

1. What is Quality?

Quality can be seen from two perspectives: customer satisfaction and meeting requirements. From the customer's point of view, quality means they are happy with the product. From the product point of view, quality means the product meets the specified requirements.

However, there are important points to remember:

Customer satisfaction is subjective — different customers may have different expectations and opinions about quality.

Also, even if a product meets all requirements, it doesn't always mean customers will like it or actively use it

2. What is Software Quality?

Software quality is when the product does what it should do and users don't have problems using it.

For example, in an online store, users can search products, add items to the cart, and complete checkout without bugs or crashes.

3. What is Software Quality Assurance (SQA)?

Software Quality Assurance is about making sure the software development process is done correctly so the final product is high quality.

It covers all stages of development — from gathering requirements to coding, testing, and release.

What does SQA involve?

It focuses on monitoring and improving the development process, making sure teams follow agreed standards and procedures, and detecting and fixing problems as early as possible.

4. What is Software Testing?

Software testing is the process of checking that the software works correctly, meets requirements, and is of good quality.

It helps us find bugs, missing functionality, and issues before the product reaches users.

Software testing includes a set of activities aimed at finding errors in the software before it is released to users.

These activities include analyzing the software, identifying differences between actual and expected behavior, and checking whether the software works the way it should

Purpose of Software Testing

The main purpose of software testing is to verify and validate the product and to find defects.

Verification – Are we building the system right?

We check the actual results against the requirements to make sure the system is implemented correctly.

Validation – Are we building the right system?

We check whether the system meets real user needs and does what users actually want.

As a result:

Testing helps detect errors early and fix problems —
finding cases where something happens when it shouldn't, or doesn't happen when it should.

5. SQA vs Software Testing

Software Quality Assurance (SQA) is preventive.

It focuses on improving and monitoring the entire software development process and covers all phases of the SDLC to prevent defects from happening.

Software Testing is detective.

It focuses on testing the actual software to find defects by comparing expected results with actual results.

SQA is preventive — improves the process to avoid defects.

Testing is detective — finds defects in the product.

6. Software Testing Life Cycle (STLC) Phases

The Software Testing Life Cycle is a set of steps followed to test software effectively. The main phases are:

1. **Requirements Analysis** – Understand what the software is supposed to do and identify testable requirements.
2. **Test Planning** – Decide what to test, how to test, resources, timelines, and tools.
3. **Test Development** – Create test cases, test data, and prepare test scripts.
4. **Test Execution** – Run the tests and compare actual results with expected results.
5. **Defect Management** – Report, track, and manage defects until they are fixed.
6. **Test Reporting** – Document testing results, coverage, and any remaining risks.

7. What is SDLC?

Software Development Life Cycle (SDLC) is the process of planning, creating, testing, and maintaining software.

It describes all the stages of software development, from understanding requirements to designing, building, deploying, and maintaining the application.

Main activities in SDLC:

1. **Requirements & Specification** – Understand what the software should do.
2. **Architecture & Design** – Plan how the software will work.
3. **Implementation** – Write the actual code.
4. **Testing** – Check that the software works correctly.
5. **Deployment** – Release the software to users.
6. **Maintenance** – Fix bugs and update the software over time.

Development Models in SDLC:

Waterfall, Agile (Scrum, Kanban, XP, FDD, DSDM), V-Model, Spiral, RAD, RUP, Cleanroom, Iterative — different approaches to organize these steps.

Development models in software development provide a structured way to plan, organize, and track a project. They help ensure the project is delivered on time and meets requirements.

Main benefits:

1. Project planning, estimating, and scheduling – Helps decide timelines and resources.
2. Standard terminologies, activities, and deliverables – Everyone understands the process and what needs to be done.
3. Project tracking and control – Makes it easier to monitor progress and manage issues.
4. Visibility for stakeholders – Everyone involved knows how the project is progressing.

Some popular SDLC development models are:

- Waterfall – traditional, sequential approach, where each phase is completed before moving to the next.
- Agile – flexible and iterative approach, including frameworks like Scrum, XP (Extreme Programming), and Kanban.
- ...and many more depending on the project needs.

7. Waterfall Model

The Waterfall model is a traditional, sequential software development process.

Work flows step by step from one phase to the next, like a waterfall. The main phases are:

- Conception / Initiation – understand the project idea and feasibility
- Analysis – gather and analyze requirements
- Design / Validation – plan system architecture and design
- Construction / Implementation – write the code
- Testing – check the software for defects
- Maintenance – fix issues and update software after release

In Waterfall, once a phase is complete, it's usually not revisited, so changes are difficult later.

Waterfall is easy to understand and well-structured, with clear phases and milestones. But it's inflexible — all requirements must be set upfront, and it's hard to go back and change things. It's not ideal for complex projects or when requirements can change.

Agile Model

Agile is a software development model that focuses on incremental development and teamwork. Teams work in small cross-functional groups, delivering the product in short cycles called **sprints**, getting feedback early, and continuously improving the product.

Agile is all about small, iterative releases, teamwork, and adapting to changes quickly.

Agile values people and collaboration, working software, customer feedback, and being flexible to change.

8. Scrum Team

A Scrum team has a **Scrum Master** to guide the process, a **Product Owner** to prioritize work, and **developers** who do the work and organize themselves.

Agile: Scrum vs Kanban

Scrum:

- Teams work in short cycles called sprints to release working software
- Scrum has defined roles (Scrum Master, Product Owner, Developers), special artifacts (like product backlog, sprint backlog), and **regular ceremonies** (daily stand-up, sprint review, sprint retrospective).

Kanban:

- A simpler, more flexible approach than Scrum.

- No fixed roles, no sprints.
- Focuses on visualizing work and continuously improving flow using a Kanban board.

Scrum works in sprints with defined roles and ceremonies, while Kanban is more flexible, without sprints or set roles, and focuses on visualizing work.

Scrum Terminology

1. **Sprint** – a short, fixed-length iteration, usually 2–4 weeks, during which the team delivers working software.
2. **Product Backlog** – a list of all features or requirements for the product.
3. **Sprint Backlog** – a list of the highest-priority items from the product backlog that the team commits to complete in the current sprint.

A sprint is a short 2–4 week cycle to deliver software. The product backlog lists all features, and the sprint backlog shows the top priority items for the sprint.

Scrum Process

1. User Stories – Features are written as user stories, describing what the user wants.
2. Estimating Work – The Scrum team estimates how much effort each story will take.
3. Sprint Backlog – The team selects stories for the sprint and works on the sprint backlog.
4. Daily Scrum Meeting – The team meets daily to discuss progress, obstacles, and next steps.

In Scrum, features are written as user stories, the team estimates and picks stories for the sprint backlog, and they meet daily to track progress.

Project Plan

A Project Plan is a formal document that summarizes all important aspects of a project — business goals, management, and financial details.

It acts like a “contract” between the Project Manager and the customer.

It usually includes:

- Project scope – what will be done
- Objectives – project goals
- Benefits – value for the business or customer
- Costs – budget
- Risks – possible problems

Plans – timelines, resources, and schedules

A project plan is a formal document that shows what the project will do, its goals, costs, risks, and timeline. It's like a contract between the manager and customer.

Project Charter / Project Plan

The Project Plan is a document that helps you understand a new software project. It answers questions like:

- What is the project name?
- What type of application is being developed?
- What are dependencies, risks, and assumptions?
- What activities are planned?
- What resources are available?
- How big is the application?
- How much development and QA effort is needed?

The project plan tells you what the project is about, its goals, scope, resources, and risks. QA uses this plan to create a QA strategy, define what is in or out of scope, align milestones, and assign resources.

Product Requirements

1. MRD (Marketing Requirements Document)

- Describes the market needs and why the product is needed.
- Includes target audience, business goals, and high-level features.
- QA uses it to understand product goals and priorities.

2. PRD (Product Requirements Document)

- Details what the product should do.
- Includes functional requirements, features, use cases, and acceptance criteria.
- QA uses it to create test cases and verify the product works as expected.

3. Use Cases

- Describe how users will interact with the product.
- QA uses them to simulate real user scenarios and ensure software behaves correctly.

If documents are not available:

- QA should communicate with the product owner or business analyst to gather requirements.
- Use meetings, user stories, or interviews to understand what needs to be tested.

MRD tells why the product is needed, PRD tells what it should do, and use cases show how users will interact. QA uses these to write test cases. If documents aren't available, QA talks to stakeholders to gather requirements.

Use Cases

- Use cases are a way to describe system requirements from the user's perspective.
 - Each use case represents a complete business operation performed by a user.
 - QA perspective: Execute end-to-end tests to ensure the requirement is implemented correctly.
-

Why do we need Product Requirements?

- They help QA understand the features and scope of the application.
- QA can decide what tests to conduct based on requirements.
- Gap analysis: Identify missing or unclear requirements.
- QA impacts product development by:
 - Clarifying requirements
 - Reducing the percentage of code changes due to unclear requirements

No Requirements? How to test?

- Use common sense
 - Compare with similar applications
 - Try exploratory testing to discover issues
-

Why do we test?

To do Quality Assurance:

- Requirements Analysis
- Test Planning

- Bug Reporting & Tracking
 - Test Automation
 - Release Certification
 - Maintain and improve the system
-

SDLC Environments for Testing

Tests may occur in different environments:

- Development – where developers write code
 - QA / Testing environment – dedicated for testing
 - Staging – pre-production, simulates production
 - Production – live environment
 - etc.
-