.*

*The tester has access to the* ***source code*** *and uses this knowledge* ***to design test cases that can verify the correctness of the software at the code level.***

***5.1.1.2.2 Black Box Testing***

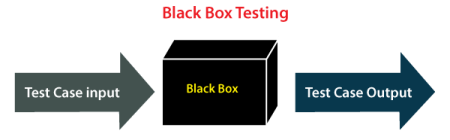
*In this testing, the test engineer will* ***analyze the software against requirements****,* ***identify the defects or bugs, and send it back to the development team.***

*Then, the developers will fix those defects,* ***do one round of White box testing****, and send it to the testing team.*

*Here,* ***fixing the bugs*** *means the* ***defect is resolved****, and the particular feature is working according to the given requirement.*

*The main objective of implementing black box testing is to specify the business needs or the customer's requirements.*

*In other words, we can say that black box testing is a* ***process of checking the functionality of an application*** *as per the customer's requirement. The* ***source code is not visible*** *in this testing; that's why it is known as black-box testing.*

**

***5.1.1.2.2.1 Types of Black Box Testing***

*Black box testing is further categorized into two parts, which are discussed below:*

*● Functional Testing*

*● Non-function Testing*

***Functional Testing***

*The test engineer will* ***check all the components systematically against requirement specifications known*** *as functional testing. Functional testing is also known as* ***Component testing****.*

*In functional testing,* ***all the components are tested by giving the value, defining the output, and validating the actual output with the expected value.***

*Functional testing is a part of* ***black-box testing*** *as its emphasizes application requirements rather than actual code. The test engineer has to test only the program instead of the system.*

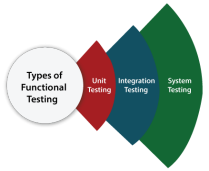
***Types of Functional Testing***

*Functional testing is also classified into various categories.*

*● Unit Testing*

*● Integration Testing*

*● System Testing*

**

***Unit Testing***

*Unit testing is the* ***first level*** *of functional testing to test any software. In this, the test engineer will* ***test the module of an application independently or test all the module functionality is called unit testing.***

***Integration Testing***

*Once we have successfully implemented the unit testing, we will go through integration testing. It is the* ***second level*** *of functional testing, where we test the* ***data flow between dependent modules or interfaces between two features is called integration testing.***

*The purpose of executing the integration testing is to test the* ***statement's accuracy between each module.***

***Types of Integration Testing***

*Integration testing is also further divided into the following parts:*

***1. Incremental Integration Testing***

*Whenever there is a* ***clear relationship*** *between modules, we go for* ***incremental integration testing.*** *Suppose, we take two modules and analyze the data flow between them if they are working fine or not.*

*If these modules are working fine, then we can add one more module and test again. And we can continue with the same process to get better results.*

*In other words, we can say that incrementally adding up the modules and testing the data flow between the modules is known as Incremental integration testing.*

***Types of Incremental Integration Testing***

*Incremental integration testing can further be classified into two parts, which are as follows:*

*Top-down Incremental Integration Testing*

*Bottom-up Incremental Integration Testing*

***Top-down Incremental Integration Testing***

*In this approach, we will add the modules step by step or incrementally and test the data flow between them. We have to ensure that the* ***modules we are adding are the child of the earlier ones.***

***Bottom-up Incremental Integration Testing***

*In the bottom-up approach, we will add the modules incrementally and check the data flow between modules. And also, ensure that the* ***module we are adding is the parent of the earlier ones.***

***2. Non-Incremental Integration Testing/ Big Bang Method***

*Whenever the* ***data flow is complex and very difficult*** *to classify a parent and a child, we will go for the* ***non-incremental integration approach****. The non-incremental method is also known as the* ***Big Bang method.***

***System Testing***

*Whenever we are done with the unit and integration testing, we can proceed with the system testing.*

*In system testing, the* ***test environment is parallel to the production environment****. It is also known as* ***end-to-end testing.***

*In this type of testing, we will undergo each attribute of the software and test if the* ***end feature works according to the business requirement****. An analysis of the software product as a complete system.*

***Non-functional Testing***

*It provides* ***detailed information on software product performance and used technologies.***

*Non-functional testing will help us* ***minimize the risk of production and related costs of the software.*** *Non-functional testing combines* ***performance, load, stress, usability, and compatibility testing.***

***Types of Non-functional Testing***

*1. Performance Testing*

*2. Usability Testing*

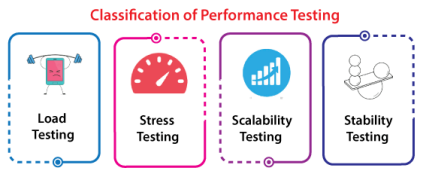
*3. Compatibility Testing*

***1. Performance Testing***

*In performance testing, the test engineer will test the working of an application by applying some load.*

*In this type of non-functional testing, the test engineer will only focus on several aspects, such as* ***Response time, Load, Scalability, and Stability of the software or an application.***

***Classification of Performance Testing***

*Performance testing includes the various types of testing, which are as follows: *

***Load Testing:*** *The goal is to measure the system's behavior under normal and heavy loads, ensuring it can handle* ***high traffic or demand without crashing or degrading the user experience.***

***Stress Testing:*** *Stress testing is to determine the breaking point of a system by pushing it beyond its designed capacity to understand* ***how it handles excessive load, resource exhaustion, or unexpected failures.***

***Scalability Testing:*** *Scalability testing is to understand the system's ability to* ***handle increased demands****—whether in terms of users, transactions, data volume, or other resource-intensive operations—****without performance degradation, failure, or bottlenecks.***

***Stability Testing:*** *To identify whether the system can operate reliably and* ***without failure during continuous use, long-running processes, or repeated executions,*** *ensuring that it* ***doesn't exhibit issues like memory leaks, crashes, or performance degradation over time.***

***Usability Testing***

*Another type of non-functional testing is usability testing. In usability testing, we will* ***analyze the user-friendliness of an application*** *and* ***detect the bugs in the software's end-user interface.***

***Here, the term user-friendliness defines the following aspects of an application:***

*● The application should be* ***easy to understand****, which means that all the features must be* ***visible to end-users.***

*● The application's* ***look and feel should be good*** *which means the application should be* ***pleasant looking*** *and make a feel to the end-user to use it.*

***Compatibility Testing***

*In compatibility testing, we will check the* ***functionality of an application*** *in specific hardware and software environments. Once the application is functionally stable then only, we go for compatibility testing.*

*Here, software means we can* ***test the application on different operating systems*** *and* ***other browsers, and hardware*** *means we can test the application on different sizes.*

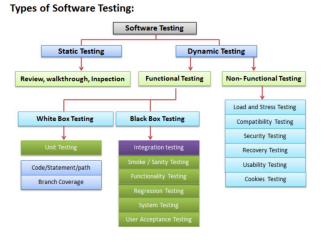
***Grey Box Testing***

*Another part of manual testing is Grey box testing. It is a collaboration of* ***black-box and white-box testing.***

*Since the Grey box testing includes access to the* ***internal coding*** *for designing test cases. Grey box testing is performed by a person who knows* ***coding as well as testing.***

******

*In other words, we can say that if a single-person team does both white-box and black-box testing, it is considered grey-box testing.*

**

*The purpose of having a* ***testing type*** *is to confirm the* ***AUT (Application Under Test).***

***AUT (Application Under Test)*** *refers to the* ***specific software application or system being tested during the software testing process.***

*The AUT can be* ***any type of software,*** *ranging from* ***Web Applications, Mobile Apps, Desktop Software, and Integrated Systems.***

*The software functional testing is mainly divided into two parts, which are as follows:*

➔ ***Manual Testing***

➔ ***Automation Testing***

***5.1.1 What is Manual Testing?***

*Testing any software or an application according to the client's needs* ***without using any automation tool is known as manual testing.***

*In other words, we can say that it is a procedure of* ***Verification and Validation.*** *Manual testing is used to verify the behavior of an application or software in contradiction of requirements specification.*

*In manual software testing, a tester carries out tests on the software by following a* ***set of predefined test cases****. In this testing, testers* ***make test cases for the codes, test the software, and give the final report about that software.***

*Manual testing is* ***time-consuming*** *because it is done by humans, and there is a* ***chance of human errors.***

***5.1.2 What is Automation Testing?***

*Automated Testing is a technique where the* ***Tester writes scripts*** *on their own and uses suitable* ***Software or Automation Tools*** *to test the software.*

*Automation testing involves using scripts or tools to conduct tests and can help* ***improve the efficiency, coverage, and reliability of testing.***

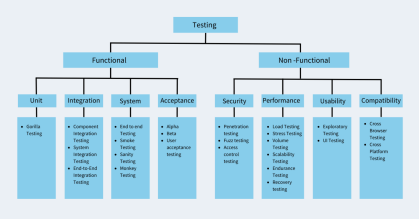
*It allows for* ***executing repetitive tasks*** *without the intervention of a Manual Tester.*

***5.1.3 Differentiation between Manual and Automation Testing:***

|  |  |  |
| --- | --- | --- |
| ***Parameters*** | ***Manual Testing*** | ***Automation Testing*** |
| ***Definition*** | *In manual testing, the test cases are executed by the* ***Human Tester****.* | *In automated testing, the test cases are executed by the* ***Software Tools****.* |
| ***Processing Time***  ***Resources***  ***Requirement***  ***Exploratory testing*** | *Manual testing is* ***time-consuming.***  *Manual testing takes up* ***human resources.***  *Exploratory testing is* ***possible*** *in manual testing.* | *Automation testing is* ***faster than manual testing.***  *Automation testing takes up* ***automation tools and trained employees.***  *Exploratory testing is* ***not possible*** *in automation testing.* |
| ***Framework***  ***requirement*** | *Manual testing* ***doesn’t use frameworks.*** | *Automation testing* ***uses frameworks*** *like Data Drive, Keyword, etc.* |

***Types of Testing***

*There are various software testing types for manual testing and automated testing, and different* ***testing methodologies such as black-box testing, white-box testing, and gray-box testing.*** *During the testing process, testers may use various* ***testing types such as functional testing, performance testing, security testing, and usability testing.***

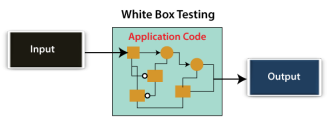
******

***What are the Software Testing Techniques and How are They Different from Testing Types?***

***White Box Testing***

➔ *White box testing techniques* ***analyze the internal structures*** *of the* ***used data structures, internal design, code structure, and the working of the software.***

➔ *It is also called* ***glass box, clear box testing, or structural testing****. White Box Testing is also known as* ***transparent testing or open box testing.*** ➔ *The developer will execute the complete white-box testing for the particular software and* ***send the specific application to the testing team.*** ➔ *The purpose of implementing white box testing is to emphasize the* ***flow of inputs and outputs over the software and enhance the security of an application.***

******

*White box testing is a software testing technique that involves testing the internal structure and workings of a software application.*

*The tester has access to the* ***source code*** *and uses this knowledge* ***to design test cases that can verify the correctness of the software at the code level.***

***5.1.1.2.2 Black Box Testing***

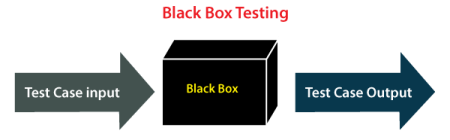
*In this testing, the test engineer will* ***analyze the software against requirements****,* ***identify the defects or bugs, and send it back to the development team.***

*Then, the developers will fix those defects,* ***do one round of White box testing****, and send it to the testing team.*

*Here,* ***fixing the bugs*** *means the* ***defect is resolved****, and the particular feature is working according to the given requirement.*

*The main objective of implementing black box testing is to specify the business needs or the customer's requirements.*

*In other words, we can say that black box testing is a* ***process of checking the functionality of an application*** *as per the customer's requirement. The* ***source code is not visible*** *in this testing; that's why it is known as black-box testing.*

**

***5.1.1.2.2.1 Types of Black Box Testing***

*Black box testing is further categorized into two parts, which are discussed below:*

*● Functional Testing*

*● Non-function Testing*

***Functional Testing***

*The test engineer will* ***check all the components systematically against requirement specifications known*** *as functional testing. Functional testing is also known as* ***Component testing****.*

*In functional testing,* ***all the components are tested by giving the value, defining the output, and validating the actual output with the expected value.***

*Functional testing is a part of* ***black-box testing*** *as its emphasizes application requirements rather than actual code. The test engineer has to test only the program instead of the system.*

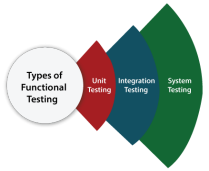
***Types of Functional Testing***

*Functional testing is also classified into various categories.*

*● Unit Testing*

*● Integration Testing*

*● System Testing*

**

***Unit Testing***

*Unit testing is the* ***first level*** *of functional testing to test any software. In this, the test engineer will* ***test the module of an application independently or test all the module functionality is called unit testing.***

***Integration Testing***

*Once we have successfully implemented the unit testing, we will go through integration testing. It is the* ***second level*** *of functional testing, where we test the* ***data flow between dependent modules or interfaces between two features is called integration testing.***

*The purpose of executing the integration testing is to test the* ***statement's accuracy between each module.***

***Types of Integration Testing***

*Integration testing is also further divided into the following parts:*

***1. Incremental Integration Testing***

*Whenever there is a* ***clear relationship*** *between modules, we go for* ***incremental integration testing.*** *Suppose, we take two modules and analyze the data flow between them if they are working fine or not.*

*If these modules are working fine, then we can add one more module and test again. And we can continue with the same process to get better results.*

*In other words, we can say that incrementally adding up the modules and testing the data flow between the modules is known as Incremental integration testing.*

***Types of Incremental Integration Testing***

*Incremental integration testing can further be classified into two parts, which are as follows:*

*Top-down Incremental Integration Testing*

*Bottom-up Incremental Integration Testing*

***Top-down Incremental Integration Testing***

*In this approach, we will add the modules step by step or incrementally and test the data flow between them. We have to ensure that the* ***modules we are adding are the child of the earlier ones.***

***Bottom-up Incremental Integration Testing***

*In the bottom-up approach, we will add the modules incrementally and check the data flow between modules. And also, ensure that the* ***module we are adding is the parent of the earlier ones.***

***2. Non-Incremental Integration Testing/ Big Bang Method***

*Whenever the* ***data flow is complex and very difficult*** *to classify a parent and a child, we will go for the* ***non-incremental integration approach****. The non-incremental method is also known as the* ***Big Bang method.***

***System Testing***

*Whenever we are done with the unit and integration testing, we can proceed with the system testing.*

*In system testing, the* ***test environment is parallel to the production environment****. It is also known as* ***end-to-end testing.***

*In this type of testing, we will undergo each attribute of the software and test if the* ***end feature works according to the business requirement****. An analysis of the software product as a complete system.*

***Non-functional Testing***

*It provides* ***detailed information on software product performance and used technologies.***

*Non-functional testing will help us* ***minimize the risk of production and related costs of the software.*** *Non-functional testing combines* ***performance, load, stress, usability, and compatibility testing.***

***Types of Non-functional Testing***

*1. Performance Testing*

*2. Usability Testing*

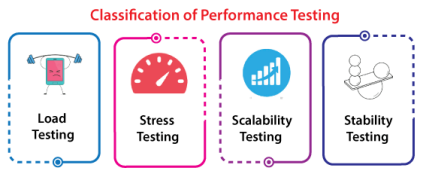
*3. Compatibility Testing*

***1. Performance Testing***

*In performance testing, the test engineer will test the working of an application by applying some load.*

*In this type of non-functional testing, the test engineer will only focus on several aspects, such as* ***Response time, Load, Scalability, and Stability of the software or an application.***

***Classification of Performance Testing***

*Performance testing includes the various types of testing, which are as follows: *

***Load Testing:*** *The goal is to measure the system's behavior under normal and heavy loads, ensuring it can handle* ***high traffic or demand without crashing or degrading the user experience.***

***Stress Testing:*** *Stress testing is to determine the breaking point of a system by pushing it beyond its designed capacity to understand* ***how it handles excessive load, resource exhaustion, or unexpected failures.***

***Scalability Testing:*** *Scalability testing is to understand the system's ability to* ***handle increased demands****—whether in terms of users, transactions, data volume, or other resource-intensive operations—****without performance degradation, failure, or bottlenecks.***

***Stability Testing:*** *To identify whether the system can operate reliably and* ***without failure during continuous use, long-running processes, or repeated executions,*** *ensuring that it* ***doesn't exhibit issues like memory leaks, crashes, or performance degradation over time.***

***Usability Testing***

*Another type of non-functional testing is usability testing. In usability testing, we will* ***analyze the user-friendliness of an application*** *and* ***detect the bugs in the software's end-user interface.***

***Here, the term user-friendliness defines the following aspects of an application:***

*● The application should be* ***easy to understand****, which means that all the features must be* ***visible to end-users.***

*● The application's* ***look and feel should be good*** *which means the application should be* ***pleasant looking*** *and make a feel to the end-user to use it.*

***Compatibility Testing***

*In compatibility testing, we will check the* ***functionality of an application*** *in specific hardware and software environments. Once the application is functionally stable then only, we go for compatibility testing.*

*Here, software means we can* ***test the application on different operating systems*** *and* ***other browsers, and hardware*** *means we can test the application on different sizes.*

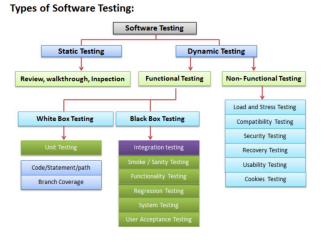
***Grey Box Testing***

*Another part of manual testing is Grey box testing. It is a collaboration of* ***black-box and white-box testing.***

*Since the Grey box testing includes access to the* ***internal coding*** *for designing test cases. Grey box testing is performed by a person who knows* ***coding as well as testing.***

******

*In other words, we can say that if a single-person team does both white-box and black-box testing, it is considered grey-box testing.*

**

*The purpose of having a* ***testing type*** *is to confirm the* ***AUT (Application Under Test).***

***AUT (Application Under Test)*** *refers to the* ***specific software application or system being tested during the software testing process.***

*The AUT can be* ***any type of software,*** *ranging from* ***Web Applications, Mobile Apps, Desktop Software, and Integrated Systems.***

*The software functional testing is mainly divided into two parts, which are as follows:*

➔ ***Manual Testing***

➔ ***Automation Testing***

***5.1.1 What is Manual Testing?***

*Testing any software or an application according to the client's needs* ***without using any automation tool is known as manual testing.***

*In other words, we can say that it is a procedure of* ***Verification and Validation.*** *Manual testing is used to verify the behavior of an application or software in contradiction of requirements specification.*

*In manual software testing, a tester carries out tests on the software by following a* ***set of predefined test cases****. In this testing, testers* ***make test cases for the codes, test the software, and give the final report about that software.***

*Manual testing is* ***time-consuming*** *because it is done by humans, and there is a* ***chance of human errors.***

***5.1.2 What is Automation Testing?***

*Automated Testing is a technique where the* ***Tester writes scripts*** *on their own and uses suitable* ***Software or Automation Tools*** *to test the software.*

*Automation testing involves using scripts or tools to conduct tests and can help* ***improve the efficiency, coverage, and reliability of testing.***

*It allows for* ***executing repetitive tasks*** *without the intervention of a Manual Tester.*

***5.1.3 Differentiation between Manual and Automation Testing:***

|  |  |  |
| --- | --- | --- |
| ***Parameters*** | ***Manual Testing*** | ***Automation Testing*** |
| ***Definition*** | *In manual testing, the test cases are executed by the* ***Human Tester****.* | *In automated testing, the test cases are executed by the* ***Software Tools****.* |
| ***Processing Time***  ***Resources***  ***Requirement***  ***Exploratory testing*** | *Manual testing is* ***time-consuming.***  *Manual testing takes up* ***human resources.***  *Exploratory testing is* ***possible*** *in manual testing.* | *Automation testing is* ***faster than manual testing.***  *Automation testing takes up* ***automation tools and trained employees.***  *Exploratory testing is* ***not possible*** *in automation testing.* |
| ***Framework***  ***requirement*** | *Manual testing* ***doesn’t use frameworks.*** | *Automation testing* ***uses frameworks*** *like Data Drive, Keyword, etc.* |

**---------------------------------------------------------------------------------------------------------------------------**

**Testing process→1. smoke testing**

Developer will be sending a build, means a url which supports test env

Test env -url will be different

URLS - dev url— ity moves to test

Test url

Stage url

Prodn url

User site url( which is final)

Smoke testing→ tester will do one level complete high level whether the application is able to launch the url and then key functionalities are working

**System testing:** complete testing ( write test scenarios(before that we would have created test strategy, test plan , test cases,defect reporting, test closure report)

Completing 1 round of testing

Automation testing

**E2E testing:every sprint e2e sprint level- quality of the product is good.**

**Regression Testing**:(Rechecking all the prev sprint cases are working fine so that any new changes defect fixes doesn't impact)

For ex: Currently ur working sprint 10 ( regression testing for Sprint 1to 9) all the cases(e2e test cases u can pick up)-**-Automation Testing** – By clicking once all my prev spring 1 to 9 should work

Sanity testing–testing after everything is completed

Focus – defect free or all the requirements are matching to the testing we have done or application developed

**RTM- Requirement Traceability Matrix.**

We creating test cases mapping to requirements

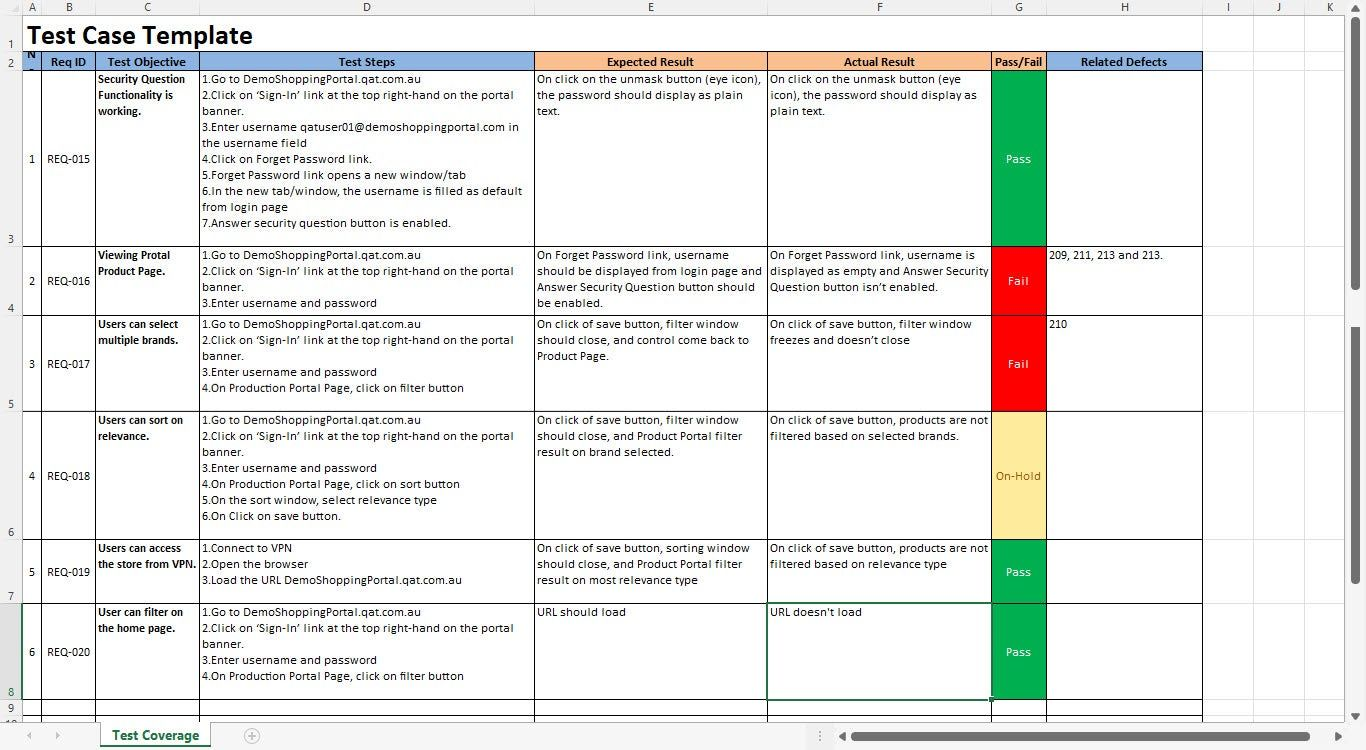
Defect– map to requirements

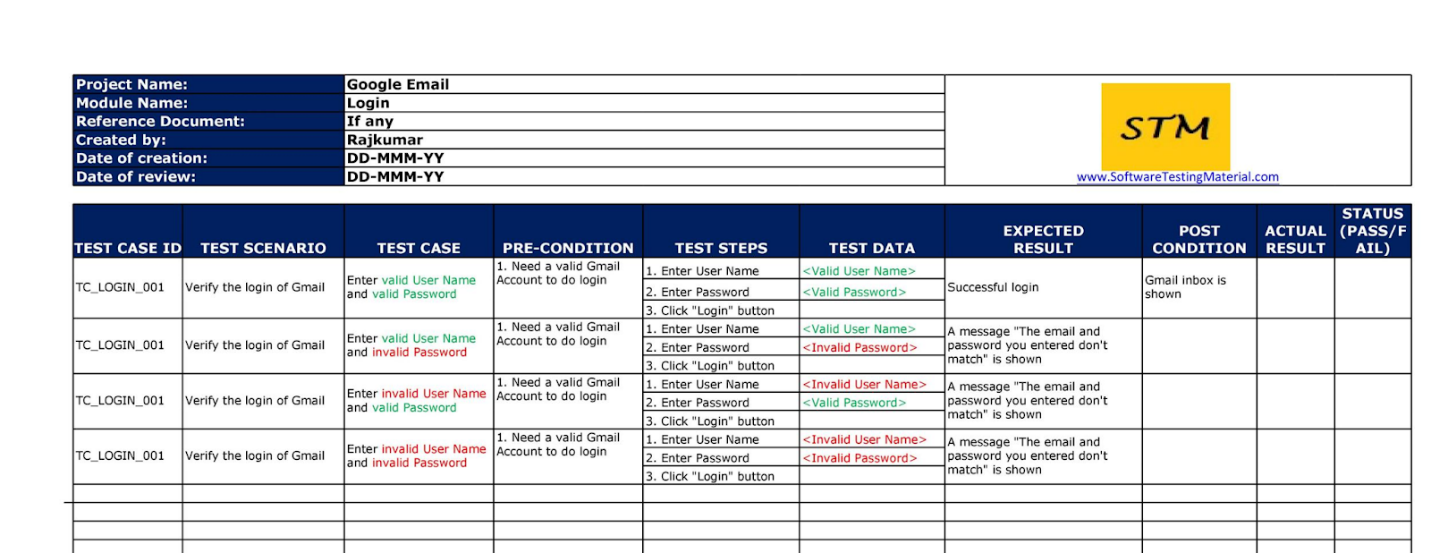
One **requirement** no can have **multiple test cases, scenarios, defects**

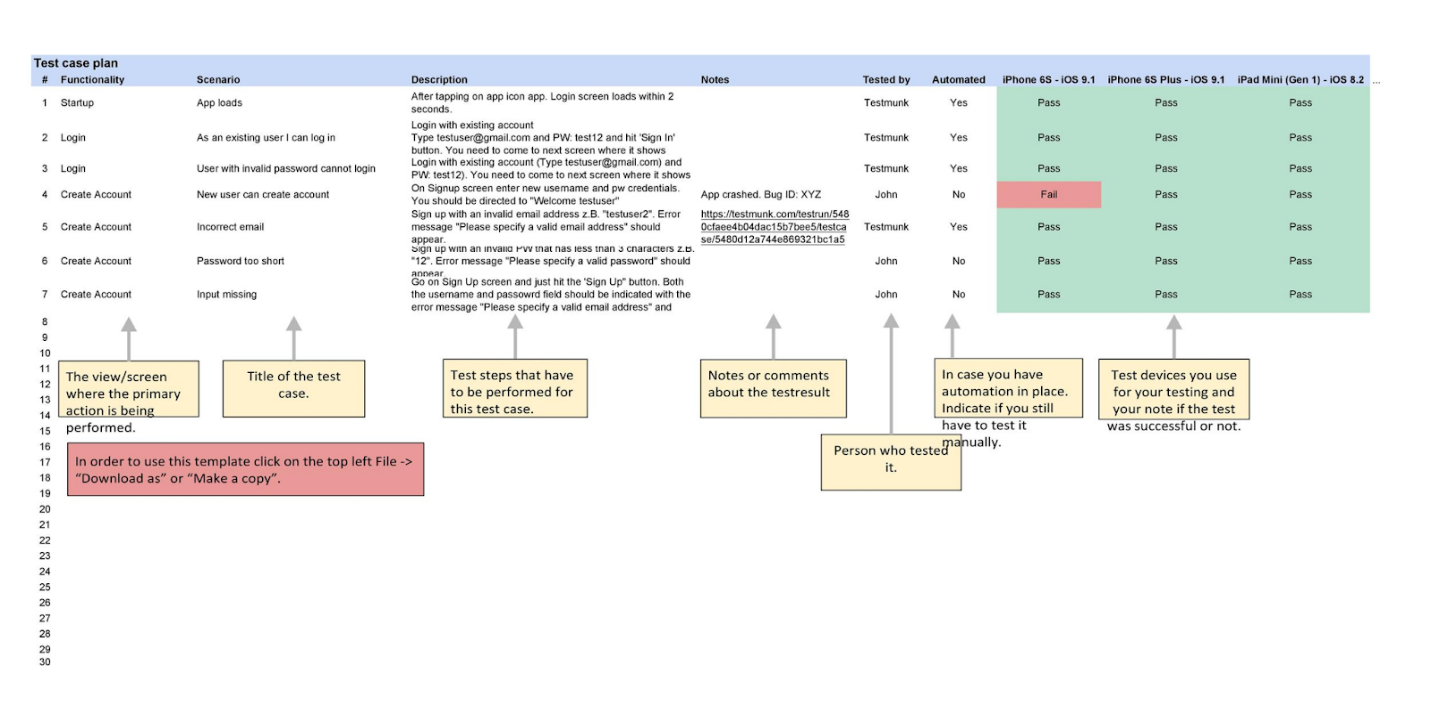
**7 teams - 7 different test cases**

**In the test case add –Test Data - give a mock data of 34 characters**

**Naming conventions : Req01.Test scenario - TS01.1, Test case ID TC01.1\_TS01.1**







**Boundary value analysis-**

**Equivalence partitioning– range of elements**

**UAT  : search about this**

**System testing**

**Integration testing**

----------------------------------------------------------------------------------------------------------------

**Test Closure Report/Test Summary Report**

1. How is the Testing done,
2. No of defects , Defect ID - mapping to the requirements
3. **If any defects identified is deferred to next sprint**
4. If all defects are closed by –developers, or any pending
5. Test Closure report- sign off that all in scope requirements are approved and signed off by the test manager

**Defect Bug Raising**

Defect- **any functionality which is not matching the requirements, show stoppers,cosmetic bugs**

**Example:**

**Launch the URL –: not able to launch ( Showstopper)- blocked– identified in smoke testing– high level testing**

**Able to launch → unable to proceed with certain important functionalities – Critical**

**Font, size, color issues- cosmetic errors**

**Test steps – expected and actual**

**Defect Life Cycle:Assignee(D1)**

**NEW(T)→ OPEN(T) →FIXED(D)-->RETEST(D)—>CLOSE(T)**

**RETEST(D)→REOPEN(T)-->RETEST(D)--CLOSE(T)**

**New(T)--> Open(T)--> Duplicate(d)(no 2 defects should be same)**

**New→Open→Deferred**

**New→Open→Rejected**

**Severity and priority**

**High severity(T) High priority(D)**

**High Tester Low Developer**

**Low Tester High developer**

**Low for both**

**Complex, major, minor- names of defects**

**Test Closure Report/Test Summary Report**

1. How is the Testing done,
2. No of defects , Defect ID - mapping to the requirements
3. **If any defects identified is deferred to next sprint**
4. If all defects are closed by –developers, or any pending
5. Test Closure report- sign off that all in scope requirements are approved and signed off by the test manager

**Defect Bug Raising**

Defect- **any functionality which is not matching the requirements, show stoppers,cosmetic bugs**

**Example:**

**Launch the URL –: not able to launch ( Showstopper)- blocked– identified in smoke testing– high level testing**

**Able to launch → unable to proceed with certain important functionalities – Critical**

**Font, size, color issues- cosmetic errors**

**Test steps – expected and actual**

**Defect Life Cycle:Assignee(D1)**

**NEW(T)→ OPEN(T) →FIXED(D)-->RETEST(D)—>CLOSE(T)**

**RETEST(D)→REOPEN(T)-->RETEST(D)--CLOSE(T)**

**New(T)--> Open(T)--> Duplicate(d)(no 2 defects should be same)**

**New→Open→Deferred**

**New→Open→Rejected**

**Severity and priority**

**High severity(T) High priority(D)**

**High Tester Low Developer**

**Low Tester High developer**

**Low for both**

**Complex, major, minor- names of defects**

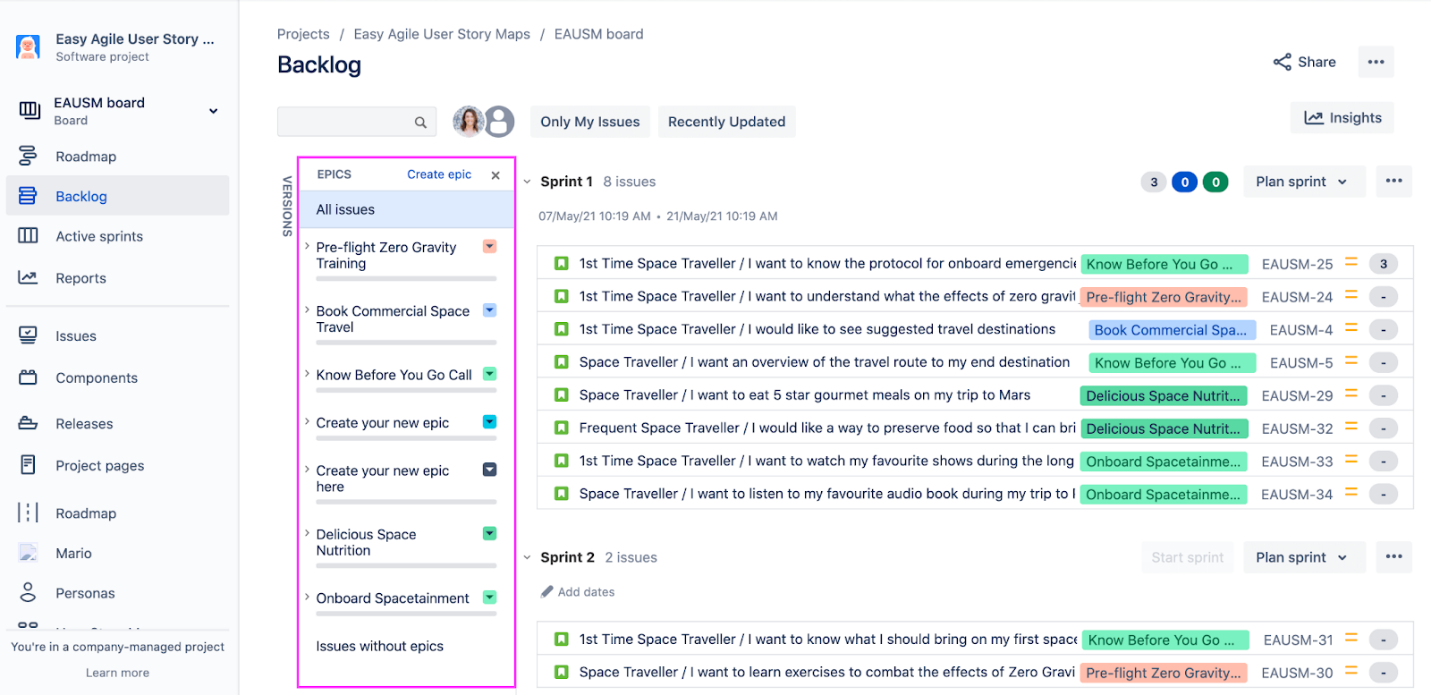
**--------------------------------------------------------------------------------------------------------------------------**

JIRA-

**Project Management Tool**

**Create a Project, branches →Create Plan**

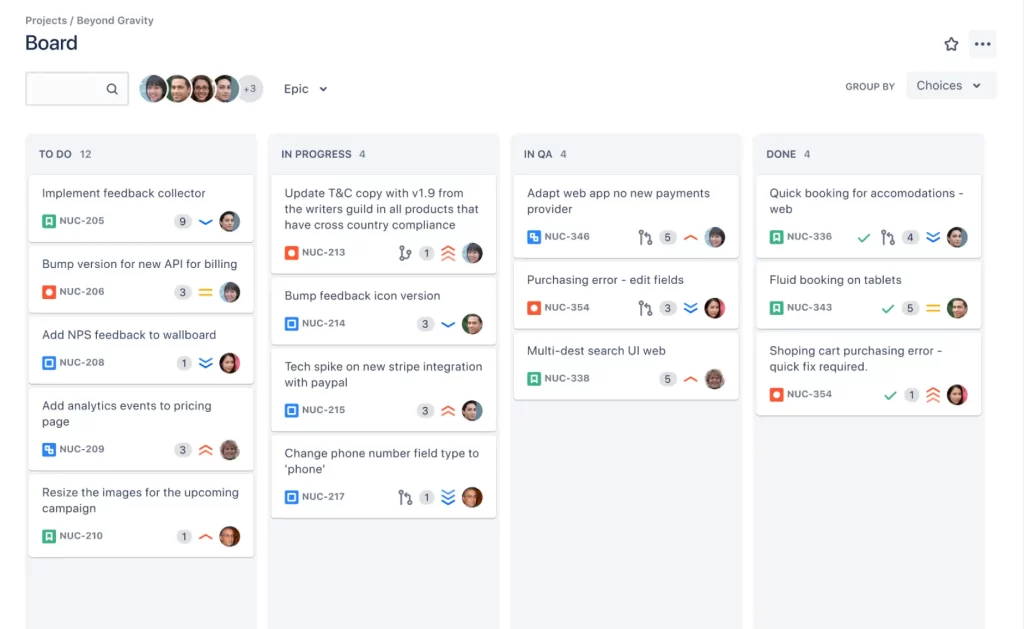
**Product Owners(Stake holders will create user stories**

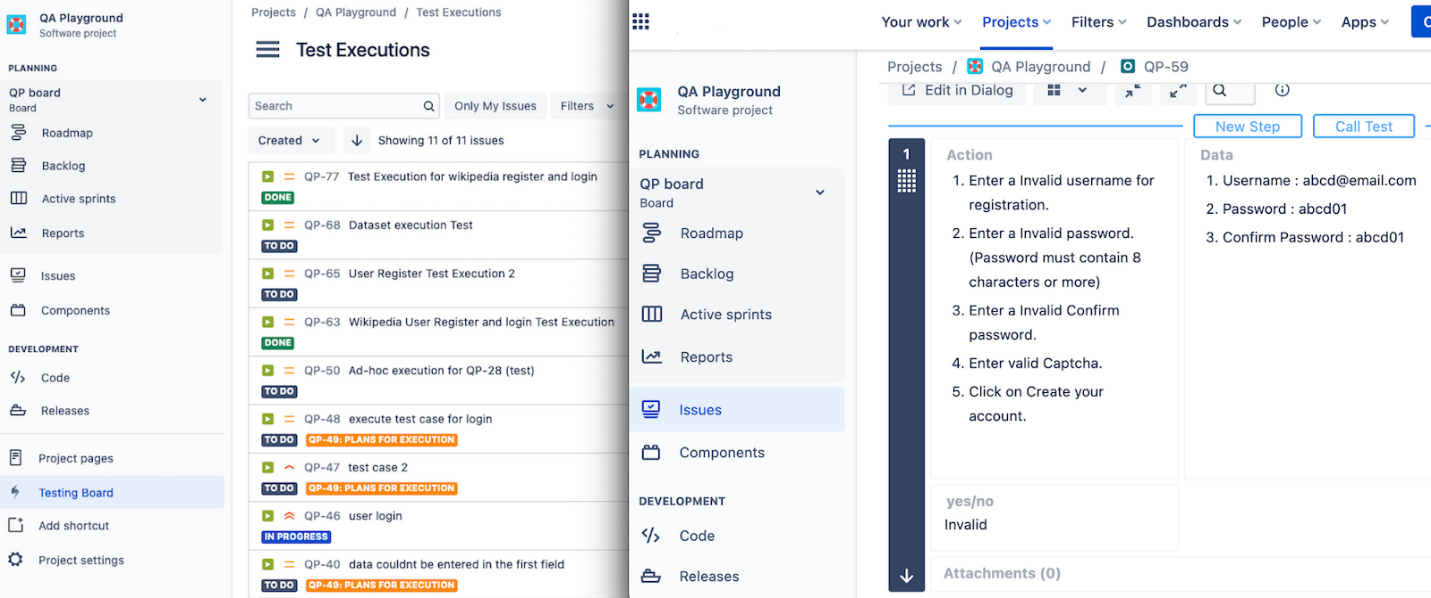
****

**For each user story→tasks will be created**

**Feature→Epic→ Tasks→Test cases/issues**

**Or Bugs**

****

****