In software testing a QA team can distinguish 2 primary categories:

* functional testing and
* non-functional testing.

Both aim to put assurance into software quality, but  
each approach comes from different angles.

### Functional Testing

* examines the software’s behavior and functionality.
* verifies whether the app functions as expected and meets the requirements.
* main question: “Does the app perform the intended tasks and produce the expected outputs?”
* Team creates test cases that simulate real-world scenarios to validate the expected behavior/functionality against predefined requirements/specs.

**Common techniques**

**Unit Testing**

* Testing individual components or modules to verify their correctness.
  + correct path
  + correct control structure
  + correct loops

**Smoke Testing**

* A preliminary round of testing to verify basic functionality before conducting more comprehensive tests.

- performed post build to verify that the critical functionalities are working fine

- executed before any detailed functionality, regression test are executed

- main purpose is to reject the app with issues so that Wa does not waste time testing broken software app.

- test cases chose to cover the most important functionality or component of the system

- the objective is not to perform exhaustive testing, but to verify the critical functionality.

**Sanity Testing**

- performed after receiving build, with minor changes in the code or functionality, to make sure that the bigs have been fixed and no further issues are introduced due the changes

- goal is to determine that the proposed functionality works roughly as expected.

- checking the right version to make sure the right build was tested

**Regression Testing**

* Testing previously tested functionalities to ensure that changes or enhancements have not introduced any new issues.
* confirm program or code change has not adversely affected existing features ( broken features)
* full or partial selection of already executed test cases that are re-executed to ensure existing functionalities work fine.
* done to ensure that the new code - the code changes do not have a side affect on this existing functionalities
* ensures old code still works once the latest code changes are done(?)
* automation written for regression

**System Integration Testing (SIT)**

* Testing the interaction between different components/modules to ensure they work together seamlessly.

- to verify the different modules or services used work well together

* example
  + interaction with the DB
  + microservices work together

- more expensive to run as they required multiple parts of the app to be up and running

**User Acceptance Testing**

- performed by the end user or the client to verify/accept before moving the app goes to prod

- done in the final phase of testing

- main purpose of UAT is to validate end to end business flow. it does not focus on cosmetic erros, spelling mistakes or system testing

- testing is carried out in a separate testing env with prod like data setup

- UAT is performed by clients/end user.

**API Testing**

- validates APIs

- to check the functionality, reliability, performance and security of the interface

- instead of using standard user inputs (keyboard) and outputs, you use software to send call to the API, get out output, and note down the system's response.

- business logic layer

communications and data exchange between two separate software system

- database and server should be configured as per the application requirements

* once the installation is done, API function should be called to check wether that API is working.

**Localization Testing**

- process of testing a globalized app whose UI

- behavior ist tested for a specific region, locale or culture.

- customizing as per the targeted language and country: default language, currency, date, time format, documentation

- content and UI

- purpose is the check appropriate linguistic and culture aspects...

- change the user interface for even the intiatl settings according to requirements

- verify typographical error, sulture appropriateness of UI, linguistics errors;

**User Interface (UI) Testing**

* Testing the graphical user interface to ensure it is user-friendly and operates as intended.

### Non-Functional Testing

* evaluates how the app performs/behaves under different conditions/environments.
* QA team assess the app’s
  + responsiveness,
  + resource utilization,
  + security vulnerabilities,
  + user-friendliness, and its
  + ability to handle high loads or concurrent users.
* Goal is to identify potential bottlenecks, risks, and usability issues.
* the relevant quality attributes therefore are
  + performance,
  + reliability,
  + usability,
  + speed,
  + scalability,
  + security,
  + stability, and
  + user experience.

**Common techniques**

**Performance Testing**

* Evaluating the software’s speed, scalability, and responsiveness under various workloads and conditions.

**Load Testing**

* Testing the software’s performance under expected and peak load conditions.

**Stress Testing**

* Assessing the software’s stability and performance beyond its normal capacity, under extreme conditions.

**Usability Testing**

* Testing the software’s user-friendliness, ease of use, and overall user experience.

**Security Testing**

* Testing the software’s vulnerability to unauthorized access, data breaches, and other security threats.

**Usability Testing**

* Evaluating the software’s user-friendliness and intuitiveness by observing real users interacting with it.

**Accessibility Testing**

* Verifying the software’s compatibility with different operating systems, browsers, hardware, and network configurations.

**Endurance Testing**

* Evaluating the software’s ability to perform consistently and reliably over a specified period and under various conditions.

**Browser Testing**

* Testing the software’s availability and accessibility to users during different times and under different scenarios.

**Mobile Testing**

* Assessing the software’s ability to handle increasing workloads and scale up with growing user demands.

**Penetration Testing**

* Testing the software’s ease of maintenance, code readability, and modularity.

**Conclusion**:

While functional testing verifies that the app behaves as intended and meets the functional requirements, non-functional testing evaluates its performance, usability, security, and other quality attributes. Both types combined in one comprehensive testing strategy support the software development teams to deliver high-quality software which allows user centric results.