# Chapter 1
# Introduction To Software Testing

## 1. Define SQA

- SQA stands for Software Quality Assurance.
- SQA is a set of activities that ensure software products meet expected quality standards.
- The primary objective of SQA is to identify and prevent defects and errors in software development processes.
- SQA involves implementing various techniques such as reviews, testing, and audits to identify and remove defects early in the development cycle.
- It also involves establishing quality standards, processes, and procedures that are followed throughout the software development life cycle.
- SQA is an important part of software development, as it helps to improve the overall quality of the software product and reduce the risk of software failures or defects.
- SQA increases customer satisfaction by delivering software products that meet their expectations and requirements.

## 2. Why is Quality Assurance necessary in different types of organizations? Justify with some examples.

Quality assurance is necessary in different types of organizations for several reasons. Here are some of the key reasons:

1. **Customer Satisfaction:**
   - Quality assurance helps to ensure that products or services meet customer expectations and requirements.
   - This leads to higher customer satisfaction and loyalty, which can be beneficial for any organization.
     For example, a software company developing a mobile app must ensure that it meets the specific requirements of its target users.

2. **Reduce the risk of defects:**
   - Quality assurance processes help to identify defects early in the software development cycle, which can reduce the cost of fixing errors and prevent potential safety hazards.
   - For example, medical devices must be developed and tested in accordance with industry safety standards to avoid putting patients at risk.

3. **Cost Savings:**
   - Quality assurance helps organizations to identify and prevent defects or errors early in the development cycle, which can reduce the cost of fixing the software.

4. **Brand Reputation:**
   - Quality assurance helps organizations to establish a strong brand reputation for delivering high-quality products or services.
   - This can be particularly important in competitive industries where customers have a lot of choices.
   - For example, a software company that consistently delivers high-quality products is more likely to attract new customers and retain existing ones.

5. **Continuous Improvement:**
   - Quality assurance processes involve continuous monitoring, evaluation, and improvement of products or services.
   - This can help organizations to stay competitive and adapt to changing market conditions.
   - For example, a tech company that continuously improves its products based on customer feedback is more likely to stay ahead of the competition.

## 3. Career as software tester

The career as a software tester can be both rewarding and challenging.

It is important to consider both the advantages and disadvantages when deciding whether a career as a software tester is right for you.

Here are some of the advantages and disadvantages of pursuing a career as a software tester:

### Advantages:

1. **High demand**: Software testers are in high demand in the tech industry, and the demand is expected to continue to grow in the coming years.
2. **Competitive salary**: The salary for software testers can be competitive, with opportunities for growth and advancement.
3. **Job security:** With the increasing importance of software testing, the job security for software testers is typically high.
4. **Continuous learning:** As a software tester, you will have the opportunity to continuously learn new skills and technologies, which can be both challenging and rewarding.
5. **Opportunity to work on exciting projects:** As a software tester, you may have the opportunity to work on exciting and innovative projects.

### Disadvantages:

1. **Can be repetitive**: Some aspects of software testing, such as executing test cases, can be repetitive and tedious.
2. **Can be stressful**: The pressure to find and report bugs can be stressful, especially when deadlines are tight.
3. **Requires attention to detail:** Software testing requires a high level of attention to detail, which may not be suited for everyone.
4. **Requires good communication skills**: Effective communication with developers and stakeholders is critical for success as a software tester.
5. **May require working off-hours:** Depending on the organization, software testers may be required to work off-hours or weekends to meet deadlines.

## 4. Scope of software testing in Nepal.

- The software industry in Nepal is growing, creating more opportunities for software testers.
- Demand for software testers is increasing in various industries, including finance, healthcare, e-commerce, education, and more.
- Software testing professionals in Nepal can work for international clients and companies remotely.
- The government of Nepal is focusing on the development of the information technology sector, which is expected to drive the growth of the software industry and create more job opportunities for software testers.
- Skilled software testing professionals are in high demand in Nepal, making it a promising career option.
- Overall, the scope of software testing in Nepal is promising, and the demand for skilled software testers is likely to increase in the coming years.
- However, it is important for individuals to gain the necessary skills and experience to stay competitive in the job market and to help advance the software testing industry in Nepal.

# Chapter 2
# Fundamentals of testing
# (2 Que – 15 marks)

***(2 Que)***

## 1. Define Error, Fault and Failure. Clarify with proper example for each term and their relationship.

**Error:**

- An error is a human action that produces an incorrect or unexpected result.
- In other words, an error occurs when a person makes a mistake while designing, coding, or testing software.
- For example, a programmer might accidentally introduce a typo or a logical error in the code while writing it.

**Fault:**

- A "fault" refers to a defect in the system's software that causes it to behave incorrectly or produce incorrect results.
- A fault can be caused by an error or a mistake in the software design, coding, or testing process.
- For example, a programmer might forget to initialize a variable, leading to unexpected behavior when the program is executed.

**Failure:**

- A failure occurs when the software does not behave as expected.
- A failure can be caused by one or more faults in the software.
- For example, a user might encounter a runtime error when trying to run the program, due to a fault in the code.

**Relationship between Error, Fault, and Failure:**

- Errors can lead to faults, and faults can lead to failures. In other words, an error made by a programmer can result in a fault in the code, which can in turn cause a failure when the user tries to use the software.

***(Asked 2 times)***

## 2. Mention the causes of software defects with classification.

## OR

## Describe the main reason that causes software to have flaws in them.

Software defects can be caused by various reasons, which can be classified into the following categories:

a. **Human errors:**
- These are mistakes made by developers during the software development process, such as coding errors or design flaws.
- For example, a developer may write a piece of code that contains a syntax error or may overlook a critical aspect of the software design.

b. **Environment errors:**
- These are issues caused by external factors that affect the performance of the software.
- For instance, system failures, hardware or software malfunctions, or network connectivity issues can cause software defects.

**c. Requirements errors:**

- These occur due to incorrect requirements gathering and analysis, which can result in inaccurate features.
- For example, if the requirements for a software system are not clearly defined, the software may not function as intended, leading to defects.

**d. Process errors:**

- These arise due to inefficient software development processes, which can result in incomplete or low-quality software.
- For instance, if a software development team does not follow best practices, such as testing or code reviews, the software may have defects that go unnoticed.

Software defects can occur in any phase of software development life cycle. i.e. during Requirements, Design, Coding, Integration and Testing.

## Q. Software failure with example.

- Software failure refers to the inability of a software system to perform its intended function.

- **For example**, imagine a banking application that has a bug causing it to calculate interest incorrectly.
- This could result in customers being charged the wrong amount of interest or not receiving the correct amount of interest on their deposits.
- This can cause financial losses for both the customers and the bank.
- This is an example of a software failure.

## Q. Explain at least 5 impacts of software defects considering Hospital Information System as an example.

- Impacts of software defects considering Hospital Information System as an example:
  a. Patient safety is at risk due to incorrect or incomplete medical information.
  b. Patient data may be lost, leading to inaccurate diagnoses or treatment decisions.
  c. Increased medical errors leading to lawsuits and damaged reputation of the hospital.
  d. Reduced efficiency of the healthcare system, causing delays in patient care and increased healthcare costs.
  e. Damage to the trust between patients and healthcare providers, leading to a loss of business for the hospital.

## Q. Describe, with examples, the way in which a defect in software can cause harm to a person, to the environment or to a company.

- A defect in software can cause harm to a person, environment or company in many ways, some examples are:

1. **Person**: A bug in medical software can result in incorrect diagnoses or treatments, leading to harm or even death of a patient.
2. **Environment**: A defect in software controlling a chemical plant can result in a dangerous release of toxic chemicals, causing harm to the environment and nearby communities.
3. **Company**: A flaw in a financial software system can result in incorrect transactions, leading to financial losses and damage to the reputation of the company.

Overall, defects in software can have serious consequences, and it's important to ensure software is thoroughly tested and validated to avoid harm.

## 3. Define Testing?

- Testing is a process of evaluating a software application to identify any defects, errors, or bugs in it.
- The main goal of software testing is to ensure that the software is reliable, efficient, secure, and meets the needs of the end-users.

## 4. What do you mean by Testing and why it is necessary in various sectors?

- Testing is a process of evaluating a software application to identify any defects, errors, or bugs in it.
- The main goal of software testing is to ensure that the software is reliable, efficient, secure, and meets the needs of the end-users.
- Testing involves running the software application under controlled conditions to identify any inconsistencies between the actual results and the expected results.

- Here are some reasons why testing is necessary in various sectors:
  1. To ensure software quality and reliability.
  2. To meet customer expectations and requirements.
  3. To minimize the risk of software failure or defects that could lead to harm to a person or company.
  4. To improve software performance and efficiency.
  5. To reduce the cost and time associated with fixing defects after software release.
  6. To gain a competitive edge by delivering high-quality software products.
  7. To maintain brand reputation and image.

**(Asked 4 times)**

## 5. What is the significance of software testing?

## OR

## What are the objectives of testing?

The significance/objectives of software testing are:

1. **Identify defects:**
   - This objective focuses on identifying defects or errors in the software that may cause problems when the software is used by end-users.
   - This ensures that the software functions as intended and meets the specified requirements.

2. **Verify if the software meets user requirements:**
   - This objective is to verify that the software meets the requirements and specifications provided by the client or end-users.

3. **Validate usability:**
   - This objective is to validate that the software is fit for use and can perform its intended functions efficiently and effectively.
   - This involves testing the software in real-world scenarios to ensure it is user-friendly, reliable, and meets the end-user's needs.

4. **Improve reliability:**
   - This objective aims to improve the reliability, stability, and performance of the software.
   - This is done by detecting and eliminating defects or errors that may lead to system failures or crashes, which can impact the user experience and business operations.

5. **Increase user satisfaction:**
- This objective is to increase end-user satisfaction by delivering software that meets their expectations and requirements.

6. **Minimize risk:**
- This objective aims to minimize the risk of software failure or malfunction that may result in financial losses, reputation damage, or even legal issues.
- Testing helps to identify and fix issues before they become critical, reducing the risk of negative consequences.

7. **Reduce development costs:**
- This objective is to reduce development costs by detecting and fixing defects early in the software development life cycle.
- This helps to minimize the cost of fixing defects in later stages of development, which can be more expensive and time-consuming.

8. **Facilitate continuous improvement:**
- This objective is to facilitate continuous improvement by providing feedback on the quality of the software and identifying areas for improvement.
- By testing the software and reporting issues, developers can improve the software's overall quality and performance.

## 9. Why do you consider testing as a process. Explain about the fundamental testing processes in detail.

- Testing is considered as a process because it involves a series of activities such as planning, designing, executing, and evaluating tests to ensure that the software meets the expected quality standards.
- The objectives of testing include finding defects, reducing risk, improving quality, validating, verifying requirements, and improving customer satisfaction.

- The fundamental testing process is explained below:

1. **Test planning:**
- In this stage, a plan is created that outlines the objectives, scope, and timelines of the testing process. The resources required for testing are also identified.
2. **Test design:**
- This stage involves creating a detailed plan for the testing process, which includes defining test cases and scenarios, and preparing test data.
3. **Test execution:**
- In this stage, tests are run according to the test plan, and the results are recorded.
- Any defects or issues are reported to the development team.
4. **Test reporting:**
- This stage involves summarizing the testing results, identifying any defects or issues, and communicating the status of the testing process to stakeholders.
5. **Test closure:**
- In the final stage, the testing process is analyzed to identify areas for improvement.
- A test closure report is prepared to document the testing process and results.

- The fundamental testing process is iterative, meaning that it repeats throughout the software development life cycle.

***(Asked 9 times) (Sure Que)***

## 10. Write in detail about the 7 major testing principles in detail and simple words.

## OR

## Explain about the testing principles.

The 7 major testing principles explained in simple words:

1. **Testing shows the presence of defects:**

- Testing helps to identify defects and errors in software that can cause problems when the software is used by end-users.
- For example, a testing process may reveal that a login feature in the software is not functioning as expected, making it difficult for users to access the software.

2. **Exhaustive testing is impossible:**

- It is not possible to test every possible combination of inputs that a software system may encounter.
- Therefore, testing should be focused on the most important and likely scenarios.
- For example, a software program that manages employee records may have hundreds of thousands of possible input combinations, making it impossible to test them all.

3. **Early testing saves time and money:**

- Finding and fixing defects early in the software development life cycle is less expensive and less time-consuming than finding and fixing them later in the development process.
- For example, if a software defect is identified during the testing process, it can be fixed before the software is released to end-users, saving time and resources.

4. **Defect clustering:**
- Defect clustering is the principle that a small number of modules or features in software contain a large number of defects.
- Testing efforts can be focused on them by identifying these modules or features.

5. **Pesticide paradox:**
- The pesticide paradox is the principle that if the same tests are repeated over and over again, eventually the tests will no longer find new defects.
- To avoid this, testing must be continuously updated and improved to identify new defects in the software.

6. **Testing is context-dependent:**
- Testing is done differently in different contexts.
- Different software projects will have different requirements, user expectations, technical constraints, and business goals, which can all affect the testing approach.
- Not all software systems carry the same levels of risk.
- So, different approaches are taken to test software in different contexts.
- For example, testing a mobile app for a game will have different requirements and testing needs compared to testing a banking software.

7. **Absence of errors fallacy:**
- The absence of defects during testing does not guarantee that the software is defect-free.
- There may be defects that have not yet been discovered, and there may be defects that are not testable using the current testing approach.
- Therefore, testing should be used as a tool to provide information about the quality of the software, but it cannot guarantee that the software is perfect.

## Q. Explain in what kinds of projects exhaustive testing is possible.

- Exhaustive testing, which means testing every possible combination of inputs, is not practical in most projects due to various constraints such as time, budget, and resources.
- However, in certain small-scale projects with limited functionality, exhaustive testing may be possible.
- Example: Simple calculator applications, where the number of possible input combinations is relatively small. So exhaustive testing might be possible.

***(Asked 3 times)***

## 11. Explain with appropriate scenario regarding Pesticide paradox and Pareto principle.

**Pesticide Paradox**

- The Pesticide Paradox states that if the same tests are repeated over and over again, eventually, they will no longer find any new defects.
- This happens because as the testing progresses, the software is being changed and improved, and the old tests may not be relevant anymore.
- Therefore, it is important to regularly review and update the test cases to ensure that they are effective in finding new defects.

**Pareto Principle**

- The Pareto principle, also known as the 80/20 rule, states that 80% of the effects come from 20% of the causes.
- In software testing, it means that 80% of the defects in a software system are likely caused by 20% of the modules or areas of the system.

- Therefore, focusing on testing the critical areas of the system, which constitute the 20%, can help improve the effectiveness of testing while minimizing the time and resources spent on testing the less critical areas.
- This helps in optimizing the testing effort and resources.

**For Example:**

- For example, consider a software application that is being developed for a retail company.
- The testing team initially writes a set of test cases that cover the most important functionalities of the application.
- They execute these tests and find a significant number of defects, which they then fix.
- However, if the same set of tests is repeatedly executed, the number of new defects found will eventually decrease, leading to the pesticide paradox.

- To overcome this, the testing team can use the Pareto principle to identify the most critical areas of the application that are likely to contain the majority of defects.
- They can then focus their testing efforts on these areas, using both manual and automated testing techniques.
- By doing this, they can ensure that the most important parts of the application are thoroughly tested, while avoiding the pesticide paradox.

- For instance, the testing team might identify that the payment processing module of the application is responsible for 80% of the defects.
- They can then focus their testing efforts on this module.
- This will help them find more defects in this critical area, without wasting time on less important functionalities.

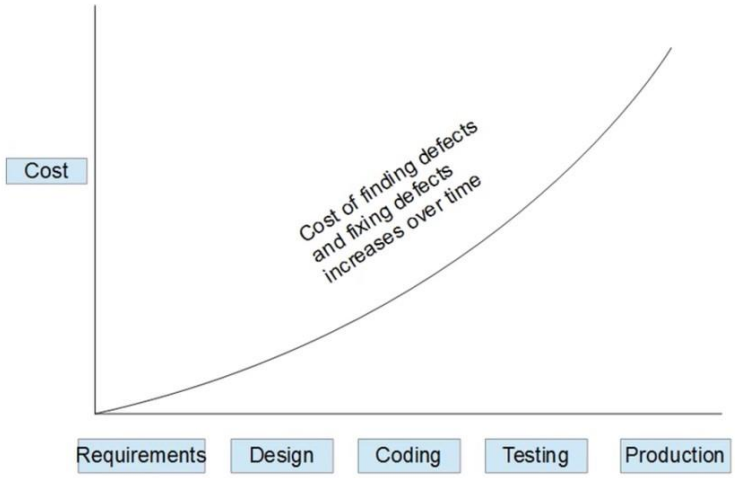
**(Asked 3 times)**

## 12. Bug cost increases over time.

### OR

### Why is it cheap to find and fix defects in the early stages of SDLC? Justify.

- The cost of fixing a bug increases over time.
- Bugs that are found earlier in the software development lifecycle are generally easier and cheaper to fix.
- If a bug is not found and fixed early, it can lead to other bugs and issues in the software, which can be even more expensive to fix.
- Finding and fixing defects early in the SDLC leads to better software quality and customer satisfaction.
- Delayed bug fixes can also result in increased testing time and resources.
- In addition, bugs that make it into production can have negative impacts on customer satisfaction and company reputation, which can also be costly to address.
- Hence, fixing defects early on in the development cycle requires less effort and fewer resources than fixing it later where the defects will spread and become more complex.



- **Example**: Imagine a scenario where a software company releases a new version of their app without thoroughly testing it.
- Users start to download the app and soon discover that there are many bugs and issues with it.
- The company now has to spend a significant amount of time and resources fixing these issues, which could have been avoided if they had tested the app more thoroughly in the early stages of the SDLC.

## 13. Ethics while testing.

- Ethics in software testing refers to the moral principles and values that should guide the behavior of software testers while conducting their work.
- Some important ethical considerations in software testing include:

1. **Respect for user privacy:**
- Testers should respect the privacy of end-users and should not collect or disclose any personal or sensitive information without their consent.
2. **Disclosure of defects:**
- Testers should report all defects they find to the appropriate stakeholders, and avoid hiding or downplaying defects to make the software appear better than it is.
3. **Transparency in testing processes:**
- Testers should be open and transparent about their testing processes, and ensure that stakeholders have access to all relevant information about the testing process and results.
4. **Avoiding conflicts of interest:**
- Testers should be transparent about the testing process and the results. They should clearly communicate any defects or issues found during testing to the relevant stakeholders.

5. **Ensuring the safety and security of the software:**
- Testers should ensure that the software is safe and secure for end-users to use.
- They should test for any potential safety hazards and report any issues that may cause harm or injury.

6. **Adherence to standards:**
- Testers should follow established testing standards and guidelines to ensure that the testing process is consistent and reliable.

## 14. Why do you think ethics is needed while testing software? Justify with any example.

- Ethics in software testing is important to ensure that the testing process is carried out in a responsible and professional manner.
- Testing software without ethics can lead to compromised security, privacy violations, and even legal issues.

- **For example,** suppose a software tester knows that a particular feature of the software is not working correctly, but they choose not to report it to their supervisor or project manager to meet the project's deadline.
- In that case, it can lead to severe consequences such as a security breach or financial loss to the end-users.
- This kind of unethical behavior can damage the tester's reputation and the reputation of the organization they work for.

## 15. How can you differentiate between automobile industry testing and software industry testing.

**Automobile Industry Testing:**

- Typically involves physical products that can be tested through physical means.
- Testing often involves stress and durability testing to ensure the product can withstand harsh conditions.
- Failure can result in injury or even death.
- Examples include crash testing, emissions testing, and safety testing.

**Software Industry Testing:**

- Involves intangible products that must be tested through digital means.
- Failure can result in financial loss, data breaches, and other negative impacts on businesses and users.
- Examples include unit testing, integration testing, and user acceptance testing.

In summary, while both industries involve testing to ensure product quality, the methods and risks associated with each type of testing can vary significantly.

## 16. "The roles of developers and testers are different." Justify your answer.

- Developers are responsible for creating the software application, while testers are responsible for ensuring that the application works as intended.
- Developers focus on writing code, debugging and fixing issues, and implementing features and functionality. Testers focus on identifying issues and defects in the application, and reporting them back to the development team for resolution.
- Developers use programming languages and tools to write and test their code, while testers use specialized testing tools and methodologies to evaluate the software.
- Developers are typically more involved in the design and architecture of the application, while testers are more focused on validating the application against specifications and requirements.
- Developers may work more closely with other members of the development team, such as designers and project managers, while testers may work more closely with quality assurance and testing teams.

# Chapter 3
# Verification and Validation
# (1 Question -7 marks)

## 1. What is Verification and Validation? (For understanding)

**Verification:**

- Verification is like checking if you've followed the instructions correctly when you're baking a cake.
- Just like how you need to follow the recipe instructions step-by-step to make a cake, software developers need to ensure that they follow the requirements specifications accurately to build a software product that meets the requirements.
- In software development, verification is the process of checking whether the software product meets its specified requirements and is error-free.
- For example, a software developer might use code review and static analysis tools to check the code for syntax errors, coding standards, and logical errors.

**Validation:**

- Validation is like tasting the cake to see if it's delicious and meets your expectations.
- Similarly, in software development, validation is the process of testing whether the software product meets the user's needs and expectations.
- For example, a tester might test the software product to ensure that it meets functional requirements, performance requirements, and usability requirements.

- ***In summary**, verification is like checking that the software product is built correctly according to the specifications, while validation is like testing whether the software product works as intended and meets the user's needs.*
- Both verification and validation are important to ensure that the software product is of high quality and meets the requirements.

## 2. Explain verification.

- In software development, verification is the process of checking whether the software product meets its specified requirements and is error-free.
- Simply, Verification is like checking that the software product is built correctly according to the specifications.
- It is usually done by performing various reviews, walkthroughs, and inspections.
- The main aim of verification is to identify defects and errors in the early stages of the software development life cycle, so that they can be fixed before the software is delivered to the end-users.
- Verification can be done manually or with the help of automated tools and techniques.
- Verification is a static testing technique, as it does not involve running the software code.
- Verification is an important part of the software development process, as it helps ensure that the software is built correctly.

**(Asked 2 times)**

## 3. How is software verification carried out?

- Software verification is carried out through a systematic process of reviewing and testing the software product at different stages of its development.
- It is done to ensure that it meets its specified requirements and is free from errors.

The verification process typically involves the following steps:

1. **Requirements analysis:**
- Requirements analysis involves analyzing software requirements to ensure they are clear, complete, and unambiguous.
2. **Design review:**
- Design review ensures that the software design meets the requirements and is consistent with industry standards and best practices.
3. **Code review:**
- After the design has been reviewed, the code is reviewed to ensure that it meets coding standards, is free from syntax errors, and is consistent with the design.
4. **Testing:**
- Once the code has been reviewed, the software product is tested to ensure that it meets the functional and performance requirements and that it is free from defects.
- This step involves running the software and testing it in a simulated environment to identify any issues that need to be addressed.

5. **Deployment:**
- Once the testing is complete, the software product is deployed to the production environment and is tested to ensure that it works as expected in the real world.

The goal of software verification is to catch errors early in the development process before the software is released to the customer.

By catching errors early, the cost and effort required to fix them are reduced, and the quality of the software product is improved.

## Q. What do you mean by verification process? With a hierarchical diagram, mention briefly its types.

Explain Que no 3.

***(Asked 2 times)***

## 4. Explain briefly about verification techniques adopted in software development organizations.

- Verification is typically carried out using different techniques, such as walkthroughs, inspections, and audits.

1. **Walkthrough:**

- A walkthrough is a type of software review in which the author of the code or software product presents their work to a group of reviewers.
- The author explains their thought process, the design choices they made, and how the code or software works.
- The reviewers provide feedback and identify any issues or areas for improvement.
- During a walkthrough, the focus is on understanding the code or software product and identifying any potential issues.
- This can be helpful in catching errors early in the development process and improving the quality of the code or software.
- It occurs early in the development process.

2. **Inspection:**

- In an inspection, a group of people, typically peers of the software developer, examine the software product in detail to identify defects.
- The inspection process is more structured than a walkthrough, and typically involves a checklist of items to be reviewed.
- The goal of an inspection is to find defects that may not have been caught during a walkthrough or testing.
- Inspection is more structured than walkthrough and it occurs later in the development process.

3. **Audit:**

- An audit is a systematic and independent examination of a software product or process to evaluate if it meets standards, specifications, and requirements.
- It is usually conducted by an external entity, such as a regulatory agency, a customer, or a third-party auditor.
- Audits can be conducted on various aspects of software development, such as code quality, documentation, security, and project management.
- The audit process typically involves a review of the software product or process, followed by a report that identifies strengths, weaknesses, and areas for improvement.

## Q. Explain about the importance of walk through, Inspections, and Audit in software verification.

**Walkthrough**: Helps identify defects and improve the quality of the software product early in the development process by reviewing the software or code by team members.

**Inspection**: Helps identify defects and improve the quality of the software product through a formal and structured review process.

**Audit**: Helps ensure that the software development process and product meet the required standards and regulations, and that appropriate documentation is in place.

***(Asked 2 times)***

## 4. What do you mean by walkthrough? Write about its various types.

- A walkthrough is a type of software review in which the author of the code or software product presents their work to a group of reviewers.
- The author explains their thought process, the design choices they made, and how the code or software works.
- The reviewers provide feedback and identify any issues or areas for improvement.
- During a walkthrough, the focus is on understanding the code or software product and identifying any potential issues.
- This can be helpful in catching errors early in the development process and improving the quality of the code or software.

There are two main types of walkthroughs: formal and informal.

1. **Formal Walkthrough:**

- A formal walkthrough is a structured and planned review process that follows a specific set of guidelines and procedures.
- It involves a team of reviewers who follow a detailed checklist to identify errors or issues in the software product or code.
- A formal walkthrough typically requires more time and effort to organize and conduct, but it can provide more thorough and consistent results.

2. **Informal Walkthrough:**

- An informal walkthrough, also known as a peer review, is a less structured and more casual review process.
- It involves a smaller group of reviewers who review the software product or code together without following a specific checklist or procedure.
- An informal walkthrough is typically quicker and easier to organize than a formal walkthrough, but it may be less thorough and consistent in its results.

In both types of walkthroughs, the author of the code or software product explains their work to the reviewers.

## 5. What is Audit? Explain about its various types.

- An audit is a systematic and independent examination of a software product or process to evaluate if it meets standards, specifications, and requirements.
- It is usually conducted by an external entity, such as a regulatory agency, a customer, or a third-party auditor.
- Audits can be conducted on various aspects of software development, such as code quality, documentation, security, and project management.
- The audit process typically involves a review of the software product or process, followed by a report that identifies strengths, weaknesses, and areas for improvement.

**For example**, a software company may be audited by a regulatory agency to ensure that its software products comply with industry standards and regulations.

The auditor will examine the software development process, the source code, and the documentation to verify that they meet the required standards.

The audit report may include recommendations for improving the software development process, such as adopting better coding practices, improving documentation, or enhancing testing procedures.

There are different types of software audits, including:

1. **Process Audit:**

- A process audit evaluates the processes and procedures used during the software development lifecycle to ensure that they are followed correctly and efficiently.

2. **Product Audit:**
- A product audit evaluates the software product itself to ensure that it meets the specified requirements and quality standards.
- The objective of a product audit is to identify any defects or issues with the software product that need to be addressed before it is released to the customer.

3. **System Audit:**
- A system audit focuses on the software system as a whole, including the hardware, software, and network components.
- It involves reviewing the system to ensure that it is secure, reliable, and meets the specified requirements and standards.

**(Asked 2 times)**

## Q. Is audit different from inspection? Explain.

Explain 4 and 5.

**Types of verification**

1. Static verification
   a. Walkthrough
   b. Inspection
   c. Audits
3. Dynamic verification
   a. Black box testing
   b. White box testing

## Q. How do you test software artifacts like requirement and design documents? What kind of defects can you unearth using such testing?

- To test software artifacts like requirement and design documents, a technique called static testing can be used.
- Static testing is a form of testing that examines software artifacts without executing the code.
- Here are some common techniques used for static testing:
  a. **Walkthrough**: This technique involves the author of the document explaining the contents of the document to a group of stakeholders, and the stakeholders asking questions or providing feedback.
  b. **Inspection**: This is a formal review process that follows a structured process to identify defects in the document.
  c. **Audit**: An audit is a systematic and independent examination of a software product or process to evaluate if it meets standards, specifications, and requirements.

The defects that can be unearthed using these testing techniques include:

a. Ambiguities or inconsistencies in the requirement or design document.
b. Missing or incomplete requirements or design elements.
c. Contradictions between the requirements or design elements.
d. Inappropriate design or architecture decisions that could lead to performance, security or scalability issues.

By detecting and addressing these defects early in the software development lifecycle, organizations can save time and resources, and deliver higher quality software products.

**(Asked 2 times)**

## 6. Why is validation needed in software products?

- Validation is the process of evaluating the final software product to ensure that it meets the user requirements.
- It is a dynamic testing process that is performed after the completion of the development process.
- A tester might test the software product to ensure that it meets functional requirements, performance requirements, and usability requirements.
- For example, if a software application is designed to perform a specific task, validation would involve testing the application to ensure that it performs that task accurately and meets the user's expectations.

Validation is needed in software product development for several reasons:

1. **Meeting Customer Requirements:**
- Validation ensures that the software product meets the requirements and expectations of the end-users or customers.

2. **Ensuring Quality:**
- Validation ensures that the software product meets the specified quality standards and is free from defects or errors.
- It helps to ensure that the software product is reliable, efficient, and easy to use.

3. **Compliance with Regulations:**
- Validation ensures that the software product meets industry standards.
- This is important for software products that are used in regulated industries such as healthcare, finance, and aerospace.

4. **Enhancing Reputation:**
- Validation helps to enhance the reputation of the software development organization by ensuring that the software product meets the specified requirements and standards.

**(Asked 2 times)**

## 7. What are various approaches for validating any software product? Mention with categories of product.

There are various approaches for validating software products, and the approach chosen will depend on the type of product being developed.

Here are some common approaches:

1. **Manual Testing:**
- This approach involves manually testing the software product to identify issues or defects.
- It is commonly used for smaller software products such as mobile apps or websites.

2. **Automated Testing:**
- This approach involves using automated tools to test the software product.
- It is commonly used for larger software products such as enterprise applications or software systems.

3. **Unit Testing:**
- This approach involves testing individual units or components of the software product to ensure that they function as expected.
- It is commonly used for software products that have a modular or component-based architecture.

4. **Integration Testing:**
- This approach involves testing the integration of different components of the software product to ensure that they work together as expected.
- It is commonly used for software products that have a distributed or interconnected architecture.

5. **Regression Testing:**
- This approach involves testing the software product after changes or updates have been made to the code to ensure that existing functionality is not affected.
- It is commonly used for software products that are updated frequently, such as web applications or mobile apps.

6. **System testing:**
- This level of testing focuses on testing the entire system as a whole, including all of its components and subsystems.
- It is used to verify that the system meets all of its functional and non-functional requirements, and to identify any defects that may exist in the system.

7. **Performance Testing:**
- This approach involves testing the performance of the software product under different conditions to ensure that it can handle the expected load or user traffic.
- It is commonly used for software products that have a high volume of users or transactions, such as e-commerce websites or financial systems.

8. **Acceptance Testing:**
- This approach involves testing the software product with end-users or customers to ensure that it meets their requirements and expectations.
- It is commonly used for commercial software products such as games, mobile apps, or consumer applications.

## Q. What are the various approaches to software validation? Explain how you validate your software design.

Explain Que no 7.

## Q. How can a software artifact be validated before delivery? Explain with some techniques.

Explain Que no 7.

**(Asked 2 times)**

## 9. If you are a project manager of a company, then how and which techniques would you perform validation to meet the project quality? Describe in detail.

- As a project manager, I would use a combination of different validation techniques to ensure that the software product meets the desired quality standards.
- The specific techniques used would depend on the project's requirements, complexity, and the stage of the development cycle.

- Here are some techniques that I would consider:

1. Manual Testing
2. Automated Testing
3. Unit testing
4. Integration Testing
5. Regression testing
6. Performance Testing
7. Acceptance Testing

***(Asked 4 times)***

## 10. Differentiate between verification and validation with proper examples.

| | Verification | Validation |
|---|---|---|
| 1. | Are we building the software, right? | Are we building the right software? |
| 2. | Helps to ensure that the software is built correctly and meets its intended functionality. | Helps to ensure that the right software is built, and that it meets the user's needs and expectations. |
| 3. | Done before validation | Done after verification |
| 4. | Aimed at preventing defects in the software. | Aimed at detecting defects in the software. |
| 5. | It is done by QA team. | It is done my testing team. |
| 6. | It is human based checking of documents and files. | It is computer-based execution of program. |
| 7. | It uses methods like walkthrough, audits and inspections. | It uses methods like black box testing, white box testing, etc. |
| 8. | It occurs during the development phase of the software lifecycle. | It occurs after the development phase of the software lifecycle. |

***(Asked 2 times)***

## 11. Explain the importance of verification and validation in an organization for any software development.

Although software verification and validation have different objectives, there are some similarities between the two:

- Both aim to ensure that the software is of high quality and meets its intended purpose.

- Verification and validation are critical in ensuring the success of any software development project.
- Here are some reasons why verification and validation are important:

1. **Improve the quality of software**: Verification and validation activities help detect and correct defects and errors in the software, ensuring that it is of high quality and performs as intended.
2. **Increase customer satisfaction:** By validating the software against the user's needs and expectations, organizations can deliver software that meets customer expectations, increasing their satisfaction with the software.
3. **Reduce development costs**: By identifying defects early in the software development process, organizations can reduce the cost of fixing defects later in the development lifecycle.
4. **Ensure compliance with standards**: Verification and validation activities help ensure that the software meets industry standards and regulatory requirements.
5. **Mitigate project risks:** Verification and validation activities help identify and mitigate project risks, ensuring that the project is completed on time, within budget, and with the desired level of quality.

# Chapter 4
# Testing Throughout the Software life Cycle
# (1 Q – 7 marks)

## 1. Why are there so many variations of development and testing models? How would you choose one for your project? What would be the selection criteria?

- There are many variations of development and testing models because different projects have different requirements, constraints, and complexities.
- Moreover, different stakeholders may have different expectations and preferences regarding project management, quality assurance, risk management, and communication.
- The choice of a development and testing model for a project should be based on several factors, such as:
    1. Project goals and objectives
    2. Project size and complexity
    3. Stakeholder requirements and expectations
    4. Development team's skills and experience
    5. Project environment and constraints

## Q. Why is Iterative and Incremental model preferred?

- An **iterative** approach means software development activities are systematically repeated in cycles known as iterations.
- A new version of the software is produced after each iteration until the optimal product is achieved.
- Continuous refinement of the area that requires improvement.

- An **incremental** approach breaks the software development process down into small, manageable portions known as increments.
- Each increment builds on the previous version so that improvements are made step by step.
- Releasing one feature at a time.

- They are used together to boost their efficiency and achieve project objectives on time.
- This model accepts changes to the application during the development cycle.

**(Asked 4 times )**

## 2. Explain the types of Test Levels.

There are generally four main levels of testing in software development, each with their own purpose and objectives:

1. **Unit testing:**
   - This level of testing focuses on testing individual units or modules of code.
   - It is typically performed by developers and is used to verify that each unit of code is working correctly according to its design specifications.

2. **Integration testing:**
   - This level of testing focuses on testing how different units or modules of code work together as a larger system.
   - It is used to identify defects that may arise when different modules are combined and to ensure that the system as a whole is functioning correctly.

3. **System testing:**
   - This level of testing focuses on testing the entire system as a whole, including all of its components and subsystems.
   - It is used to verify that the system meets all of its functional and non-functional requirements, and to identify any defects that may exist in the system.

4. **Acceptance testing:**
   - This level of testing focuses on testing the system from the perspective of the end user or customer.
   - It is used to verify that the system meets all of the user's requirements and expectations and that it is ready for deployment.

## 3. Write about Unit testing. How does unit test help in testing life cycle?

- Unit testing is a type of software testing that focuses on verifying the functionality of individual, independent units or components of a software system.
- It involves testing individual units of code in isolation to ensure that they function as expected.
- The purpose of unit testing is to catch and fix bugs in the code early in the development cycle, before integration with other code components.
- It be performed manually or through automated testing tools, using techniques such as test-driven development (TDD) or behavior-driven development (BDD).
- It is focused on testing small units of code such as functions, methods, or classes.
- It helps to improve code quality, reduce development costs, and increase overall software reliability.

- Unit testing helps in the testing life cycle by detecting bugs and issues early in the development process, allowing for quick and easy identification and correction.
- This can ultimately save time and reduce costs, as issues that are found later in the development process are generally more difficult and expensive to fix.

**For example**, suppose a developer has written a function that adds two numbers together. The unit test would pass two numbers as input to the function and check whether the output matches the expected result. If the output is correct, the unit test passes. If the output is incorrect, the unit test fails, indicating that there may be a bug or issue with the function that needs to be addressed.

## 4. Explain Integration Testing.

- Integration testing is a level of software testing that focuses on testing the interactions between different components or modules of a software system to ensure that they work together as intended.
- It is used to identify defects that may arise when different modules are combined and to ensure that the system as a whole is functioning correctly.
- It is usually performed after unit testing and before system testing.
- The purpose of integration testing is to identify and resolve defects that may arise from the integration of different components or modules of the software system.
- It verifies that the software system is integrated correctly and that the individual components are working together as expected.

- There are several approaches to integration testing such as Top-Down, and Bottom-Up.
    a. In **Top-Down approach**, the high-level modules are tested first and then the lower-level modules are added and tested one by one.
    b. In **Bottom-Up approach,** the lower-level modules are tested first and then the high-level modules are added and tested one by one.

**For example**, in a banking system, the integration testing will involve testing the interaction between different modules like account creation, transaction processing, customer data management, and security management. Integration testing ensures that all these modules work together correctly and that the data flows between them are accurate and secure.

***(Asked 2 times)***

## 5. Explain about Acceptance testing. In which phase of SDLC is this necessary?

- This level of testing focuses on testing the system from the perspective of the end user or customer.
- It is used to verify that the system meets all of the user's requirements and expectations and that it is ready for deployment.
- Acceptance testing is performed after the completion of system testing.

An **example** of acceptance testing could be testing a web-based e-commerce application before its launch.

The testers will perform various tasks, such as creating an account, searching for products, adding them to the cart, and placing orders.

They will ensure that the application meets the requirements and functions as expected.

If any discrepancies are found, they will be reported to the development team to fix them before the application's launch.

- Acceptance testing is usually performed in the final phase of the software development life cycle (SDLC), which is the deployment or release phase.
- It is done to ensure that the software meets the requirements and specifications provided by the stakeholders and is ready for deployment in the real-world environment.
- Acceptance testing is typically carried out after the completion of all other testing phases such as unit testing, integration testing, and system testing.

## 6. Explain different kinds of Test Types.

The different types of testing are:

1. **Functional Testing:**

- Functional testing focuses on testing the features and functionality of the software application.
- Simply, it is carried out to ensure that the software application meets the user's requirements and works as expected.
- Examples of functional testing include black box testing, unit testing, integration testing, and system testing.

2. **Non-Functional Testing:**

- Non-functional testing is a type of testing that evaluates the software application's non-functional aspects, such as performance, compatibility, usability and security.
- It helps to ensure that the system meets the non-functional requirements specified by the stakeholders.
- Examples of non-functional testing include performance testing, compatibility testing, security testing usability testing.

3. **Structural Testing:**

- This type of testing is focused on verifying the internal structures of the software system, such as code or architecture.
- It aims to test the structure of the software and identify any defects related to the code design.
- Examples of structural testing include White Box testing.

4. **Regression Testing:**

- This type of testing is performed to ensure that the changes made in the software system have not introduced any new defects or affected the existing functionality.
- It ensures that the software system continues to work as expected after any modifications or changes have been made.

5. **Re-testing:**

- This type of testing is performed to ensure that previously identified defects have been fixed correctly.
- It verifies that the fixes have been implemented and that no other defects have been introduced in the process.

## 6. Both black box testing and white box testing can be used in all levels of testing. Explain with examples.

**Black box Testing**

- In this method, the tester does not have any knowledge of the internal workings of the software, and tests it purely based on its external behavior or inputs/outputs.
- Testers who perform black box testing only have knowledge of the software's functional requirements and use this knowledge to design and execute test cases.
- They test the software on the basis of its inputs, outputs, and behavior, looking for bugs, errors, and defects.
- It is helpful for testing the functionality of the software from the user's perspective and can identify errors or defects that would not be detected through other testing methods.
- It is also known as functional and specification based technique.

**White Box Testing**

- White box testing involves testing the internal workings of the software code.
- Testers who perform white box testing have knowledge of the software's internal architecture and use this knowledge to design and execute test cases.
- They examine the code and test it on the basis of the code's internal behavior, looking for bugs, errors, and defects.
- It can identify structural or performance issues that may not be visible through Black Box testing and can uncover errors or defects that are specific to the codebase.
- It is also known as structured based technique.

- Yes, both black box testing and white box testing can be used in all levels of testing in software development.
- Here are some examples of how they can be used:

1. Unit Testing:
- Black box testing can be used to test the input/output behavior of a function or module.
- White box testing can be used to test the internal logic of a function or module.

2. Integration Testing:
- Black box testing can be used to test the interaction between different modules or components.
- White box testing can be used to test the integration of different modules or components.

3. System Testing:
- Black box testing can be used to test the overall functionality of the system from an end-user perspective.
- White box testing can be used to test the system's internal behavior and performance.

***(Asked 2 times)***

## 7. How can a product be tested using black box testing?

- Black box testing is a type of testing in which the tester tests the product without any knowledge of the internal workings of the product.
- The tester only focuses on the inputs and outputs of the product.

To test a product using black box testing, the tester follows these steps:

1. **Identify the input**:
- The tester needs to identify the input data that needs to be provided to the product to test its functionality.

2. **Determine the expected output:**
- The tester needs to determine the expected output for the given input data.

3. **Execute the test case**:
- The tester then executes the test case by providing the input data to the product and comparing the actual output with the expected output.

4. **Report defects**:
- If the actual output is different from the expected output, the tester reports the defect to the development team.

For **example**, suppose a tester is testing a calculator application using black box testing.

1. Tester identifies input data as numbers and operations.
2. Expected output is the result of the operation performed by calculator.
3. The tester enters input data and compares the actual result with the expected result.
4. If the actual result is different from the expected result, the tester reports the defect to the development team.

***(Asked 4 times)***

## 8. Differences between Functional testing and Non-Functional testing. Explain different types of testing that falls under them.

- Functional testing and non-functional testing are two types of testing that are performed on software applications to ensure their quality and reliability.
- Both are essential in ensuring the quality and reliability of software applications.

1. **Functional Testing:**

- Functional testing focuses on testing the features and functionality of the software application.
- Simply, it is carried out to ensure that the software application meets the user's requirements and works as expected.
- The goal of functional testing is to identify defects in the software application's functionality that could cause it to not work as intended.

Examples of functional testing include:

1. Unit testing
2. Integration testing
3. System testing
4. Black box testing

The importance of functional testing is:

1. Ensuring product quality
2. Meeting user expectations
3. Preventing defects
4. Improving user experience
5. Cost-effectiveness

2. **Non-functional Testing:**

- Non-functional testing is a type of testing that evaluates the software application's non-functional aspects, such as performance, compatibility, usability and security.
- It helps to ensure that the system meets the non-functional requirements specified by the stakeholders.

Examples of non-functional testing include:

1. **Performance Testing**: It verifies how well the software performs under a given workload, including its speed, scalability, and stability.
2. **Security Testing:** It ensures that the software is secure against unauthorized access and data breaches.
3. **Compatibility Testing**: It checks whether the software can run on different platforms and browsers.
4. **Usability Testing**: It checks whether the software is user-friendly and easy to use.

The importance of non-functional testing is:

1. Enhancing User Experience:
2. Reducing Business Risks
3. Improving Software Quality
4. Cost Reduction

## Q. How does Non-Functional testing assist in software testing Life cycle?

- Non-functional testing ensures that the software product meets the user's expectations in terms of performance, usability, security, etc.
- It also helps to identify and mitigate risks and issues that may affect the product's quality and value.
- By performing non-functional testing at different stages of the software testing life cycle, such as during development, testing, and production, it is possible to detect and address issues early, which can save time, cost, and resources.

***(Asked 2 times)***

## 9. Differentiate between Retesting and Regression testing.

1. **Regression testing**

- Regression testing is performed to ensure that the changes made to the software, such as new features or bug fixes, have not affected the existing functionality of the software.
- It involves rerunning previously executed test cases to ensure that the existing features still work as expected.
- Tests are done on the unchanged or existing functionality of the software after changes have been made to other areas.
- This ensures that the software product is still working as expected.

2. **Re-testing**

- Re-testing is performed to ensure that a defect that was previously identified and fixed in the software has been successfully resolved.
- The test cases used for retesting are the same as the ones used during the previous testing to ensure that the same steps are followed.
- Retesting is usually performed after the developers have fixed the reported issues and released a new version of the software.
- The results of retesting are compared with the previous test results to ensure that the issues have been resolved.

In summary, differences between regression testing and retesting are:

- Regression testing is performed after a change is made to the software, while retesting is performed after a bug has been reported and fixed.
- Regression testing involves rerunning previously executed test cases to ensure that the existing functionality still works as expected, while retesting involves running the same test case that initially failed due to a defect to ensure that it has been fixed.

## Q. Whose duty is it to prepare the left side of V model, and who prepares right side of V model and how it contributes to software quality.

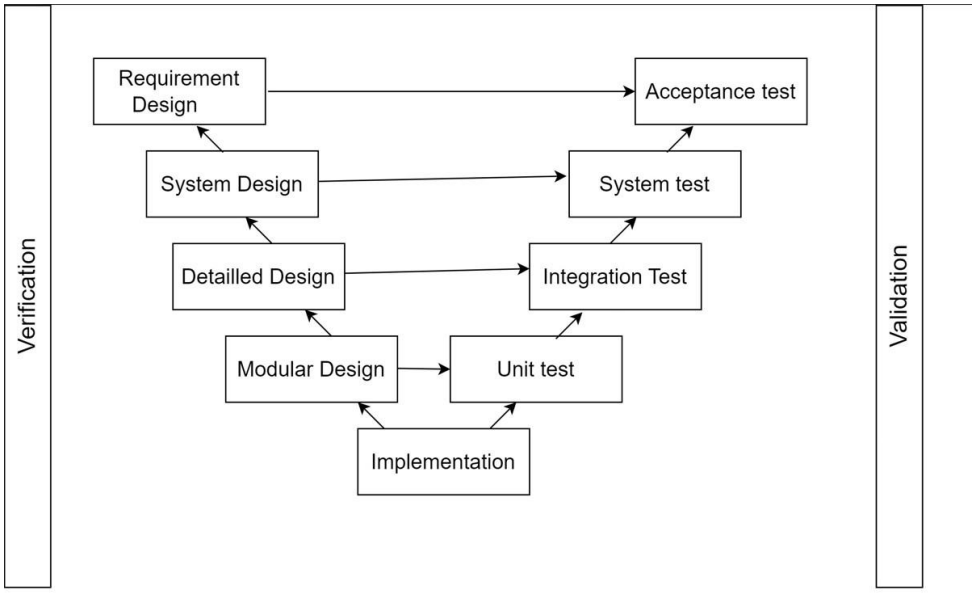



Fig: V model

The left side of the V model is prepared by Development team whereas right side of the V model is performed by the Testing team.

Left side of the V model:

1. Requirement analysis
2. System design
3. Detailed design
4. Implementation

Right side of the V model:

1. Unit testing
2. Integration testing
3. System testing
4. Acceptance testing

## Q. What kind of testing is performed when you have deadline approaching near and you have not tested anything? Explain the importance of such testing.

- When a deadline is approaching near and there has been no testing done, the most appropriate type of testing to perform would be functional testing.
- This is because functional testing focuses on ensuring that the software application meets the requirements and functions as intended.

- The importance of performing functional testing in such a scenario is to quickly identify any major defects or issues with the software application that may prevent it from meeting the desired requirements.
- By identifying these issues early on, the development team can work towards resolving them and ensuring that the application meets the necessary requirements and is fit for release.

- It is important to note that while functional testing can help identify major defects, it is not a substitute for thorough testing.
- Ideally, testing should be planned and executed throughout the software development life cycle to ensure that defects are identified and addressed as early as possible.

## Q. Do you think management can save money by not keeping test specialist? How does it impact the delivery deadlines and remove collection?

- No, I don't think management can save money by not keeping test specialists.
- Test specialists play a crucial role in ensuring the quality of the software product.
- By not having test specialists, the software product may have more defects, which could lead to delays in delivery and increased costs due to rework.

- Without test specialists, the development team would have to spend more time testing, which would lead to slower delivery times.
- Additionally, without test specialists, the management team would not have accurate information on the quality of the software product, which could lead to incorrect decisions and negative impact on customer satisfaction.

- Moreover, by not keeping test specialists, management may also miss out on opportunities to improve the quality of their product and reduce future costs by identifying and addressing defects early in the development cycle.
- In the long run, not having test specialists could actually end up costing the company more money and hurting their reputation in the market.

## Q. "The proof of the pudding is in the eating." Justify this statement based on the approaches a software testing industry follow to fail-proof their system. What are the techniques followed to achieve such success?

- The statement "The proof of the pudding is in the eating." means that the true value or quality of something can only be judged or proven by experiencing or using it.
- This statement is applicable to the software testing industry as well.
- In order to fail-proof their systems, software testing industries follow various approaches and techniques such as:

1. Test Planning
2. Test Design
3. Test Execution
4. Defect Management
5. Continuous Improvement

- By following these approaches and techniques, software testing industries can ensure that their systems are fail-proof and meet the quality standards.
- The statement "the proof of the pudding is in the eating" applies here as well, as the quality of the software can only be proven by using and experiencing it, which is made possible by the rigorous testing process.

# Chapter 5
# Static techniques
# (1 Q – 7 marks)

***(Asked 2 times)***

## 1. Difference between Static and Dynamic testing.

1. **Static Testing:**

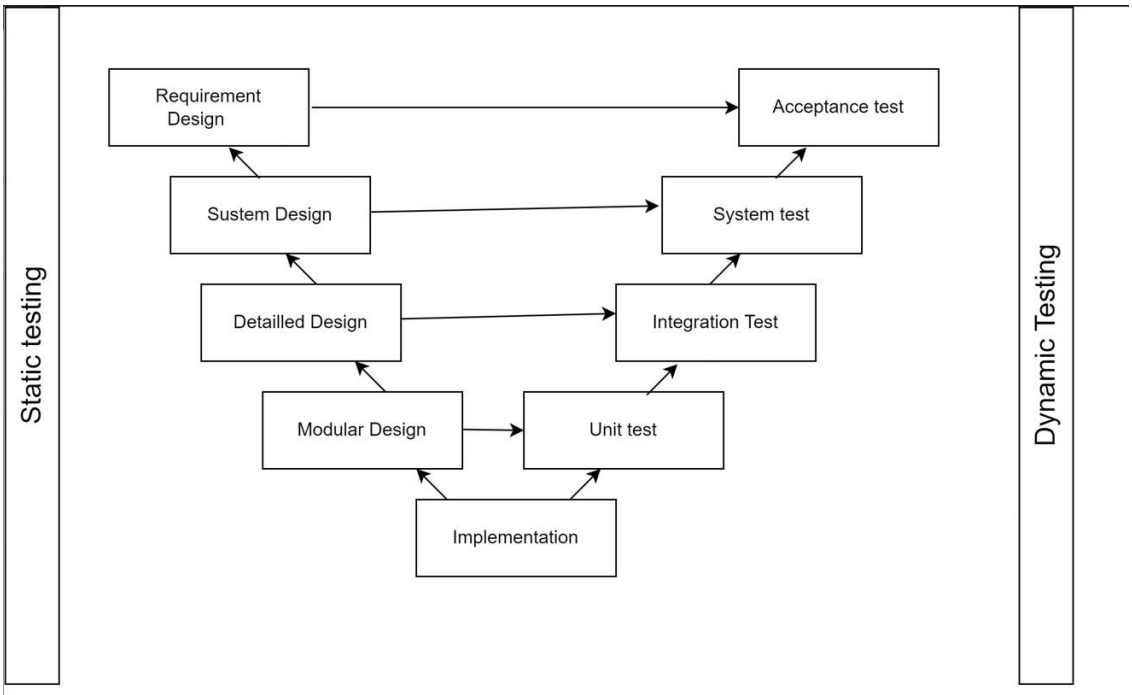
- Static testing is a technique in which software is tested without executing the code.
- This technique is performed during the early stages of the software development life cycle (SDLC).
- Static testing can be applied to any artifact produced during the software development process, such as requirements documents, design documents, code, and test cases.
- This technique can be performed manually or with the help of automated tools.
- Walkthroughs, inspections, reviews and audits are considered static testing techniques as they do not involve running the software.
- Instead, they focus on reviewing and analyzing the software documentation, code, and design.
- Static testing helps in finding defects at an early stage, which reduces the cost of fixing them in the later stages.
- It is performed in verification stage.

2. **Dynamic Testing:**

- Dynamic testing is a technique in which software is tested by executing the code.
- This technique is performed during the later stages of the SDLC.

- Dynamic testing involves testing the software against functional and non-functional requirements.
- This technique helps in finding defects that are difficult to detect through static testing.
- It is performed in validation stage.
- Unit testing, integration testing, system testing, performance testing, etc. are performed in dynamic testing.

- In summary, static testing is performed without executing the code and helps in finding defects early in the SDLC, while dynamic testing involves executing the code and helps in testing the software against functional and non-functional requirements.
- Both techniques are important for ensuring the quality of software.




**Static Vs Dynamic testing**

## 2. What are the impacts of static and dynamic testing?

- Static testing helps to prevent defects, while dynamic testing helps to detect defects that were not prevented by static testing.
- This, in turn, leads to higher quality software that meets the user's requirements.

**(Asked 3 times)**

## Q. Why is static technique different from dynamic testing?

Explain Que no 1.

**(Asked 6 times)**

## 3. Static techniques find causes of failures. Justify it.

- Static techniques such as reviews, inspections and walkthroughs allow testers and developers to review and analyze the code, design, and documentation before the actual execution of the software.
- By reviewing the code and documentation, the team can identify defects at an early stage, before they can cause failures during execution.
- By identifying and addressing these issues early in the development process, the team can prevent or reduce the number of failures that might occur during execution.
- This technique is performed during the early stages of the software development life cycle (SDLC).
- Static testing can be applied to any artifact produced during the software development process, such as requirements documents, design documents, code, and test cases.

- **For example**, suppose a team identifies a logical error in the code during a walkthrough.
- By fixing this error before the software is executed, the team can prevent the software from producing incorrect results during testing or when it is released to users.

The significance of static testing is:

1. Early detection of defects
2. Improved code quality
3. Improved team collaboration
4. Compliance with industry standards

***(Asked 2 times)***

## 5. Explain Formal Review process. What are the main roles in the review process?

- The formal review process is a systematic examination of a software artifact (such as requirements specification, design documents, source code, test cases, etc.) by a group of individuals who are not directly involved in its creation.
- The main purpose of the review process is to find defects, improve the quality of the artifact, and share knowledge among the participants.
- Examples of review process are walkthrough, peer code review and inspection.

The main roles in the review process are:

1. **The Author:**

- The person who writes the code, document or any other work product that needs to be reviewed.

2. **The Moderator:**

- Moderator is also known as the review leader, who leas the review process.
- He/She schedules the review, plans the review, manages the review process and ensures that the review is conducted efficiently.

3. **The Reviewer:**
- The person who examines the work product to identify defects and suggests improvements.

4. **The Scribe (Recorder):** (Once asked in SQ)
- Scribe is also known as the recorder.
- His main role in the meeting is to record defects and suggestions for process improvement.

5. **The Manager:**
- The manager ensures that it is conducted according to the organization's policies and procedures and provides support and resources to the review team.

## Q. Describe the activities in the review process.

The main activities in formal review include:

1. **Planning:**
- This involves defining the objectives, scope, and participants for the review.
2. **Kickoff meeting:**
- This is a meeting held at the beginning of the review to inform the participants about the software being reviewed, the objectives of the review, and the review process.
3. **Review process:**
- This is the main activity where the participants review the software and identify any issues or defects.
4. **Rework:**
- This involves addressing the issues and defects identified during the review process. The software is modified or reworked to fix the issues and improve its quality.
5. **Follow-up:**
- This involves verifying that the issues identified during the review have been addressed and that the software has been improved as a result of the review.

## Q. What are the success factors for review?

Some of the success factors for the review process are:

1. **Well-defined objectives and scope:**
   - The objectives and scope of the review should be clearly defined and communicated to all stakeholders involved.

2. **Trained reviewers:**
   - The reviewers should be trained in the review process, the type of review being conducted, and the specific tool being used for the review.

3. **Effective communication:**
   - Effective communication between the reviewers and the author is crucial for the success of the review process.

4. **Timely feedback:**
   - The feedback provided by the reviewers should be timely so that the author can make necessary changes to the document before it progresses to the next phase.

5. **Formal review process:**
   - A formal review process ensures that the review is conducted systematically and consistently, and that the review results are documented and tracked.

6. **Management support:**
   - Management support is essential to ensure that the review process is given adequate resources and attention.

7. **Continuous improvement:**
   - The review process should be periodically evaluated and improved based on feedback and lessons learned.

## Q. Explain some static analysis tools.

- Static analysis tools are software tools that examine the source code or other artifacts of a program without executing it.
- Some examples of static analysis tools are:

1. **Code reviews:**

- This is a manual static analysis technique where developers or team members review code for defects and other issues.

2. **Lint:**

- A static analysis tool that checks source code for errors and warnings.
- Lint tools can check for issues such as syntax errors, uninitialized variables, etc.

3. **Code complexity tools:**

- These tools analyze the code to determine how complex it is.
- This can help identify areas of the code that may be difficult to maintain or understand.

4. **Security scanners:**

- These tools analyze the code for security vulnerabilities such as SQL injection, cross-site scripting, and buffer overflows.

## Q. Explain Process Quality

- Process quality refers to how well a software development process is defined, executed, and managed.
- It involves ensuring that the development process is efficient, effective, and consistent in producing high-quality software.
- Process quality can be achieved through the use of industry best practices, such as Agile and DevOps methodologies, and adherence to standards such as ISO and CMMI.
- Effective process quality can result in increased productivity, reduced costs, improved customer satisfaction, and better software quality overall.
- Regular monitoring and measurement of the development process are necessary to identify areas for improvement and ensure continuous process quality.

# Chapter 6
# Test Design techniques
# (2 Que – 15 marks)

## 1. How is any test design prepared?

The process of preparing a test design typically involves the following steps:

1. **Understand the system or software being tested:**
   - Before creating any tests, it's important to have a clear understanding of the system or software being tested, as well as any relevant requirements or criteria that must be met.

2. **Define testing objectives:**
   - Identify the specific goals of the testing, such as verifying that the system meets certain requirements or finding and fixing defects.

3. **Plan the testing:**
   - Determine the scope of the testing, including which features or functions will be tested and how the tests will be conducted.

4. **Design the tests:**
   - Based on the testing objectives and scope, create a set of test cases that cover all relevant scenarios.
   - Test cases should include both positive and negative test cases.

5. **Implement the tests:**
   - Execute the test cases according to the test plan.
   - This may involve manual testing, automated testing, or a combination of both.

6. **Report and track issues:**
   - Document any issues or defects that are identified during testing and report them to the development team for resolution.

***(Asked 2 times)***

## 2. What are test design techniques. What are the categories of test design techniques.

## OR

## Explain Specification based technique, Structured based technique and Experience based technique.

- Test design techniques are methods used by software testers to create test cases that are designed to effectively test a particular aspect or feature of the software being tested.
- These techniques help to ensure that the testing is thorough, efficient, and effective in uncovering potential issues or defects in the software.
- There are three main categories of test design techniques.

1. **Specification-based techniques or black box techniques:**

- Specification-based techniques, also known as black-box testing techniques, are based on analyzing the specifications or requirements for the software being tested.
- This method does not require knowledge of the codebase.
- These techniques include equivalence partitioning, boundary value analysis, decision table testing, state transition testing, and use case testing.

2. **Structured-based techniques or white box techniques:**

- These techniques are based on analyzing the internal structure or code of the software being tested.
- They require knowledge of the programming languages and implementation details used in the software.
- These techniques include statement coverage, branch coverage, condition coverage, and more.

3. **Experience-based techniques:**

- Experienced-based techniques rely on the tester's knowledge, skills, and experience to identify potential issues and defects in the software.
- The objective of experienced-based testing is to find defects that may not be identified through other testing techniques.

***(Asked 3 times)***

## 4. Explain Specification based techniques with examples for each technique.

- Specification-based techniques, also known as black-box testing techniques, are based on analyzing the specifications or requirements for the software being tested.
- This method does not require knowledge of the underlying code or implementation details.
- These techniques include equivalence partitioning, boundary value analysis, decision table testing, state transition testing, and use case testing.
- Each of them is explained below:

1. **Equivalence Partitioning**

- Equivalence partitioning is a technique used to reduce the number of test cases required for testing a range of input values.
- It involves dividing the input values into groups or partitions that are expected to behave similarly.
- This technique can help to ensure that all possible scenarios are tested with a minimum number of test cases.

- **Example**: Suppose that a system accepts numeric values between 1 and 100.
- Using equivalence partitioning, the input values can be divided into three partitions: values less than 1, values between 1 and 100, and values greater than 100.
- Test cases can then be created to test each of these partitions.

### 2. Boundary Value Analysis

- Boundary value analysis is a test design technique that involves testing the boundary values of input parameters.
- It involves selecting test cases that are at the boundaries of input ranges and testing them to ensure that the system behaves as expected.

- **Example**: Suppose that a system accepts numeric values between 1 and 100.
- Using boundary value analysis, test cases can be created for the boundary values, values just below the boundary and values just above the boundary i.e. 0,**1**,2, 99,**100**,101.

- If the input is 1, the system behaves correctly.
- If the input is 100, the system behaves correctly.
- If the input is just above 1, the system behaves correctly.
- If the input is just below 100, the system behaves correctly.
- If the input is below 1 or above 100, the system behaves incorrectly and displays an error message.

### 3. Decision Table Testing

- Decision table testing is a technique used to test complex business logic or decision-making processes.
- It involves creating a table that lists all possible combinations of inputs and their corresponding outputs.
- Then testing each combination to ensure that the system behaves as expected.

- **Example**: Suppose that you are testing a system that calculates the price of a product based on its type and quantity.

The system has the following rules:

1. If the product is a book, the price is $10 per unit.
2. If the product is a toy, the price is $5 per unit for quantities of up to 50 units and $4 per unit for quantities above 50 units.
3. If the product is a game, the price is $20 per unit for quantities of up to 25 units and $18 per unit for quantities above 25 units.

Using decision table testing, you would create a table that lists all the possible combinations of product types and quantities, along with their corresponding prices.

For example:

| **Product Type** | **Quantity** | **Price** |
|---|---|---|
| **Book** | Any | $ 10 |
| **Toy** | 1 to 50 | $ 5 |
| **Toy** | 50 | $ 4 |
| **Game** | 1 to 25 | $ 20 |
| **Game** | 25 | $ 18 |

By testing each combination in the decision table, you can ensure that the system calculates the correct price for each combination of product type and quantity.

4. **State Transition Testing**

- State transition testing is a technique used to test software that has different states or modes of operation.
- In this technique, we create test cases that ensure that the software transitions correctly from one state to another.

**Example**: Consider a traffic light system that has three states: green, yellow, and red.

Using state transition testing, you would create test cases to ensure that the system behaves correctly when transitioning between these states.

For example, you might create test cases to verify the following:

1. The system must start in the green state and transition to the yellow state after a certain amount of time.
2. The system must transition from yellow to red state after a certain amount of time, and then back to green state after a certain amount of time.
3. The system must not allow any other state transitions, such as from red directly to green, or from yellow back to green.

By testing these state transitions, you can ensure that the traffic light system behaves as expected in all possible scenarios.

4. **Use Case Testing**

- Use case testing is a technique used to test the system's behavior in different scenarios or use cases.
- Use cases describe the different ways in which users interact with the system to achieve their goals.

- **Example**: Suppose that a system is designed to allow users to create and edit documents.
- We would identify different scenarios that the system is intended to support, such as creating a new document, editing an existing document, deleting a document, and so on.
- Test cases can then be created to test each of these scenarios to ensure that the system behaves as expected.

## Q. Suppose you have a login form with inputs email and password fields. Draw a decision table for possible test conditions and outcome using decision table testing.

Here is a decision table for testing a login form with email and password fields using decision table testing:

| Test Conditions | Valid Email | Valid password | Expected Outcome |
|---|---|---|---|
| Condition 1 | Yes | Yes | Login Successful |
| Condition 2 | No | Yes | Invalid Email |
| Condition 3 | Yes | No | Invalid password |
| Condition 4 | No | No | Invalid Email and Password |

- In this table, we have identified four possible test conditions for the login form.
- The first column describes each test condition, while the second and third columns indicate whether the email and password fields are valid, respectively.
- The fourth column describes the expected outcome for each test condition.

1. If the user enters a valid email and password, the login should be successful.
2. If the user enters an invalid email, the outcome should be "Invalid Email", and if the user enters an invalid password, the outcome should be "Invalid Password".
3. If both the email and password are invalid, the outcome should be "Invalid Email and Password".

By using this decision table, we can create test cases for each test condition to ensure that the login form behaves as expected under all possible scenarios.

## 5. What is structured based technique? What types of test fall under it?

- Structured-based techniques, also known as white-box testing techniques, are used to test software based on its internal structure or design.
- These techniques involve analyzing the software's code or design to identify the different paths or branches that the software can take during execution and creating test cases to ensure that all these paths are exercised.
- They require knowledge of the programming languages and implementation details used in the software.
- Structured-based techniques are useful for testing software that is complex or critical, such as software used in aerospace, defense, and medical applications.
- By testing the software's internal structure, we can ensure that it is reliable, secure, and performs as expected in all the different scenarios that it is designed to handle.
- Two types of tests that fall under structured-based techniques are statement coverage and decision coverage.

1. **Statement coverage**

- Statement coverage aims to execute every statement in the code at least once.
- In this technique, we create test cases to ensure that each statement in the code is executed at least once during testing.

**For example**, let's say we have a code block that performs a simple addition of two numbers.

To achieve statement coverage, we would create test cases that execute every line of code in the block at least once.

2. **Decision Coverage**

- Decision coverage aims to execute every possible decision in the code at least once.
- In this technique, we create test cases to ensure that every possible decision point in the code is executed at least once during testing.

**For example**, let's say we have a code block that performs a simple comparison of two numbers.

To achieve decision coverage, we would create test cases that test both possible outcomes of the comparison to ensure that the code behaves as expected in all cases.

**(Asked 2 times)**

## 6. Highlight the difference between structure based and specification-based technique.

**Specification-based Testing:**

- Focuses on the requirements and specifications of the software.
- Includes techniques like equivalence partitioning, boundary value analysis, and decision table testing.
- The output is test cases that verify that the software meets the requirements.
- Objective is to ensure that the software meets the specified requirements.

**Structured-based Testing:**

- Focuses on the internal structure and design of the software.
- Includes techniques like statement coverage and decision coverage.
- The output is test cases that ensure the software is implemented correctly.
- Objective is to ensure that the software is implemented correctly.

## 7. What is an experienced based technique? Reflect its importance.

- Experienced-based techniques rely on the tester's knowledge, skills, and experience to identify potential issues and defects in the software.
- It is also known as ad-hoc testing.
- The objective of experienced-based testing is to find defects that may not be identified through other testing techniques.
- Experienced based technique does not rely on pre-defined test cases or scripts, but instead allow the tester to explore the software freely and use their knowledge and intuition to identify defects.
- They are useful in these situations.
    a. where the requirements are unclear or not well defined.
    b. when the software is too complex to test thoroughly with other techniques.
- One of the examples of experienced-based techniques is error guessing.

The importance of experienced based techniques is:

1. Finding defects that other testing techniques may miss.
2. Saving time and effort: Other techniques, including writing test cases are more expensive.
3. Enhancing tester skills
4. Providing valuable feedback

## Q. What kind of systems are suitable to test using experience-based technique? Why?

1. Complex Systems

2. System with poorly defined requirements

## Q. What kind of characteristics must experience based tester possess?

1. Good communication skills
2. Strong problem-solving skills
3. Good understanding of the software
4. Creative thinking

# 8. Experience based techniques are used to complement black box and white box testing technique. Explain.

- Experienced based techniques enhance the black box and white box testing.
- It adds to them and makes the overall testing process more thorough and effective.
- Experienced-based testing can help identify potential issues and defects that may have been missed by pre-defined tests used in black box and white box testing.
- By using experienced-based testing in combination with these other techniques, testers can identify more defects and issues, resulting in a higher quality and more reliable software product.
- This can ultimately improve the quality and reliability of the software and help ensure that it meets the needs and expectations of the users.

***(Asked 4 times)***

## 9. How do you choose which testing techniques are best? Justify your answer technically.

## OR

## What are the criteria for selecting a particular test technique for a software?

The criteria for choosing various testing techniques are explained below:

1. **Test Objectives:**

- The test objectives determine the type of testing technique that should be used.
- For instance, if the objective is to test the functionality of the software, black box testing may be the best option.
- On the other hand, if the objective is to test the internal logic of the software, white box testing may be more suitable.

2. **Software requirements:**

- The complexity and nature of the software requirements can influence the selection of a test technique.
- For instance, if the requirements are well-defined and unambiguous, then specification-based techniques can be used.
- However, if the requirements are not well-defined, then experienced based testing may be more appropriate.

3. **Software design:**

- The software design can also influence the selection of a test technique.
- For instance, if the software has a complex architecture, then white box testing techniques such as structural testing may be more appropriate.

4. **Time and Budget Constraints:**
- The amount of time and budget available for testing also plays a crucial role in selecting the appropriate testing technique.
- For instance, if the time and budget are limited, it may be more efficient to use experienced-based testing to quickly identify potential issues and defects.

5. **Risks involved:**
- The risks associated with the software can also influence the selection of a test technique.
- For example, if the software has critical functionalities, then structural testing might be a good option.

***(Asked 3 times)***

## 10. What internal and external factors influence the decision to choose the resting technique?

## OR

## What are internal and external factors that impact the decision for test technique?

The decision to choose a testing technique for a software project can be influenced by various internal and external factors, including:

**Internal Factors:**

1. The project's requirements and objectives.
2. The complexity of the software being developed.
3. The size of the software being developed.
4. The available testing resources (e.g., time, budget, personnel)
5. The software development lifecycle model being used (e.g., Agile, Waterfall)

**External Factors:**

1. The domain in which the software is being developed.
2. Industry standards and regulations.
3. Customer requirements.
4. The availability of testing tools and technologies.
5. The market competition and pressure to deliver high-quality software.

All of these factors can play a role in the decision-making process when it comes to selecting the appropriate testing technique for a software project.

# Chapter 7
# Test Management
# (1 SQ – 5 marks)

***(Asked 4 times)***

## 1. Explain In-house project and project for clients in detail.

**1. In-house projects**

- In-house projects refer to software development projects that are developed and maintained by an organization's own team of developers.
- These projects are usually developed for internal use.
- In-house projects may have a smaller scope compared to client projects.
- In-house projects may have more flexible timelines as they are developed for internal use only.

**Example**

1. Developing an internal project management tool for use by employees.
2. Internal HR system for tracking employee data
3. Enterprise resource planning (ERP) system to manage various business processes.

**Advantages of in-house projects:**

1. Greater control and flexibility over the development process.
2. Better alignment with the organization's business needs and goals.
3. Higher level of security and confidentiality of the project's sensitive data.
4. Better collaboration and communication among team members.

2. **Project for clients**

- Projects for clients are software development projects undertaken by a company for external clients or customers.
- The software development process is focused on meeting the specific needs and requirements of the client.
- These projects are usually developed for external use, and the project team consists of employees of the company as well as external stakeholders such as clients, vendors, or contractors.
- Client projects are funded by the client.
- Client projects may have stricter timelines due to agreements (contract) with the client.

**Examples of projects from clients:**

1. Developing a website for a client's business.
2. Testing a mobile application for a client's product.
3. Developing a software application for a client's customers.

**Advantages of working in project for clients**

1. **Exposure to new technologies and industries**: Working on client projects can expose you to new technologies and industries that you may not have experienced before.
2. **Opportunity for collaboration**: Working with clients often involves collaboration with people from different backgrounds and experiences, which can be beneficial in terms of knowledge sharing and networking.
3. **Potential for a larger audience**: Developing software for clients can potentially reach a larger audience and gain more exposure.
4. **Opportunity for revenue generation:** Developing software for clients can generate revenue for your organization through project fees and potential future business opportunities.
5. **Provides a sense of achievement**: Successfully delivering a project for a client can provide a sense of achievement and satisfaction for the team involved.

## 3. Explain the benefits and drawbacks of independent testing within an organization.

- Independent testing within an organization is a process of testing software or applications by a separate team that is not involved in the development process.
- Here are some benefits and drawbacks of independent testing within an organization:

**Benefits:**

1. **Unbiased perspective**: Independent testers are not involved in the development process, so they can provide an unbiased perspective and identify issues that developers may overlook.

2. **Improved quality**: Independent testers can help improve the quality of the software or application by identifying defects and ensuring that it meets the requirements.

3. **Reduced risk**: Independent testing can help reduce the risk of defects being released into production, which can result in costly rework and damage to the organization's reputation.

**Drawbacks:**

1. **Additional cost**: Independent testing requires additional resources and budget, which can increase the overall cost of the project.

2. **Time-consuming**: Independent testing can be time-consuming, as it involves a separate testing process that must be performed after the development process.

3. **Communication challenges:** Independent testers may have limited communication with the development team, which can lead to misunderstandings and delays in the testing process.

***(Asked 4 times)***

## 4. Explain about various types of testers.

The various types of testers are:

1. **Manual Testers:**
- Manual testers execute test cases and test scenarios manually without using any automation tools or scripts.

2. **Automation Testers**:
- Automation testers use automation tools and scripts to execute test cases and scenarios.

3. **Performance Testers:**
- Performance testers are responsible for testing the performance, scalability, and reliability of software applications and systems.

4. **Security Testers**:
- Security testers are responsible for testing the security of software applications and systems.

5. **User Acceptance Testers**:
- User acceptance testers are responsible for testing software applications and systems from an end-user perspective.

6. **Mobile Testers**:
- Mobile testers are responsible for testing software applications and systems on mobile devices, such as smartphones and tablets.

***(Asked 3 times)***

## 5. Write the role and responsibilities of a test leader in a simple word.

OR

## What are the key components to be noted by any Test Leader.

- The role of a test leader is to lead the testing effort and ensure that the testing process is effective and efficient.
- Here are some of the responsibilities of a test leader:

1. **Test Planning:**
- Creating a plan for testing that explains what they will do, how they will do it, and when they will do it.

2. **Test Strategy:**
- Making a strategy for testing that explains the tools and methods they will use to make sure the testing works well.

3. **Test Execution:**
- The test leader makes sure the test cases are executed on time, and any problems are tracked and fixed.

4. **Test Reporting:**
- The test leader is responsible for creating test reports that explain how the testing is going, what problems they found, and what still needs to be done.

5. **Communication**
- Talking to the people involved in the project, like developers and managers, to make sure they know what is happening with the testing.

6. **Team Management:**
- The test leader is responsible for managing the testing team, including assigning tasks, providing guidance and support, and ensuring that the team is motivated and working effectively.

7. **Continuous Improvement:**
- The test leader is responsible for identifying areas for improvement in the testing process and implementing changes to improve efficiency.

## Q. If you were a test manager for a university examination software system, how do you perform your testing activities? Describe in detail.

Explain Que no 5.

**(Asked 3 times)**

## 6. Write roles and responsibilities of a test technician.

## OR

## Life of a Software Tester

- A test technician/software tester is a member of the testing team.
- The role of a software tester involves supporting the testing process and ensuring that testing is carried out accurately and efficiently.
- Here are some of the responsibilities of a software tester:

1. **Test Planning:**

- A software tester spends a considerable amount of time planning and designing tests.
- They work closely with the test lead or test manager to understand the project's requirements, define test cases, and create test plans.

2. **Test Execution:**

- A software tester is responsible for executing test cases, following test scripts, and recording test results.

3. **Test Reporting:**

- A software tester creates reports that document the test results and defects identified during testing.

4. **Communication:**

- A software tester communicates regularly with other members of the testing team, developers, project managers, and business owners to ensure that they are aware of the testing progress and any issues or concerns.

5. **Learning:**
- A software tester must keep up to date with new testing methods, tools, and technologies.
- They attend training sessions, read industry publications, and participate in professional development activities to improve their skills.

6. **Problem-Solving:**
- A software tester is responsible for identifying defects and working with the development team to resolve them.
- They must be able to troubleshoot and identify the root cause of issues quickly.

7. **Continuous Improvement:**
- A software tester must be able to identify areas for improvement in the testing process and implement changes to improve the efficiency and effectiveness of the testing effort.

## 8. List out test planning and estimation activities?

Sure, here are the test planning and estimation activities:

1. **Test Planning:**
   - The first step in test planning and estimation is to create a comprehensive test plan that outlines the testing approach, objectives, scope, schedule, and resources required for the testing effort.

2. **Test Strategy:**
   - The test strategy defines the testing methods, tools, and techniques to be used to ensure that the testing is effective and efficient.

3. **Test Estimation:**
   - Test estimation involves estimating the effort and resources required for the testing effort.
   - This includes estimating the number of test cases, the time required for test preparation, execution, and reporting, and the resources required for testing.

4. **Test Approach:**
   - The test approach outlines how the testing will be performed and the test types and levels that will be used. Th

5. **Entry Criteria:**
   - Entry criteria define the conditions that must be met before testing can begin.

6. **Exit Criteria:**
   - Exit criteria define the conditions that must be met before testing can be considered complete.

***(Asked 2 times)***

## 9. What do you mean by test plans? What are the things to keep in mind while planning a test?

## OR

## Test Planning activities.

- A test plan is a document that outlines the approach, scope, objectives, resources, and schedule for testing a software application.
- It was created during the planning phase of the software development life cycle.

Test planning activities involve the following steps:

1. **Understand the requirements**:
- Before starting to plan the test, it is important to understand the requirements of the software application being tested.
- Determine the purpose of testing, including the functional and non-functional requirements, user expectations, and any relevant standards or regulations.

2. **Define the scope:**
- Determine the scope of the testing effort, including what will be tested, the level of testing required, and any specific testing scenarios or use cases.

3. **Determine the test approach:**
- Choose the appropriate testing approach, such as manual testing, automated testing, or a combination of both.

4. **Identify the testing environment:**
   - Determine the hardware, software, and network requirements for testing, and ensure that the necessary resources are available.

5. **Create test cases:**
   - Develop test cases that cover all aspects of the software application being tested, including positive and negative test scenarios, boundary tests, and error conditions.

6. **Assign roles and responsibilities:**
   - Identify the testing team and assign roles and responsibilities to ensure that all aspects of the testing effort are covered.

7. **Define test metrics:**
   - Determine the metrics that will be used to measure the effectiveness of the testing effort, such as defect density, test coverage, and test execution time.

8. **Develop a test schedule:**
   - Create a schedule that outlines the testing activities, including the start and end dates for each phase of testing.

9. **Determine the test deliverables:**
   - Identify the test deliverables, such as test reports, test cases, and test scripts, and ensure that they are delivered on time and meet the desired quality standards.

**(Asked 2 times)**

## 10. Explain Entry Criteria in detail.

- Entry criteria refers to the set of conditions that must be met before testing can begin.
- They ensure that the testing environment is ready and that the testing team has the necessary information to begin testing.

Entry criteria includes the following:

1. **Completion of development:**
- The software application must be fully developed before testing can begin.

2. **Availability of test resources:**
- The necessary hardware, software, and network resources must be available and configured for testing.
- This includes test tools, test environments, and test data.

3. **Test environment setup:**
- The testing environment must be set up and configured, including the installation and configuration of test tools.

4. **Test data availability:**
- Sufficient test data must be available for testing, including both positive and negative test scenarios.

5. **Test Plan Approval:**
- The test plan must be approved by all stakeholders, including the development team, project managers, and business owners.

6. **Test Team Availability:**
- The test team must be available to start testing.
- This includes testers and other team members who will be responsible for executing the test plan.

***(Asked 5 times)***

## 11. Explain Exit Criteria in detail.

- Exit criteria refers to the conditions that must be met before testing can be considered complete.
- These criteria help the testing team ensure that the software application has been thoroughly tested and is ready for release.
- This helps ensure that the software application is of high quality and meets the needs and expectations of the users and stakeholders.

Exit criteria includes:

1. **Test Coverage:**

- The testing team must ensure that they have tested all the functionality, features, and requirements specified in the test plan.
- This includes both positive and negative test scenarios.

2. **Resolving identified defects:**

- All identified defects must be addressed and closed.
- This includes verifying that the defects have been fixed and retesting to ensure that they have been resolved.

3. **Stability:**

- The software application must be stable and free of critical defects that could impact the user experience or cause data loss.

4. **Performance:**

- The performance of the software application must meet the requirements specified in the test plan.
- This includes both functional and non-functional requirements.

5. **User Acceptance:**
   - The software application must be accepted by the users, stakeholders, and business owners.
   - This includes ensuring that the software meets the business requirements and is ready for release.

6. **Regression Testing:**
   - Regression testing must be performed to ensure that any changes made to the software application have not introduced new defects or issues.

***(Asked 3 times)***

## Q. Differentiate between entry criteria and exit criteria in testing.

Entry criteria ensure that testing can begin, and exit criteria ensure that testing is complete, and the software application is ready for release.

Explain Que no 10 and 11.

## Q. Considering the login system with username (email address only) and password, write 5 entry and exit criteria case.

Here are five entry and exit criteria cases for a login system with username and password:

**Entry Criteria:**

1. The testing team has access to the testing environment, which includes the login system, testing tools, and necessary hardware and software.
2. The user database should be set up with at least one user account.
3. The test data has been prepared and is ready for use.
4. The system should have a valid SSL certificate installed and enabled.
5. The testing team has access to all necessary documentation, such as the software requirements specification and design documents.

**Exit Criteria:**

1. The user should be able to successfully login with a valid email address and password combination.
2. The user should not be able to login with an invalid email address or password.
3. The system should properly handle any errors or exceptions that occur during the login process.
4. The login process should be performed securely, with no vulnerabilities or security issues.
5. All necessary tests, such as functional, security, and usability tests, should be completed and passed before the system is released to production.

## 12. Explain test strategy. How do you know which strategies (among preventive and reactive) to pick for the best chance of success?

Test strategy is a high-level document that outlines the approach to be used for testing an application or software system.

It defines the testing objectives, methods, scope, environment, and tools to be used in the testing process.

**Preventive Test Strategy:**

- The preventive test strategy focuses on preventing defects from occurring in the first place.
- This approach involves identifying potential defects early in the development process and taking steps to prevent them from causing problems later on.
- The preventive strategy includes activities such as requirement reviews, design reviews, code inspections, and static analysis.
- The objective of the preventive strategy is to identify and address defects before they can impact the quality of the software.

**Reactive Test Strategy:**

- The reactive test strategy, on the other hand, focuses on identifying and fixing defects after they have been introduced into the software.
- This approach involves testing the software at various stages of development to identify defects and then fixing them before the software is released.
- The reactive strategy includes activities such as functional testing, regression testing, and performance testing.
- The objective of the reactive strategy is to find and fix defects that have not been caught by the preventive strategy.

**(Asked 3 times)**

## 13. Explain Test Progress monitoring with example.

- Test progress monitoring is the process of tracking and evaluating the progress of testing activities during the testing phase of a software development project.
- Its goal is to ensure that testing is progressing according to plan, identify any issues or risks, and make necessary adjustments to meet project goals and timelines.

Test progress monitoring is important for several reasons:

1. **Ensuring Testing is on Track:**
- Test progress monitoring helps ensure that testing is progressing as planned and that testing activities are being completed on schedule.
2. **Early Identification of Risks:**
- Test progress monitoring helps identify risks and issues early in the testing process.
3. **Improving Test Coverage:**
- Test progress monitoring helps identify gaps in test coverage, allowing the testing team to focus on areas of the software that require additional testing.
4. **Improving Quality:**
- Test progress monitoring helps ensure that defects are identified and resolved early in the testing process, improving the quality of the software and reducing the cost of fixing defects later in the project.
5. **Providing Visibility to Stakeholders:**
- Test progress reports provide project stakeholders with visibility into the testing process, helping them make decisions about the project.

**Example**: Suppose a software testing project has been planned for four weeks, and the team has completed testing for two weeks. The test manager would monitor the progress of testing activities to ensure that the team is on track to complete testing within the remaining two weeks. If there are any delays or issues, the test manager will take corrective actions to get the project back on track.

## 14. Explain Test Reporting

- Test reporting involves communicating the status and results of testing activities to stakeholders.
- It includes creating test reports that provide information on the testing progress, results, and any issues or concerns.
- Test reports help stakeholders make informed decisions about the software's readiness for release.
- **Example**: A test report may include information on the number of test cases executed and the number of defects found.
- It may also include information on the areas of the software that have been tested and the areas that still need to be tested.

**(Asked 3 times)**

## 15. Explain Test Controlling in detail.

- Test controlling is the process of ensuring that the testing activities are executed as per the plan.
- Test control involves monitoring the actual testing activities and comparing them to the planned activities.
- Test control activities ensure that the testing process remains in control and any deviation is identified, analyzed and corrected.
- Test control helps in identifying any risks or issues that may impact the testing process and finding ways to mitigate them.
- Test control also helps in maintaining the quality of the testing process and ensuring that the testing activities meet the desired quality standards.
- Test control requires close collaboration with stakeholders to keep them informed about the status of the testing activities and seek their inputs for any necessary changes.
- Test control requires continuous review and updating of the testing plan to ensure that it remains relevant and effective.

For **example**, during the testing of a mobile application, the test control process may involve regularly monitoring the progress of test execution, analyzing the test results, identifying any issues or defects, and making necessary corrections or changes to the testing plan to ensure that the testing process remains on track and meets the desired quality standards.

## Q. Do you think test control is required for proper management of testing.

- Yes, test control is an essential aspect of proper management of testing.
- Test control enables the test manager to monitor the progress of the testing, identify areas of improvement, and take corrective actions to ensure that the testing objectives are met.
- Without proper test control, the testing process can become unmanageable, leading to delays, cost overruns, and poor-quality products.

**(Asked 2 times)**

## 16. Explain Configuration Management

- Configuration management is the process of tracking and controlling changes to software and hardware systems throughout their development and deployment lifecycle.
- The goal of configuration management is to ensure that the software and hardware artifacts used in testing are accurate, up-to-date, and complete.
- Configuration management can include version control, change control, and release management processes.
- Examples of software artifacts that may be managed through configuration management include source code, test cases, test plans, test data, and documentation.
- Configuration management can help prevent errors and inconsistencies by ensuring that everyone involved in testing is using the same version of the software and hardware artifacts.
- Configuration management also helps in tracing the source of the problem if there is any issue.

**(Asked 3 times)**

## 17. Define Risk. What do you mean by Product Risk.

## OR

## Explain about project risk.

- Risk is the possibility of something going wrong that can have a negative impact on a project or product.
- It can be anything that may potentially harm the product, such as missed deadlines, budget overruns, low-quality deliverables, or changes in requirements.

- Product risks are the risks associated with the quality, reliability, and performance of the software product being developed or tested.
- These risks can be related to the design, implementation, or testing of the product.
- Product risks can arise due to factors such as complex functionality, new technologies, or unfamiliar platforms.
- A product risk can cause delays in the project timeline, increase the project cost, or result in a low-quality product that fails to meet customer expectations.

For **example**, if a software product is being developed using a new technology that the development team is not familiar with, there is a risk that the product may not perform as expected or may contain defects that are difficult to identify and fix.

This can impact on the project timeline and budget and may result in dissatisfied customers.

- It is important for a product team to identify and assess risks early in the project/product life cycle, as it allows them to develop a risk management plan to minimize the impact of potential risks.

***(Asked 2 times)***

## 18. What do you mean by risk in software testing. Explain simple words and points with examples.

- In software testing, risk refers to the possibility of something going wrong that could negatively impact the success of the project or product.
- This could be anything that may cause harm or result in a loss to the project, its stakeholders or customers.

Examples of risks in software testing include:

1. **Technical Risks:** These include risks related to hardware or software failure, compatibility issues, or inadequate testing tools.
2. **Resource Risks**: These include risks related to insufficient staffing or resources, or lack of access to necessary equipment or software.
3. **Schedule Risks:** These include risks related to missed deadlines or unrealistic project timelines.
4. **Business Risks**: These include risks related to the impact of the project on the business, such as legal or financial risks.
5. **Data security risk:** Data breaches and vulnerabilities can lead to loss of data, damage to the company's reputation, and legal and financial repercussions.

- In order to manage and mitigate risks in software testing, it is important to identify and assess them early in the project and implement strategies to address them.
- This includes developing a risk management plan, identifying and prioritizing risks, and regularly monitoring and reviewing the risk management process.

**(Asked 2 times)**

## 19. In what ways does a risky project impact the organization?

OR

Describe a risk as a possible problem that would threaten the achievement of one or more stakeholder's project objectives.

A risky project can have various impacts on the organization, such as:

1. **Delays and Increased Costs:**
   - Risky projects often encounter unexpected problems that lead to delays and additional costs.
   - This can impact the organization's bottom line and revenue streams.

2. **Reputation:**
   - If a project fails due to risks, it can damage the organization's reputation and credibility in the industry.
   - This can lead to loss of business and potential clients.

3. **Stakeholder Dissatisfaction:**
   - A risky project may not meet the expectations of stakeholders.
   - This can lead to dissatisfaction and mistrust, which can impact the organization's future projects.

4. **Employee Morale:**
   - High-risk projects can put a lot of pressure on the project team, leading to stress and anxiety.
   - This can result in lower employee morale and productivity.

**(Asked 3 times)**

## 20. What do you know about risk management? Explain about the activities in risk management.

OR

Risk Management in testing.

OR

Describe Risk management. How do you avoid a project from being a total failure.

- Risk management is a process to minimize, monitor, and control the probability and impact of potential negative events.
- It involves identifying, assessing, and prioritizing risks.
- It involves developing and implementing strategies to mitigate or avoid these risks.
- The goal of risk management in testing is to ensure successful completion of testing within a given timeframe and budget.

The activities included in risk management are:

1. **Risk Identification**: This involves identifying potential risks and documenting them.
2. **Risk Assessment:** This involves evaluating the likelihood and impact of identified risks.
3. **Risk Mitigation**: This involves developing strategies to minimize or eliminate the identified risks.
4. **Risk Monitoring**: This involves tracking the identified risks and assessing the effectiveness of the mitigation strategies.
5. **Risk Reporting**: This involves communicating the identified risks, their potential impact, and the mitigation strategies to the project stakeholders.

***(Asked 4 times)***

## 21. Explain incident management.

## OR

## What is an incident in software development. What is the proper way to handle incident mechanism.

- An incident refers to any event or occurrence that disrupts the software testing process.
- Examples of incidents in software testing can include a website crashing, a software feature not working as expected, or a database corruption.
- Incident management helps to ensure the quality and reliability of the software being tested.
- The goal of incident management is to minimize the impact of the incident on the software testing process and to prevent it from recurring in the future.
- The process involves the following steps:

1. **Incident Identification**:
- Identifying and recording the incident as soon as it occurs.
- This can be done through automated monitoring tools or manually by the testers.

2. **Incident Categorization and Prioritization**:
- Categorizing the incident based on its impact, severity, and priority.
- This helps in prioritizing the incidents and allocating resources accordingly.

3. **Investigation**:
- Investigating the incident to understand the root cause and determine the appropriate course of action to resolve it.

4. **Incident Resolution**:
- Taking the necessary steps to resolve the incident, which can include fixing the defect, updating the software, or implementing a workaround.

5. **Incident Closure**:
- Closing the incident after it has been resolved and verifying that it has been fully resolved.

6. **Incident Reporting**:
- Reporting the incident and its resolution to the relevant stakeholders, including the testing team, development team, and project manager.

***(Asked 2 times)***

## Q. What steps should be taken to increase the chance of success of any project?

The steps are:

1. Define clear objectives.
2. Develop a solid project plan.
3. Build a capable team.
4. Assign roles and responsibilities.
5. Set up a communication plan.
6. Monitor progress.
7. Manage risks.
8. Conduct testing.
9. Document everything.
10. Review and evaluate.

# Chapter 8
# Tool Support for Testing
# (1 SQ – 5 marks)

***(Asked 2 times)***

## 1. What different kinds of testing tools can we use in an organization?

## OR

## What are the different types of test tools necessary for test process activities?

- There are many different types of testing tools that can be used in an organization.
- It depends on the type of testing being performed and the specific needs of the organization.
- Here are some common categories of testing tools:

1. **Test management tools:**

- These tools help manage the testing process, including creating and organizing test cases, tracking test results, and generating reports.
- Examples include JIRA.

2. **Test automation tools:**

- These tools are used to automate the execution of tests, reducing the amount of manual effort required.
- Examples include Selenium, and Cucumber.

3. **Performance testing tools:**
- These tools are used to simulate high user loads and measure the performance of applications under different conditions.
- Examples include JMeter.

4. **Security testing tools:**
- These tools are used to identify vulnerabilities in applications and ensure that they are secure against potential attacks.
- Examples include Bug V.

5. **API testing tools:**
- These tools are used to test the functionality and performance of APIs.
- Examples include Postman, and Swagger.

6. **Mobile testing tools:**
- These tools are used to test the functionality and performance of mobile applications on different devices and platforms.
- Examples include Appium and Espresso.

## Q. Is compiler a testing tool? Write your view.

- No, a compiler is not considered as a testing tool.
- A compiler is a software program that translates source code into executable code and is typically used in the software development process as part of the build process.
- Compilers do not provide any functionality specifically designed for testing software.

***(Asked 2 times)***

## 2. What are the benefits of using testing tools in the software development process?

- Software testing is a time consuming and tedious activity.
- Many time you might need to do repetitive tests.
- Tools help to automate repetitive tasks and make testing easier.
- Using testing tools in the software development process can provide several benefits, including:

1. **Increased efficiency:**
- Testing tools can automate repetitive tasks, such as running regression tests, allowing testers to focus on more complex tasks.

2. **Improved accuracy:**
- Testing tools can help identify defects and vulnerabilities that may be missed by manual testing, ensuring that the software is of higher quality.

3. **Faster time-to-market:**
- By automating testing tasks, testing tools can help accelerate the testing process, allowing the software to be released to market faster.

4. **Cost savings:**
- By reducing the amount of manual effort required for testing, testing tools can help save costs associated with testing.

5. **Improved collaboration:**
- Testing tools can help facilitate communication and collaboration between developers and testers.

Overall, using testing tools can help ensure that software is of higher quality, is released faster, and is more cost-effective to develop and maintain.

## Q. Would you prefer to embrace testing tools or go against the use? Give your opinion.

I would prefer the use of tools. Explain Que no 2.

**(Asked 2 times)**

## 3. What are the potential risks of test automation and tool support for testing.

## OR

## What are the challenges of introducing automated tools in an organization?

- Potential risks of test automation and tool support for testing include:

1. **Impact on project success:**

- The tools used in a project can have a significant impact on its success.
- The wrong tool can lead to delays, errors, and other issues that can negatively affect the project.

2. **Initial investment:**

- Implementing automated testing and acquiring testing tools can require a significant initial investment, which may not be feasible for all organizations.

3. **Compatibility:**

- The tools used in a project need to be compatible with the existing infrastructure and systems used in the organization.
- If they are not, it can cause integration issues that can be difficult and time-consuming to resolve.

4. **Underestimation of the learning curve:**
- New tools may require a significant amount of training and learning on the part of the users.
- This can lead to delays in the implementation and increased costs for training.

5. **Resistance to change:**
- Employees may resist the introduction of a new tool, which can lead to further delays and costs in implementing the tool.

6. **False sense of security:**
- Automated testing can provide a false sense of security if tests are not designed or executed correctly, leading to potential defects being missed.

7. **Limited test coverage:**
- Automated testing may not be able to cover all aspects of the software being tested, potentially leading to defects being missed.

8. **Technical challenges:**
- Implementing and maintaining automated testing and testing tools may require technical expertise that may not be available within the organization.

## Q. Why does a project manager need to think carefully about the tools that are to be used in the organization?

Because it by lead to various challenges.

Explain Que no 3.

## Q. “People often make mistakes by underestimating the time, cost and effort for the initial introduction of a tool.” Validate this statement.

- This statement is valid as often people underestimate the time, cost, and effort required for introducing a new tool in the organization.
- It is important to carefully consider the time, cost, and effort required before introducing a new tool into an organization.

- Challenges occur while introducing a tool to an organization.

- Explain Que no 3

## 4. What things should be considered for "Tool supporting for execution and logging"?

- When considering tool support for execution and logging in software testing, there are several important things to consider by the organizations, including:

1. **Test environment:**
- The tools selected for execution and logging should be compatible with the test environment, including the hardware and software configurations, as well as any other tools being used for testing.

2. **Test data management:**
- The tools should support effective management of test data, including the ability to generate test data, import/export data, and store and retrieve data during testing.

3. **Reporting and analysis:**
- The tools should provide effective reporting and analysis capabilities, including the ability to generate detailed reports on test results, track progress, and identify areas for improvement.

4. **Integration and automation:**
- The tools should support integration with other testing tools and processes and should provide automation capabilities to reduce the manual effort required for execution and logging.

5. **Security:**
- The tools should be designed with security in mind, including the ability to encrypt data, manage access controls, and prevent unauthorized access or modification of test data or results.

6. **Usability:**
- The tools should be user-friendly and easy to use, with clear interfaces and intuitive navigation, to ensure that testers can focus on testing rather than struggling with complex or confusing tool interfaces.

***(Asked 3 times)***

## 5. "Introducing a new testing tool to a company may bring chaos. "What should be considered by the management before introducing such tools to an organization? Support your answer taking scenario of any local level company.

Before introducing a new testing tool to a company, management should consider the following:

1. **Need**:
- Evaluate the need for the tool and ensure it aligns with the company's overall goals and objectives.
2. **Compatibility**:
- Assess the compatibility of the tool with the existing testing environment and systems used in the organization.
3. **Cost and resources:**
- Consider the cost and resources required for implementation, training, and customization.
4. **Benefits and risks:**
- Evaluate the potential benefits and risks of the tool and determine if the benefits outweigh the risks.
5. **Skill level:**
- Consider the skill level of the testing team and determine if additional training will be required.

For **example**, let's consider a local level company that develops mobile applications.

The management team wants to introduce a new automated testing tool to improve their testing process.

Before making a decision, the management should consider the above points.

By carefully evaluating these factors, the management team can introduce the new tool without causing chaos in the organization.

## Q. What are the factors to determine to choose a testing tool?

Ans: Que no 5.

## Q. Assume yourself as a Test Leader, in your opinion, what should be considered before introducing a tool into your enterprise? What are the things need to care for in order to produce a quality product?

Ans: Que no 5.

## Q. Explain what happens if one has over reliance on a tool?

- Over-reliance on a tool can have negative consequences for an organization. Here are some of the potential issues:

1. Decreased critical thinking of tester.
2. Reduced creativity opportunities for new ideas or approaches.
3. Tools can have blind spots or limitations.
4. Increased risk of errors.
5. Limited perspective of tester.

# Chapter 9
# Testing Web and Mobile Application
# (1 Que – 7 marks)

## 1. Why is it necessary to test web applications?

It is necessary to test web applications for several reasons, including:

1. **Ensure Quality:**
   - Testing ensures that the web application meets the quality standards, requirements, and specifications.
   - It helps to identify and fix issues, and bugs that could affect the application's performance, usability, and security.

2. **Improve User Experience:**
   - Testing helps to ensure that the web application provides a user-friendly experience and meets the user's expectations.

3. **Reduce Costs:**
   - Testing helps to identify and fix issues early in the development cycle, reducing the cost of fixing issues later.

4. **Ensure Compatibility:**
   - Testing ensures that the web application works correctly across different devices, browsers, and operating systems.

5. **Improve Security:**
   - Testing helps to identify and fix security vulnerabilities, ensuring that user data is safe and secure.

**(Asked 7 times)**

## 2. What are the testing techniques for web application testing?

OR

In any web application testing, what sort of techniques should be undertaken for qualitative output?

OR

In order to be safe from various threats, how can any web application be tested?

There are several testing techniques for web application testing, including:

1. **Functional Testing:**
   - This technique tests the functionality of the web application, such as whether the buttons, links, forms, and other features of the app work as expected.
   - It involves verifying if the web application meets the requirements and specifications.

2. **Integration Testing:**
   - This technique tests the web application's integration with other systems, services, or third-party applications.
   - It involves testing how the app interacts with other systems, such as payment gateways, social media, or APIs, and ensures that the integration works smoothly.

3. **Usability Testing:**
   - This technique tests how user-friendly and easy to use the web application is.
   - It involves testing the ease of navigation, readability, accessibility, and overall user experience.

4. **Compatibility Testing:**
   - This technique tests whether the web application works correctly across different browsers, devices, and operating systems.
   - It involves verifying if the app works correctly on different configurations and ensuring that the app behaves consistently on all devices.

5. **Performance Testing:**
   - This technique tests how the web application performs under different loads and scenarios.
   - It involves testing how the app behaves under heavy traffic, high load, and other performance-related issues.

6. **Security Testing:**
   - This technique tests the web application's security features to ensure that user data is safe and secure.
   - It involves testing the app for vulnerabilities such as SQL injection, cross-site scripting, and other security threats.

7. **Regression Testing:**
   - This technique tests the web application after changes or updates to ensure that new updates or features do not cause any issues or bugs.
   - It involves testing the app's functionality, usability, compatibility, performance, and security after making changes to the app.

8. **Accessibility Testing:**
   This technique tests the web application's accessibility to people with disabilities, such as vision or hearing impairment.

## 3. What are the challenges while performing web application testing?

There are several challenges that testers face when performing web application testing. Some of the common challenges include:

1. **Browser Compatibility:**
   - Web applications must run on various browsers such as Chrome, Firefox, and Edge, and each browser may interpret the web page differently.
   - This means testers must check the application's compatibility with different browsers and versions, which can be a time-consuming task.

2. **Device Compatibility:**
   - Web applications must also work on different devices such as desktops, laptops, tablets, and smartphones, which have different screen resolutions, processing power, and hardware specifications.
   - Ensuring compatibility across multiple devices can be a challenge, particularly when testing responsive designs.

3. **Performance:**
   - Web applications need to be fast, responsive, and reliable to provide a good user experience.
   - Testers need to ensure that the application can handle high traffic volumes, and that it performs well on different network conditions such as slow internet connections.

4. **Security:**
   - Web applications often deal with sensitive user data such as passwords, payment information, and personal information.
   - Testers need to check for security vulnerabilities, such as SQL injections, cross-site scripting (XSS), and authentication flaws.

5. **Usability:**
- Usability testing is an essential aspect of web application testing.
- Testers need to ensure that the application is user-friendly, easy to navigate, and intuitive.

6. **Integration:**
- Web applications often integrate with other systems, such as third-party APIs and databases.
- Testing these integrations can be a complex task, particularly when dealing with multiple systems and data formats.

7. **Accessibility:**
- Web applications must be accessible to users with disabilities.
- Testers need to ensure that the application meets accessibility standards and that it can be used by users with visual, auditory, or physical impairments.

**(Not asked)**

## 4. Explain various mobile app testing techniques.

- There are several mobile app testing techniques that testers can use to ensure the quality of mobile applications.
- Here are some of the commonly used techniques:

1. **Functional Testing:**

- This technique focuses on testing the application's functionality, ensuring that all features and functionalities are working as expected.

2. **Performance Testing:**

- This technique is used to test the application's performance, including factors such as response time, memory usage, CPU usage, and battery consumption.
- Testers use various tools to measure the performance of the application under different conditions, such as high traffic volumes and low network connectivity.

3. **Usability Testing:**

- This technique focuses on testing the application's usability, including factors such as ease of use, navigation, and user satisfaction.
- Testers use different methods, such as surveys, user feedback, and heuristic evaluations, to assess the application's usability.

4. **Security Testing:**

- This technique is used to test the application's security, including factors such as authentication, data encryption, and protection against common security threats such as malware, phishing, and hacking.

5. **Compatibility Testing:**
- This technique focuses on testing the application's compatibility with different devices, operating systems, and screen sizes.

9. **Accessibility Testing:**
- This technique tests the mobile application's accessibility to people with disabilities, such as vision or hearing impairment.

***(Asked 5 times)***

## 5. Explain different types of challenges while testing mobile applications.

## OR

## Why is it difficult to test any mobile based application?

Here are some of the common types of challenges that testers face while testing mobile applications:

1. **Device Diversity:**
- Mobile devices come in various screen sizes, resolutions, operating system, and hardware configuration.
- Testing mobile applications on all possible devices is not possible, so testers need to prioritize which devices to test on based on factors such as popularity, screen size, and operating system version.

2. **User Experience:**
- Mobile apps are all about delivering an excellent user experience, which can be challenging to test.
- Testers need to consider factors such as ease of use, responsiveness, speed, and accessibility when testing mobile apps.

3. **Network Connectivity:**
- Mobile applications often rely on network connectivity to function properly.
- Testers need to test the app in different network conditions such as 2G, 3G, 4G, and Wi-Fi to ensure that the app works correctly.

4. **Security:**
- Mobile devices store a vast amount of personal and sensitive data, making them a target for hackers and cybercriminals.
- Testers need to test the app for vulnerabilities such as data breaches, malware, and hacking which is hard.

5. **Performance:**
- Mobile apps need to be fast and responsive, and testers need to test the app's performance in different scenarios such as low battery, high CPU usage, and low memory.

6. **Compatibility:**
- Mobile apps need to work correctly with other apps on the device, such as the camera, GPS, and other sensors.
- Testers need to ensure that the app works well with other apps and does not cause conflicts.

8. **Accessibility:**
- Mobile applications must be accessible to users with disabilities.
- Testers need to ensure that the application meets accessibility standards and that it can be used by users with visual, auditory, or physical impairments.

## 6. Explain about mobile application testing with emulator versus testing with real device.

Mobile application testing can be done either on an emulator or a real device.

Both approaches have their advantages and disadvantages.

- **Testing with an emulator** involves simulating the behavior of a mobile device on a computer.
- Emulators provide a cost-effective way to test mobile applications without needing to own several devices, which can be expensive.
- Emulators can also help to speed up the testing process by allowing testers to automate testing and run tests in parallel.

**Limitations:**

- Emulators cannot replicate the exact behavior of a real device, especially when it comes to hardware-specific features such as sensors, camera, and battery life.
- Testing on emulators also does not take into account the network conditions, such as slow or intermittent connections.

- On the other hand**, testing with a real device** involves testing the application on an actual mobile device.
- Testing on real devices provides a more accurate and realistic representation of the application's behavior, especially when it comes to hardware-specific features.
- Testing on real devices can also help to uncover issues related to network conditions and user experience that may not be detected on an emulator.
- However, testing on real devices can be time-consuming and expensive, as testers may need to acquire multiple devices to test on different platforms and versions.

Both approaches are necessary for thorough mobile application testing, and testers should use a combination of both methods to ensure comprehensive testing coverage.

***(Asked 2 times)***

## 7. Differentiate between web app testing and mobile app testing.

| | **Web Application Testing** | **Mobile Application Testing** |
|---|---|---|
| Platform | Runs on web browsers. | Runs on mobile devices. |
| Accessibility | Can be accessed from anywhere with an internet connection. | Requires a mobile device to access. |
| Testing Challenges | Compatibility issues with different browsers and devices. | Compatibility issues with different devices and platforms. |
| Performance Testing | Focuses on page loading times, server response times, and other web-specific metrics. | Focuses on CPU usage, memory consumption, and battery usage of the mobile device. |
| User Experience Testing | Focuses on UI/UX design, ease of use, and user journey. | Focuses on touch and gesture-based interactions, screen resolution, and responsiveness. |
| Hardware Testing | Does not require hardware-specific testing. | Requires testing of hardware-specific features, such as sensors, camera, and battery life. |
| App Store Approval | Not subject to the approval process as mobile apps. | Must pass through an approval process before being listed on app stores. |

# Chapter 10
# Quality management
# (1 SQ – 5 marks)

**(Asked 3 times)**

## 1. Explain Six Sigma. (6 σ)

- Six Sigma is a quality management methodology that can be used as a software quality measurement technique.
- It is a data-driven approach that aims to identify and eliminate defects in processes, including software development processes.
- In Six Sigma, the goal is to reduce the number of defects or errors in a process to a level that is statistically significant, with a target of no more than 3.4 defects per million opportunities.
- This involves using statistical tools and techniques to analyze data and identify areas for improvement.
- Six sigma methodology can be explained as DMAIC framework.

  (Define, Measure, Analyze, Improve, Control)

1. **Define**

- In the Define phase, the problem or opportunity for improvement is clearly defined, and the goals and objectives of the project are established.
- The team identifies the customer's needs and expectations and sets a goal to reduce the defect.

2. **Measure**

- In the Measure phase, data is collected to measure the current performance of the system being improved.
- This includes identifying key performance indicators (KPIs) and establishing a baseline for the process.

3. **Analyze**
- In the Analyze phase, the data collected in the Measure phase is analyzed to identify the root causes of the problems or issues identified in the Define phase.
- This includes using statistical tools and techniques to identify trends, patterns, and relationships in the data.

4. **Improve**
- In the Improve phase, solutions are developed and tested to address the root causes identified in the Analyze phase.
- This may involve testing and refining different solutions until the most effective solution is identified.

5. **Control**
- In the Control phase, the improved process or system is implemented and monitored to ensure that it remains stable and meets the desired performance goals.
- This includes establishing a monitoring and control plan to sustain the improvements made in the project.

***In short***

1. ***Define****: Define the problem or opportunity, goals, and customer requirements.*
2. ***Measure****: Collect data on the process and establish a baseline.*
3. ***Analyze****: Analyze the data to identify the root causes of defects or issues.*
4. ***Improve****: Develop and implement solutions to address the root causes.*
5. ***Control****: Establish control systems and procedures to sustain the improvements.*

**(Asked 2 times)**

## 2. Explain Software Quality Standards.

- Software quality standards are a set of guidelines that define the requirements and expectations for software quality.
- These standards help ensure that software products are developed in a consistent and reliable manner and meet the needs of their users.
- Standards cover various aspects of the software development process, including requirements gathering, design, coding, testing, and maintenance.
- Specific areas such as security, usability, performance, and compatibility may also be addressed.
- Widely recognized software quality standards include ISO 9001.
- Standards provide a common framework for software development and testing practices and help organizations to demonstrate their commitment to quality.
- Following software quality standards can improve the overall quality of software products, reduce the risk of defects and failures, and increase customer satisfaction.
- Implementing these standards can also enhance an organization's reputation and competitiveness in the marketplace.
- There are various software standards and some of them are:

1. **ISO (International Organization for Standardization)**

- ISO is an independent, non-governmental international organization that develops and publishes standards for various industries, including software.
- ISO 9001 and ISO 12207 are examples of software quality standards developed by ISO.
- ISO 9001 provides a framework for quality management, while ISO 12207 provides a framework for software life cycle processes.

2. **CMMI (Capability Maturity Model Integration)**

- It stands for Capability Maturity Model Integration.
- It is a framework that helps organizations improve their processes for developing software and other products.
- It provides a roadmap for organizations to achieve greater efficiency, consistency, and quality.
- It focuses on process improvement in key areas such as project management, requirements, design, development, and testing.

**(Asked 3 times)**

## 3. Explain ISO.

- ISO stands for International Organization for Standardization.
- It is a non-governmental organization that develops and publishes standards for various industries, including software.
- ISO standards are voluntary, but many organizations choose to adopt them in order to demonstrate their commitment to quality and improve their processes.
- ISO standards are developed by a group of experts from around the world through a collaborative process and are regularly reviewed and updated to ensure they remain relevant and effective.
- ISO standards can offer several benefits to organizations, including:
    1. Improved quality: help organizations improve their products, services, and processes.
    2. Increased customer satisfaction
    3. Enhanced reputation of the company
    4. Competitive advantage of the company in market
    5. Reduced costs of finding bugs
    6. Increased efficiency
    7. Improved risk management:
- **ISO 9001** and **ISO 12207** are examples of software quality standards developed by ISO.
- ISO 9001 provides a framework for quality management, while ISO 12207 provides a framework for software life cycle processes.

**(Asked 5 times)**

## 4. Explain CMMI.

- It stands for Capability Maturity Model Integration.
- It is a framework that helps organizations improve their processes for developing software and other products.
- It provides a roadmap for organizations to achieve greater efficiency, consistency, and quality.
- It helps organizations identify areas for improvement in their processes.
- It enhances the quality of the organization's software products.
- It focuses on process improvement in key areas such as project management, requirements, design, development, and testing.

- It has several maturity levels, with each level building on the previous one to provide a more advanced framework for process improvement.
- It is based on best practices and is designed to be flexible to meet the unique needs of different organizations.
- It is widely used in the software industry to enhance product quality.
- It can be applied to different industries beyond software.
- The CMMI Institute maintains the CMMI framework.
- The CMMI Institute provides training, certification, and consulting services to help organizations implement the CMMI framework.
- Overall, CMMI is a structured approach to improve processes, achieve better results, and enhance product quality.

- One of the organizations that have implemented CMMI is IBM.
- IBM has adopted CMMI as a standard for process improvement across its software development teams.