Software design and testing

- Software design
  - Is the process of defining software functions, objects, methods and structure of the programming codes, so that it matches the desired expectations
- Software testing is the next step wherein the software is put through trials to ensure that it matches expected requirements and is error free.

Software design

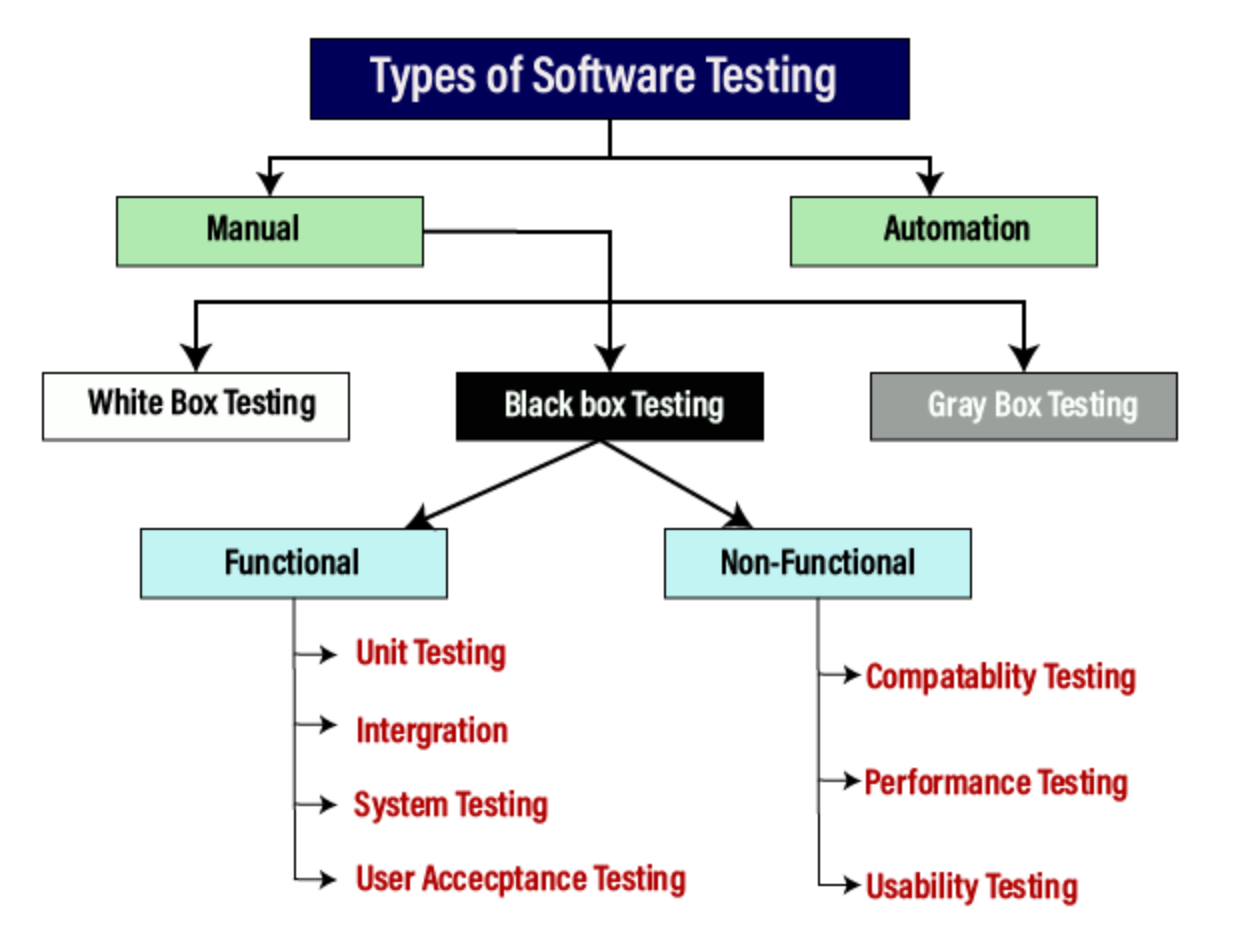
the design phase deals with transforming the customer requirements into a form implementable using a programming language. It is divided into 3 phases:

- Interface design
  - is the specification of the interaction between a system and its environment
- Architectural design
  - Is the specification of the major components of a system.
- Detailed design
  - is the specification of the internal elements of all major system components, their properties, relationships and processing. It decompose the system components into units

Software Testing (ST)

- is a methods to check whether the actual software product matches expected requirements and ensure that it is error free.
- testing is sometimes call it white box and black box testing.
- Simply testing means the verification of application under test (AUT)

Types of ST


Types of Software Testing
Manual
Automation
White Box Testing
Black box Testing
Gray Box Testing
Functional
Non-Functional
Unit Testing
Intergration
System Testing
User Accecptance Testing
Compatablity Testing
Performance Testing
Usability Testing

Remark for system testing

- **Alpha-testing**: is the system testing performed by the development team
- **Beta-testing**: is the system testing performed by a friendly set of customers
- **Acceptance testing**: after the software has been delivered, the customer performed the acceptance testing to determine whether to accept the delivered software or to reject it

Remark

- The process of software testing aims not only at finding faults in the existing software but also at finding measures to improve the software in terms of efficiency, accuracy, and usability.

Types of ST

- Functional testing
  - Unit testing
  - Integration testing
  - User acceptance testing: a type of testing to evaluate the application against real-life scenarios
- Non-functional testing
  - Performance
  - Usability
  - scalability
- Maintenance testing

Types of ST

- **Functional Testing:** a type of software testing to verify whether the application delivers the expected output.
- **Non-functional Testing:** a type of software testing to verify whether the non-functional aspects of an application (e.g., stability, security, and usability) are working as expected.

Types of ST

- Functional testing
- **Unit testing**: a type of testing done on an individual unit in an application
- **Integration testing**: a type of testing done on groups of application units to see how they work together
- **Acceptance testing**: a type of testing to evaluate the application against real-life scenarios

Non-functional Testing,

- **Security Testing**: Testing that checks if the software is secure and protects against unauthorized access or threats.
- **Performance Testing**: Testing that assesses how well the software performs in terms of speed, stability, and resource usage.
- **Load Testing**: A type of performance testing that evaluates how the software handles expected and peak loads.
- **Usability Testing**: Testing that measures how user-friendly and easy-to-use the software is.

When software Bugs (errors/fault/failure/problem) occurs?

- The software does not do something that the specification says it should do
- The software does something that the specification says it should not do
- The software is difficult to understand, hard to use or slow
- Note: Most bugs are not because of mistakes in the code BUT may be in terms of Specification (55%), design (25%), code (15%), others (5%)

Remark

- **Mistake** – a human action that produces an incorrect result.
- **Fault [or Defect]** – an incorrect step, process, or data definition in a program.
- **Failure** – the inability of a system or component to perform its required function within the specified performance requirement.
- **Error** – the difference between a computed, observed, or measured value or condition and the true, specified, or theoretically correct value or condition.
- **Specification** – a document that specifies in a complete, precise, verifiable manner, the requirements, design, behavior, or other characteristic of a system or component, and often the procedures for determining whether these provisions have been satisfied.

ST Process (Cont.)

- **Software testing can be divided into two steps:**
- **Verification:** it refers to the set of tasks that ensure that software correctly implements a specific function.
- 2. **Validation:** it refers to a different set of tasks that ensure that the software that has been built is traceable to customer requirements.
- **Verification:** “Are we building the product right?”
  Validation: “Are we building the right product?”

Verification Example

Verification: Are we building the product right?
Through verification, we make sure the product behaves the way we want it to. For example, on the left in Figure 1, there was a problem because the specification said that players should collect $200 if they land on or pass Go. Apparently a programmer implemented this requirement as if the player had to pass Go to collect. A test case in which the player landed on Go revealed this error.

Verification
Are we building the product **right**?

"I landed on "Go" but didn't get my $200!"

Validation
Are we building the **right** product?

"I know this game has money and players and "Go" – but this is not the game I wanted."

Figure 1: Verification vs. Validation

Validation Example

Validation: Are we building the right product?

Through validation, we check to make sure that somewhere in the process a mistake hasn't been made such that the product build is not what the customer asked for; validation always involves comparison against requirements. For example, on the right in Figure 1, the customer specified requirements for the Monopoly game – but the programmer delivered the game of Life. Maybe the programmer thought he or she "knew better" than the customer that the game of Life was more fun than Monopoly and wanted to "delight" the customer with something more fun than the specifications stated. This example may seem exaggerated – but as programmers we can miss the mark by that much if we don't listen well enough or don't pay attention to details – or if we second guess what the customer says and think we know better how to solve the customer's problems.

Verification
Are we building the product **right**?

"I landed on "Go" but didn't get my $200!"

Validation
Are we building the **right** product?

"I know this game has money and players and "Go" – but this is not the game I wanted."

Figure 1: Verification vs. Validation

What are the different types of software testing (Approaches To Software Testing)?

- **Manual Testing:** Testing software manually by humans without any automation tool or script
- (i.e. The process of checking the functionality of an application as per the customer needs without taking any help of automation tools is known as manual testing)
- **Automation Testing:** Testing software using tools or scripts that automatically interact with the software. The human tester only needs to execute the script and let it do the rest of the testing
- (i.e., Automation testing is a process of converting any manual test cases into the test scripts with the help of automation tools, or any programming language is known as automation testing).
- With the help of automation testing, we can enhance the speed of our test execution because here, we do not require any human efforts. We need to write a test script and execute those scripts.

Manual Testing

- **Manual Testing:** Manual testing includes testing software manually, i.e., without using any automated tool or any script. In this type, the tester takes over the role of an end-user and tests the software to identify any unexpected behavior or bug.
- There are different strategies (stages) for manual testing such as
  - unit testing,
  - integration testing,
  - system testing, and
  - user acceptance testing.

Automation Testing (Cont.)

- **Automation Testing:** Automation testing, which is also known as Test Automation, is when the tester writes scripts and uses another software to test the product. This process involves the automation of a manual process.
- Automation Testing is used to re-run the test scenarios that were performed manually, quickly, and repeatedly.

Manual Testing Strategies

- Unit testing: this testing is followed by the programmer to test the unit of the program.
- Integration testing
  - It focuses on the construction and design of the software.
- System testing
  - Your software is compiled as a whole and then tested as a whole
- Acceptance testing: in this level a system is tested for acceptability

What are different levels of software testing?

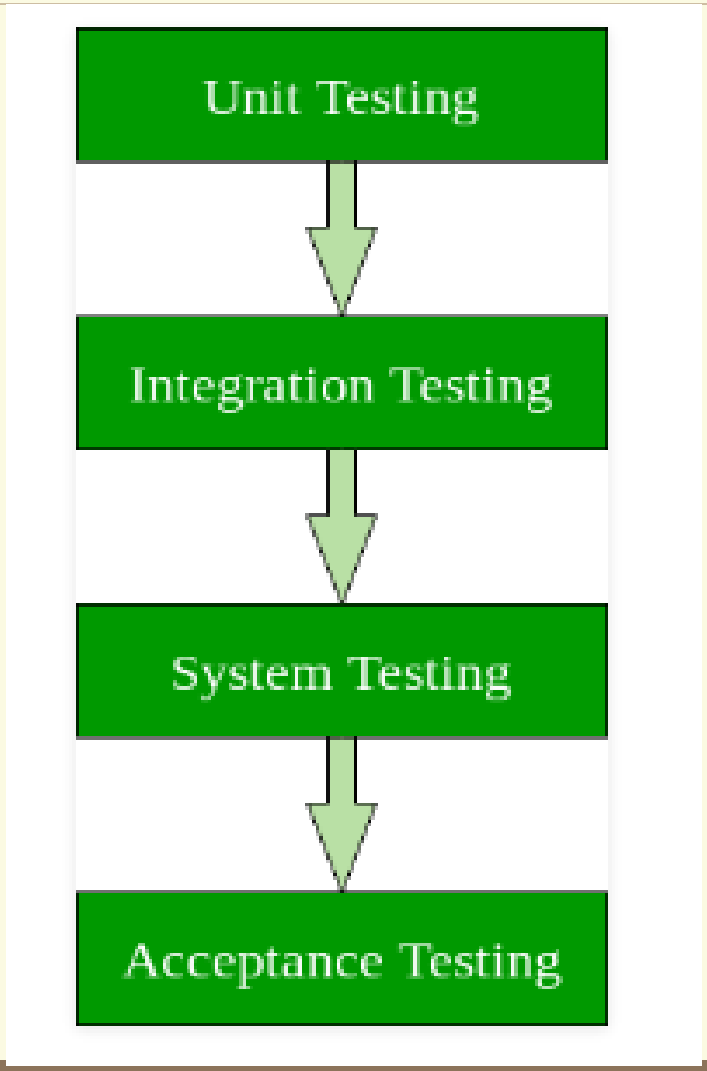
1. **Unit Testing:** A level of the software testing process where individual units/components of a software/system are tested. The purpose is to validate that each unit of the software performs as designed.

2. **Integration Testing:** A level of the software testing process where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units.

3. **System Testing:** A level of the software testing process where a complete, integrated system/software is tested. The purpose of this test is to evaluate the system's compliance with the specified requirements.

4. **Acceptance Testing:** A level of the software testing process where a system is tested for acceptability. The purpose of this test is to evaluate the system's compliance with the business requirements and assess whether it is acceptable for delivery.

Cont.


Unit Testing
Integration Testing
System Testing
Acceptance Testing

Manual vs Automation Testing

Definition	Manual testing is an approach in which a human tests software by manually using it like a user would.	Automated testing is an approach in which software testers use automated testing tools/scripts to speed up the process.
Benefits	• Easily capture bugs that sometimes automated scripts cannot detect • Lower cost • High flexibility	• Fast and effective • Great long-term ROI • Transparent results for the entire QA team
Drawbacks	• Prone to human error • Time-consuming and tedious • Not everything can be tested manually	• Require upfront investment • Machines cannot provide insight into how the app's UX or design appeal to the human user • May lack flexibility
Types of Testing to use	• Exploratory testing • Usability testing • Ad-hoc testing	• Regression testing • Load & Performance testing • Unit testing & Integrated testing
When to use	• During the very first rounds of testing to have a general understanding of the application • When testing infrequent, specific, and ad-hoc scenarios • When assessing the UI/UX of the application	• When continuous feedback on application quality is required • When there are features that have to be tested repeatedly after each release cycle

Cont.

Acceptance testing is conducted by business owners, the purpose of acceptance testing is to test whether the system does in fact, meet their business requirements

Regression Testing is the testing of software after changes has been made; this testing is done to make sure that the reliability of each software release, testing after changes has been made to ensure that changes did not introduce any new errors into the system

Alpha Testing Usually in the existence of the developer at the developer's site will be done.

Beta Testing Done at the customer's site with no developer in site.

Functional Testing is done for a finished application; this testing is to verify that it provides all of the behaviors required of it

Alpha and Beta testing

- Alpha testing is an in-house testing process performed within the organization by the developer or tester
- Beta testing is deployed on the customer site for getting tested by the intended users or clients in the real envirnoment.

What are the different techniques of Software Testing?

- **Black Box Testing (specification-based testing):** is a ST technique where the functionality of the system is tested without comprehending or looking into the internal structure

white box testing or structural testing

- **White-Box Testing:** The technique of testing in which the tester is aware of the internal workings of the product, has access to its source code, and is conducted by making sure that all internal operations are performed according to the specifications is known as white box testing or structural testing

White Box Testing Tools

Below is a list of top white box testing tools.

- Parasoft Jtest
- EclEmma
- NUnit
- PyUnit
- HTMLUnit
- CppUnit

Example: NUnit is a unit-testing framework for all .Net languages.

Cont.

Black Box Testing	White Box Testing
Internal workings of an application are not required.	Knowledge of the internal workings is a must.
Also known as closed box/data-driven testing.	Also known as clear box/structural testing.
End users, testers, and developers.	Normally done by testers and developers.
This can only be done by a trial and error method.	Data domains and internal boundaries can be better tested.

Gray Box Testing

- **Grey Box Testing** or Gray box testing is a software testing technique to test a software product or application with partial knowledge of internal structure of the application.
- The purpose of grey box testing is to search and identify the defects due to improper code structure or improper use of applications.

Gray Box Testing (Cont.)

- Gray Box Testing is a software testing method, which is a combination of both White Box Testing and Black Box Testing method.

- In White Box testing internal structure (code) is known
- In Black Box testing internal structure (code) is unknown
- In Grey Box Testing internal structure (code) is partially known

Techniques used for Grey box Testing are

Techniques used for Grey box Testing are-

- **Matrix Testing:** This testing technique involves defining all the variables that exist in their programs.
- **Regression Testing**: To check whether the change in the previous version has regressed other aspects of the program in the new version. It will be done by testing strategies like retest all, retest risky use cases, retest within a firewall.
- **Orthogonal Array Testing or OAT**: It provides maximum code coverage with minimum test cases.
- **Pattern Testing:** This testing is performed on the historical data of the previous system defects. Unlike black box testing, gray box testing digs within the code and determines why the failure happened

Steps to perform Grey box Testing are

- **Step 1**: Identify inputs
- **Step 2**: Identify the outputs
- **Step 3**: Identify the major paths
- **Step 4**: Identify Subfunctions
- **Step 5**: Develop inputs for Subfunctions
- **Step 6**: Develop outputs for Subfunctions
- **Step 7**: Execute test case for Subfunctions
- **Step 8**: Verify the correct result for Subfunctions
- **Step 9**: Repeat steps 4 & 8 for other Subfunctions
- **Step 10**: Repeat steps 7 & 8 for other Subfunctions

Validation: Are we building the right product?

Through validation, we check to make sure that somewhere in the process a mistake hasn't been made such that the product build is not what the customer asked for; validation always involves comparison against requirements. For example, on the right in Figure 1, the customer specified requirements for the Monopoly game – but the programmer delivered the game of Life. Maybe the programmer thought he or she

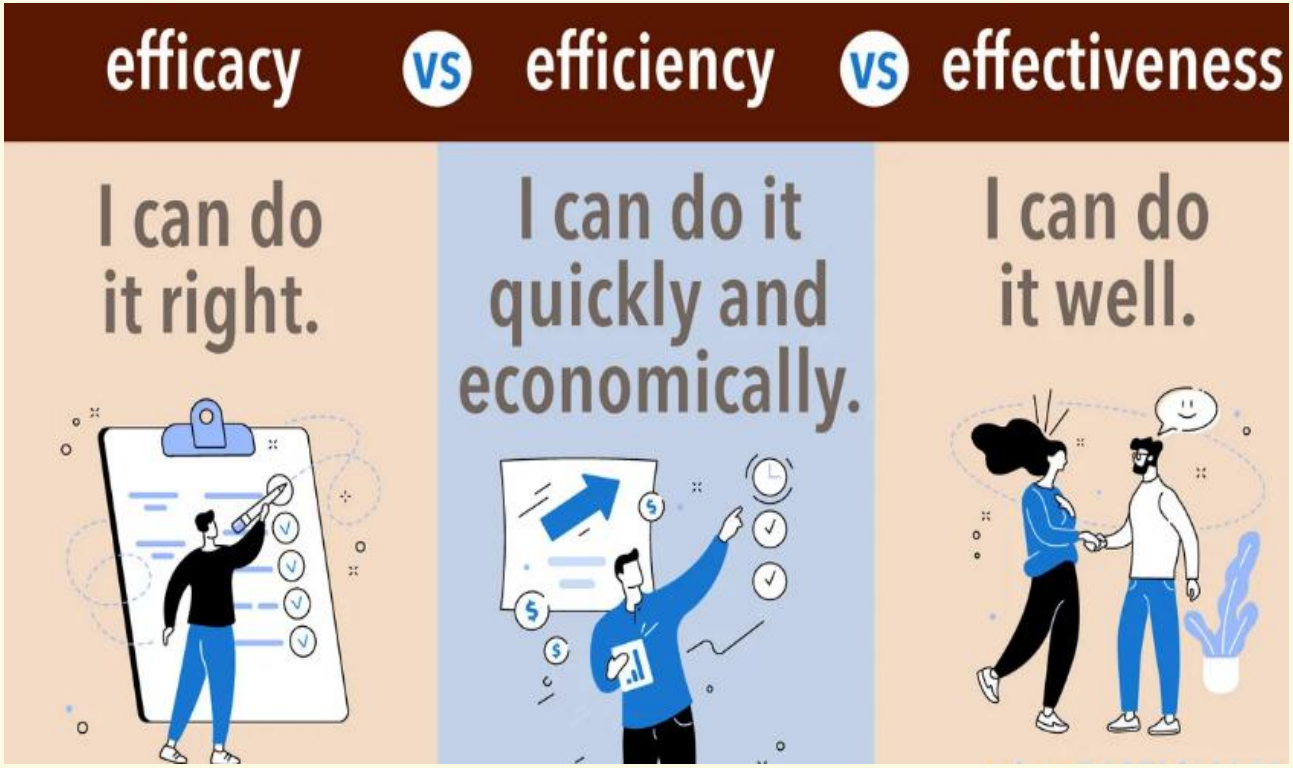
Question??

- What is the difference between: effectiveness, efficiency, or **Efficacy**?

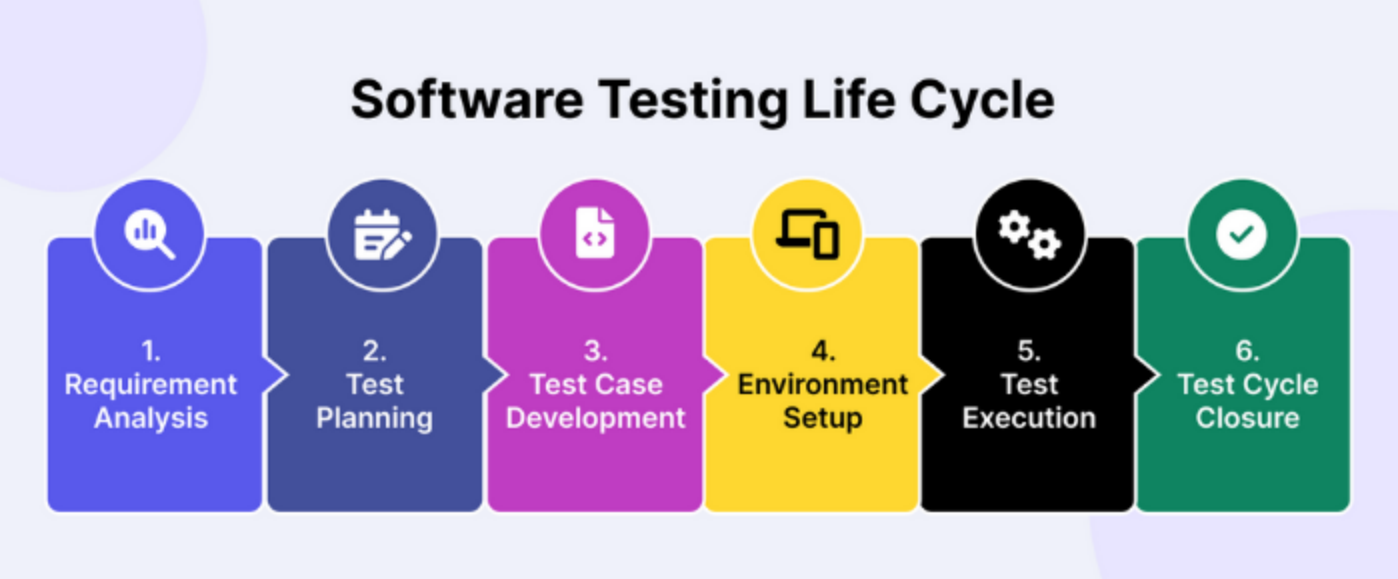
Efficacy vs. Efficiency vs. Effectiveness

Word	Example
efficacy	If a shampoo cleans your hair, it's **efficacious**.
efficiency	If a shampoo cleans your hair in only one minute, it's **efficient**.
effectiveness	If a shampoo cleans your hair really well, it's **effective**.

Cont.


efficacy
VS
efficiency
VS
effectiveness
I can do it right.
I can do it quickly and economically.
I can do it well.

ST lifecycle



Requirement Analysis

- In this stage, software testers work with stakeholders involved in the development process to identify and understand test requirements.

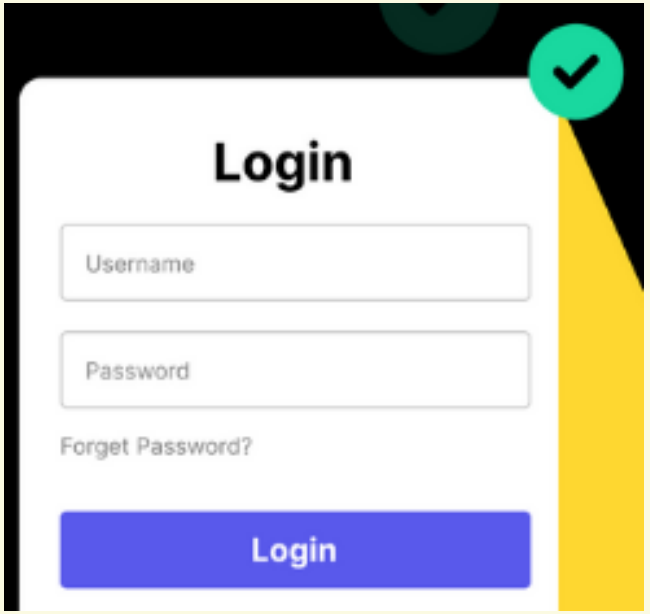
Test Planning

After thorough analysis, a test plan is created. Test planning involves aligning with relevant stakeholders on the test strategy:

- **Test objectives**: Define attributes like functionality, usability, security, performance, and compatibility.
- **Output and deliverables**: Document the test scenarios, test cases, and test data to be produced and monitored.
- **Test scope**: Determine which areas and functionalities of the application will be tested (in-scope) and which ones won't (out-of-scope).
- **Resources**: Estimate the costs for test engineers, manual/automated testing tools, environments, and test data.
- **Timeline**: Establish expected milestones for test-specific activities along with development and deployment.
- **Test approach**: Assess the testing techniques (white box/black box testing), test levels (unit, integration, and end-to-end testing), and test types (regression, sanity testing) to be used.

Test Case Development

- After defining the scenarios and functionalities to be tested, we'll write the test cases.
- Login page Example:



Login Page Test Cases (An Example)

- Testing if the Login page functions as expected under valid inputs. These test cases explore scenarios where users do what they are supposed to do, such as:
- Valid username and password combination successfully logs the user in.
- Testing with the minimum allowed username and password length.

Cont.

- Testing with a username and password containing alphanumeric characters.
- Successful login with the "Remember Me" option selected
- Testing login with a username that contains both uppercase and lowercase characters.
- Successful login using a valid email address as the username.
- Successful login using a valid phone number as the username.
- Successful login with multi-factor authentication (MFA) enabled.

Cont.

- Testing login with a username that includes special characters (e.g., @, #, $).
- Successful login using social media accounts (if applicable).
- Successful login using biometric authentication (e.g., fingerprint, face recognition).
- Testing login after a password reset to ensure the new password works.
- Successful login after an account recovery process.
- Successful login with localization settings (testing with different languages).
- Testing login with different browsers (e.g., Chrome, Firefox).

Test Environment Setup

- This step can be done in parallel with Test Case Development. A test environment is the software and hardware configurations under which the application is tested, including a database server, front-end running environment, browser, network, hardware, etc.

Test Execution

- With clear objectives in mind, the QA team writes test cases, test scripts, and prepares necessary test data for execution.
- Tests can be executed manually or automatically. Manual testing is suitable when human insights and judgment are required, while automation testing is preferable for repetitive flows with minor adjustments. Once the tests are executed, any defects found are tracked and reported to the development team, who promptly resolve them.

Test Cycle Closure

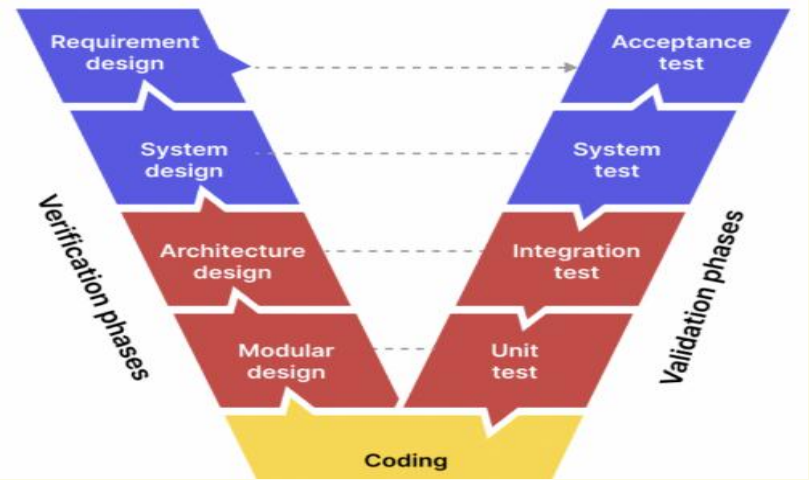
- This is the final phase of Software Testing. Software testers will gather to analyze what they found from the tests, evaluate the effectiveness, and document key takeaways for future reference.

Popular Software Testing Models

- V-model
- Test Pyramid model
- The Honeycomb Model
- نموذج قرص العسل

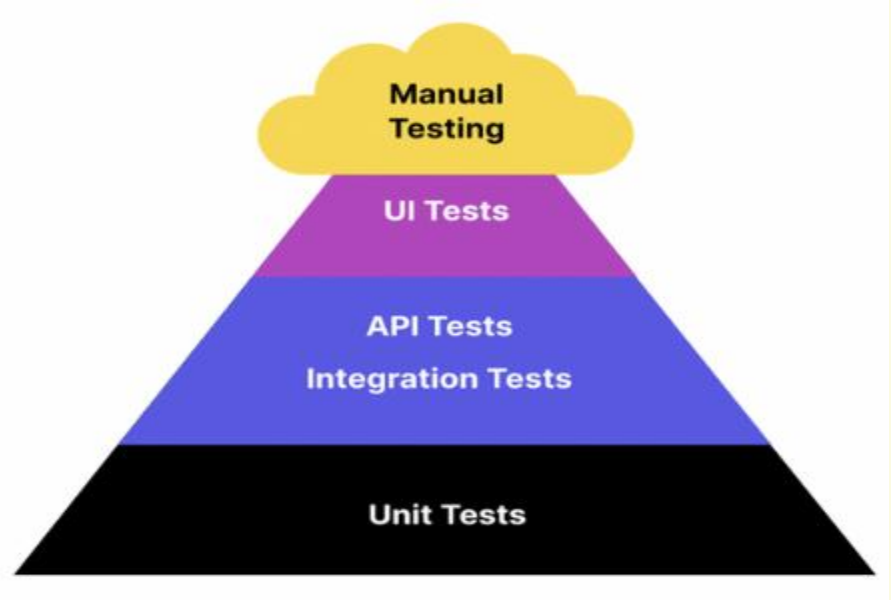
V-model

- In the past, QA teams had to wait until the final development stage to start testing. Test quality was usually poor, and developers could not troubleshoot in time for product release. The V-model solves that problem by engaging testers in every phase of development. Each development phase is assigned a corresponding testing phase. This model works well with the waterfall process model



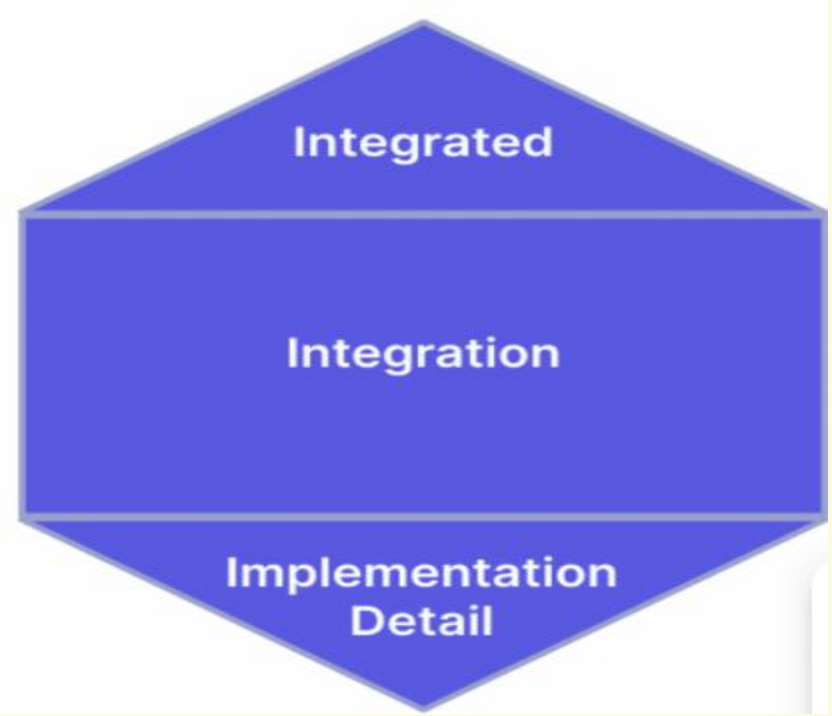
Test Pyramid model

- As technology advances, the Waterfall model gradually gives way to the widely used Agile testing methods. Consequently, the V-model also evolved to the Test Pyramid model, which visually represents a 3-part testing strategy.



The Honeycomb Model

The Honeycomb model is a modern approach to software testing in which Integration testing is a primary focus, while Unit Testing (Implementation Details) and UI Testing (Integrated) receive less attention. This software testing model reflects an API-focused system architecture as organizations move towards cloud infrastructure.



Question?

- **Is Software Quality Assurance (SQA) and software testing same?**

Answer

- No, Software Quality Assurance (SQA) and software testing are not the same, although they are related concepts within the field of software development.
- SQA is a broader and more encompassing concept that involves ensuring the quality of the entire software development process. SQA هو مفهوم أوسع وأكثر شمولاً يتضمن ضمان جودة عملية تطوير البرمجيات بأكملها
- Software testing, on the other hand, is a specific subset of SQA. It is the process of systematically evaluating a software application to identify and rectify defects or issues in the software.
- Together, these processes help ensure the delivery of high-quality software products that meet user expectations and industry standards.

Software Testing and quality assurance (QA)

- Software Testers focus on finding application bugs, conflicts, and errors. Software testers work to detect and fix errors, whereas quality assurance assists in preventing any mistakes or issues during the software development process.
- **Quality assurance** considers every step of the development process, but software testing includes activities in the program after the codes are written.

Software Testing vs quality assurance (QA)

- يقدم ضمان الجودة وثائق العملية والمعايير والمبادئ التوجيهية وتوصيات تحسين العملية. فهو يضمن أن عملية التطوير نفسها قوية وفعالة. يركز الاختبار بشكل أكبر على الأهداف قصيرة المدى، مثل تحديد وإصلاح العيوب في إصدار البرنامج الحالي.

Differences Between Software Testing and Quality Assurance

Criteria	Software Testing	Quality Assurance
Objective	The primary objective of software testing is to identify defects or bugs in the software. Testing is a dynamic process where testers actively execute the software to find issues such as functionality errors, performance problems, security vulnerabilities, and usability issues	Quality assurance is a broader process that focuses on ensuring that the entire software development process is carried out correctly. It aims to prevent defects from occurring in the first place by implementing best practices, processes, and standards
Timing	Testing is typically carried out after the software has been developed. It is a validation process that confirms whether the software meets the specified requirements.	Quality assurance starts at the beginning of the software development process and continues throughout the entire lifecycle. It encompasses activities like process audits, standards enforcement, and training to improve the development process itself.

Cont.

Responsibility	Testers are responsible for conducting testing activities. They execute test cases, report defects, and ensure that the software functions as expected.	Quality assurance is the responsibility of the entire development team and often involves quality assurance professionals, process experts, and project managers. It requires a collaborative effort to improve processes and maintain quality standards.
Deliverables	The primary deliverables of testing are test plans, test cases, test reports, and defect logs. These artifacts provide information about the software's quality and reliability.	Quality assurance delivers process documentation, standards, guidelines, and process improvement recommendations. It ensures that the development process itself is robust and efficient.
Long-term vs. Short-term	Testing is more focused on short-term goals, such as identifying and fixing defects in the current software release.	Quality assurance has a long-term focus, aiming to improve processes and prevent defects in future software projects.

Usability Testing (UT)

- Usability means: The software should be easy to use & safe
- One method to assess the effectiveness and level of satisfaction of the developed Software among the intended audience is usability testing (UT)
- UT demonstrates how effectively the application functions in terms of effectiveness, enjoyment, ease of use, usefulness, satisfaction, improvements, and attitudes towards future usage from the perspectives of the users and other relevant stakeholders.
- Testing for usability is a method for determining how well liked and efficient the software is among users and other relevant stakeholders. (UT serves as a technique used to assess Software's effectiveness and user and stakeholder satisfaction)

Usability Testing (UT)

- UT demonstrates how effectively the software functions in terms of effectiveness, enjoyment, ease of use, usefulness, satisfaction, improvements, and attitudes towards future usage from the perspectives of the users, and other relevant stakeholders.
- How? Using system usability scale (SUS)

Usability Metrics

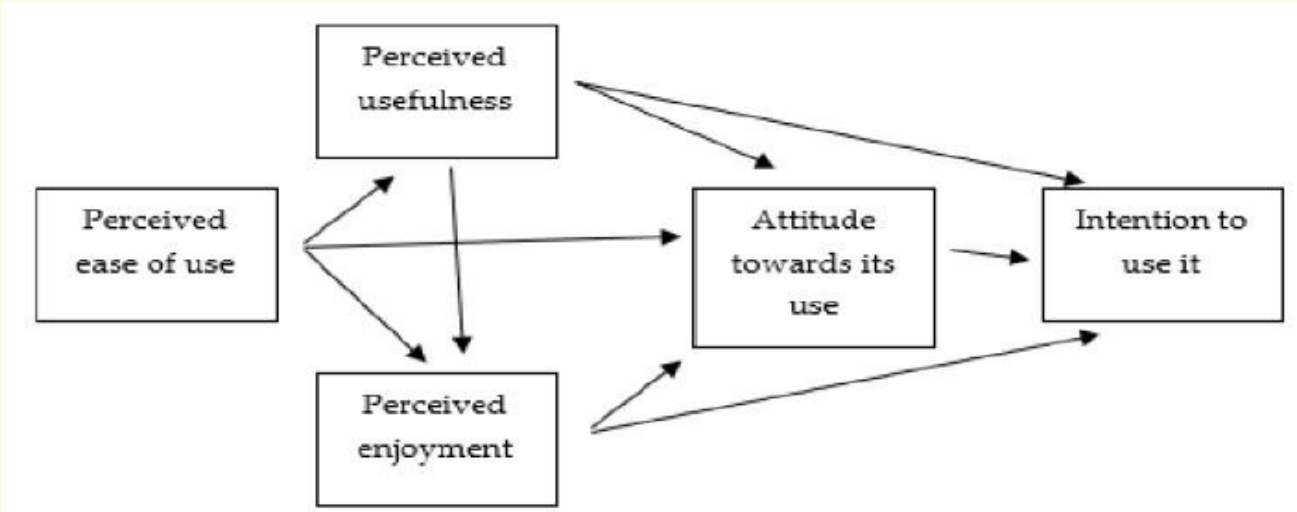
- System usability scale (SUS). SUS contains a set questionnaire used to gather information from users and relevant stakeholders.
- The SUS form contains ten questions on ease of learning, efficiency, ease of memorization, occurrence of execution errors, and level of satisfaction. Every question incorporates a five-point scale varying from one (totally disagree) to five (totally agree).

Fun Toolkit

- One of the funtollkit is the Smileyometer rating scale which is a visual analogue scale that is easy to complete by circling one face for each question without having to write anything down

Technology Acceptance Model (TAM)

- TAM is the most extensively studied model of user acceptance that looks at why a user chooses to use or not use technology. Perceived usefulness (PU), perceived ease of use (PEU), perceived enjoyment (PE), attitude towards its use (AU), and intention to use (IU) are the five components that make up TAM (see below)



Cont.

- PEU and PU are both self-efficacy viewpoint variables in the TAM model.
- The concept of perceived utility (PU) refers to how much people believe a technology could be useful.
- PEU gauges how user-friendly people believe the technology to be.

PU & PEU Examples

Example Questionnaires for PU and PEU Component Assessments

Question#	PU sample questionnaire	PEU sample questionnaire
1	Do serious games have the potential to be used by more people?	Do serious games offer an easy-to-interact interface to learners with ASC?
2	Is it possible to use serious games as an aid to learner's learning?	Was the game rules and learning mechanics process simple and interest?
3	Is it feasible to use serious games as a strategy to reduce children' social isolation and offer them novel experiences that can improve their cognitive abilities?	Do serious games require a lot of effort to be used?

Example

In order to assess people's attitudes towards certain technologies, we use the number of questions with agreements (AgrP) and disagreements (DisP) to determine the degree of agreement of each proposition, referred to as the DAP. Based on the DAP results, an individual's level of agreement or disagreement will be done. For example, the following terms are used in the computation of DAP: strongly agree (SA), partially agree (PA), neutral (N), partially disagree (PD), and totally disagree (TD). The number of disagreements (DisP) and agreements (AgrP) in the questions is used to compute DAP using the above-mentioned terms, as follows:

$$AgrP = SA + PA + \left(\frac{N}{2}\right), \qquad DispP = TD + PD + \left(\frac{N}{2}\right) \tag{1}$$

$$DAP = 100 * \left(\frac{AgrP}{AgrP + DisP}\right) \tag{2}$$

Case Study Example for a Sample of **Five** users

DAP Values Interpretations and Survey Results

(a) Interpretations of DAP Values

DAP value	Proper phrase
90 or more	*Extreme agreement*
80 to + 89,99	*Substantial agreement*
70 to + 79,99	*Moderate agreement*
60 to + 69,99	*Low agreement*
50 to + 59,99	*Negligible agreement*
40 to + 49,99	*Negligible disagreement*
30 to + 39,99	*Low disagreement*
20 to + 29,99	*Moderate disagreement*
10 to + 19.99	*Substantial disagreement*
9.99 or less	*Extreme disagreement*

(b) Survey Results

Perceived Ease of Use (PEU)								
Question#	TD	PD	N	PA	SA	DisP	AgrP	DAP
1	0	0	0	2	3	0	5	100
2	0	0	0	3	2	0	5	100
3	3	0	2	0	0	4	1	20
Perceived Usefulness (PU)								
1	0	0	0	2	3	0	5	100
2	0	0	0	1	4	0	5	100
3	0	0	0	1	4	0	5	100

$$AgrP = SA + PA + \left(\frac{N}{2}\right), \qquad DispP = TD + PD + \left(\frac{N}{2}\right) \tag{1}$$

$$DAP = 100 * \left(\frac{AgrP}{AgrP + DisP}\right) \tag{2}$$