Lecture#01

Quality	Assurance	Control
Meeting customer expectations or needs	Confidence given by organization	Test or verify actual product

1. **Software Quality Attributes / Factors:**

5.1 Functionality: Checks if the software does what it's supposed to do.

Example: A calculator app must correctly add, subtract, etc.

5.2 Performance: Checks how fast and smooth the software works.

Example: A video game should not freeze or lag.

5.3 Reliability: Checks if the software gives correct results under different conditions.

Example: An online form should work on Chrome, Firefox, and mobile.

5.4 Testability: Checks how easy it is to test the software.

Example: If the app has clear inputs and outputs, it's easier to test.

5.5 Availability: Checks if the software is ready when needed.

Example: An online banking app should work 24/7.

5.6 Interoperability: Checks if the software can work with other systems.

Example: A messaging app integrating with a smartwatch to send replies.

5.7 Security: Checks how well software protects from hackers & unwanted access.

Example: A password-protected app with two-step login.

5.8 Flexibility: Checks how easily the software can adapt to changes.

Example: An app working on Windows 8 should also update to work on Windows 11.

5.9 Efficiency: Checks if the software uses system resources wisely.

Example: A music app shouldn't use up all your phone battery or memory.

5.10 Usability: Checks how easy it is for users to use the software.

Example: A to-do list app should be simple with clear buttons.

Lecture#02 (SKIP)

Lecture#03

1. **Software:** According to the IEEE definition, software has four parts:
   - **Computer Programs:** The actual code that runs the software.
   - **Procedures:** The steps or rules to use the software.
   - **Documentation:** The guides or manuals that explain how to use it.
   - **Data:** The information the software needs to work, like user details or settings.

Errors	Fault	Failures
A mistake in the code or logic by a person.	When that error causes the software to work incorrectly.	When faulty part is actually used, and something goes wrong.
Example: A programmer sets wrong discount formula.	**Example:** That causes wrong total bill sometimes.	**Example:** A customer sees the wrong amount and complains.

2. **Software Quality (IEEE):** The degree to which a system, component, or process meets customer or user needs or expectations.
3. **SQA (IEEE):** A planned and systematic pattern of all actions necessary to provide adequate confidence that an item or product conforms to established technical requirements.

Lecture#04

Software Quality Assurance	Software Quality Control
It is a process to make sure the software will be good.	It is a process to check if the software meets the quality needs.
It tries to stop mistakes before they happen.	It finds and fixes mistakes after they happen.
It involves in full software development life cycle	It involves in full software testing life cycle
It happens before QC.	It happens after QA is done.
It is a Low-Level Activity, it can identify an error and mistakes which QC cannot.	It is a High-Level Activity, it can identify an error that QA cannot.
It requires the involvement of the whole team	It requires the involvement of the Testing team
Identifies flaws in the process.	Identifies flaws in the product.
Proactive (prevents defects)	Reactive (finds defects in final product).
Example: Using checklist during development.	**Example**: Testing app after it's built to find bugs.

1. Objectives of SQA Activities:

1.1 Software Development (Process-Focused)

- Assuring an acceptable level of confidence that the software will conform to functional technical requirements.
- Assuring an acceptable level of confidence that the software will conform to managerial scheduling and budgetary requirements.
- Initiation and management of activities for the improvement and greater efficiency of software development and SQA activities.

1.2 Software Maintenance (Product-Focused)

- Assuring an acceptable level of confidence that the **software maintenance activities** will conform to the functional technical requirements.

- Assuring an acceptable level of confidence that the **software maintenance activities** will conform to managerial scheduling and budgetary requirements.
- Initiate and manage activities to improve and increase the efficiency of **software maintenance** and SQA activities.

Lecture#05

1. Impediments to Software Quality Assurance and Software Quality Control:

1.1 Planning Issues:

- There's no separate test plan.
- The app is complicated but has no clear requirements.
- Requirements keep changing, or development gets delayed.

1.2 Incorrect Test Management:

- There's not enough money for testing. There aren't enough tools or tech.

1.3 Lack of Experience:

- New developers make more mistakes, and new QA engineers take longer to find issues.

1.4 Insufficient Integration of Testing and Development

- Testing and development teams don't work well together.

1.5 Problems with Test Automation

- Teams lack knowledge or pick the wrong tools.
- Apps change too often, slowing down automation.

1.6 Low-Quality Product Requirements

- Requirements are missing, unclear, or contradictory which leads to more mistakes.
- Better requirements help reduce errors.

2. Quality Assurance in Requirement Analysis Phase: In this phase, QA ensures the software requirements are clear and correct before development starts. Here's what happens:

- **Check All Requirements:** Review the specification document and use cases carefully.
- **List Main Scenarios:** Write down the big-picture scenarios of how the software should work.
- **Ask Questions:** Clear doubts by talking to stakeholders (like clients or managers).
- **Give Suggestions:** Share ideas to improve features or point out logical problems.
- **Raise Issues:** Note any defects or unclear parts in the requirements document.
- **Track Defects:** Keep a record of issues found in the requirements.

- **Make Test Scenarios:** Create high-level test ideas to check the software later.
- **Create Traceability Matrix:** Link requirements to test cases to ensure everything is covered.

3. Quality Assurance in Requirement Design Phase:

- If the design isn't good, even clear requirements won't help, and building the software correctly becomes impossible.
- Use a checklist during the design process to improve quality, as shown by industry practices.

Lecture#06 (SKIP)

Lecture#07

1. Three General Principles of QA:

➢ **Know What You Are Doing:** Understand what is being built, how it is being built and what it currently does.

How to apply this:

- Use project management tools (like milestones, schedules).
- Follow reporting and tracking procedures to monitor progress.

➢ **Know What You Should Be Doing:** Clearly understand the software's goals — what the customer expects it to do.

How to apply this:

- Have clear requirements documents.
- Create acceptance tests to check if the product meets those requirements.
- Get frequent feedback from users to stay aligned with their needs.

➢ **Know How to Measure the Difference:** Measure how much the current product matches the plan (what you are doing vs what you should be doing).

Four ways to measure:

1. **Formal methods –** Use mathematical proofs to verify parts of the system.
2. **Testing –** Give inputs and check if outputs are correct.
3. **Inspections –** Manually review documents, code, and designs using checklists.
4. **Metrics –** Track numbers (e.g., number of bugs, test coverage, defect density).

2. Software Quality Challenges:

➢ Complex Software requires different monitoring procedures than trivial applications. Complex systems (like online banking apps or hospital systems) can't be checked using the same methods as simple ones (like a calculator).

- Quality Standards Change Throughout the Project. In the beginning: focus on good requirements, During development: focus on clean code, Before release: focus on testing and fixing bugs.
- The measures of the quality must be specific to the project being evaluated and must assess the effectiveness of the entire development process, not just individual segments.
- Quality cannot be directly checked in the product; it must planned right from the beginning.
- Quality goals must be clearly defined and effectively monitored.

3. Changing View of Quality:

Past	Present
Quality is the job of blue-collar workers and direct labor on the product.	Quality is everyone's job, including white-collar workers, indirect labor, and overhead staff.
Quality defects should be hidden from customers and management.	Defects should be highlighted and fixed openly.
Quality problems lead to blame, excuses, and justification.	Quality problems lead to teamwork and solutions.
Fixes for quality issues need little documentation.	Documentation is key for "lessons learned" to avoid repeating mistakes.
More quality increases project costs.	Better quality saves money and boosts business.
Quality is internally focused.	Quality is customer-focused.
Quality happens during project execution.	Quality starts at project planning and must be planned.

Lecture#08 (SKIP)

Additional Important Topics:

1. Why Quality Is Important:

- Quality is critical for survival and success.
- Customers demand quality.
- Everybody wants quality
- Everybody has a different perception of quality.
- Essentially quality means satisfying customer.

2. Why Businesses Should Concerned About Quality:

- Quality is a competitive issue now
- Quality is a must for survival
- Quality gives you global reach
- Quality is cost effective
- Quality helps retain customers and increase profits