**Types of Software Testing**

**Unit Testing**

* Unit testing is the most basic level of software testing, but it plays a very important role in ensuring code quality.
* In unit testing, developers test small pieces of code, like functions, methods, or classes, individually and in isolation.
* The idea is simple: before combining everything into a big system, make sure that each building block works perfectly on its own. For example, if a program has a function to calculate the sum of two numbers, unit testing will check if it consistently returns the correct output for various inputs.
* Developers usually write unit tests alongside their code using frameworks like JUnit for Java or pytest for Python.
* Doing unit testing early saves a lot of trouble later. Catching a bug in a small piece of code is far easier and cheaper than finding it after the whole system is built. In modern software development, writing unit tests is often considered a best practice and is even encouraged before writing the actual functionality, a method known as Test-Driven Development (TDD).
* Overall, unit testing lays the foundation for a stable and reliable application. The QA team can also do it manually but now we have automated scripts in Maven.

**API Testing**

* In today’s world where applications often communicate with each other through APIs (Application Programming Interfaces), API testing has become critical.
* API testing ensures that these connections are reliable, secure, and work exactly as intended.
* Postman is one of the most popular tools for API testing because of how simple yet powerful it is. Using Postman, testers can send different types of HTTP requests like GET, POST, PUT, and DELETE to the API and verify if the responses are correct.
* For instance, if there is an API that adds a product to a shopping cart, a tester would use Postman to send a POST request with product details and then check if the server responds with a success message. API testing does not rely on a user interface, making it faster and easier to find backend bugs that could otherwise go unnoticed.
* GET is used to request data from a server without changing anything.The data is sent through the URL and can be seen in the address bar.
* POST is used to send data to a server to create or update something. The data is sent in the request body, keeping it hidden from the URL.
* There are other methods as well, like PUT, PATCH ,DELETE etc,they all have their appropriate uses in Postman Testing.
* By validating APIs early, developers can ensure that different parts of the application communicate properly and consistently. Postman also allows writing automated tests and organizing them in collections, which makes it easier for ongoing development and testing cycles.

**Integration Testing**

* While unit testing focuses on checking individual components, integration testing ensures that these components work properly when combined together. Even if each part of a system passes its own unit tests, problems can still appear when they interact. Integration testing plays a crucial role in catching such issues early, preventing bigger problems later during full system testing.
* Key aspects of integration testing include:
* **Data Flow Validation**: Checking if information passes correctly between modules.
* **System Interaction**: Making sure different parts of the system communicate as expected.
* **Stability Checks**: Ensuring that small changes in one module do not break other connected modules.
* An example can be seen in a React application. After a user submits a form with their details, integration testing would verify whether:
* The data is properly sent to the server.
* The backend correctly updates the database.
* The React virtual DOM reflects the updated data without errors.
* Integration tests are considered successful when both the database shows the correct updates and the user interface (DOM) reflects those changes accurately.
* These tests are vital for maintaining smooth communication between services, databases, user interfaces, and external APIs. By focusing on how well modules work together, integration testing makes sure that the complete system remains reliable and user-friendly as it grows and evolves.

**Load Testing**

* Load testing is a type of testing where we simulate real-world user traffic to see how the system behaves under heavy usage. It is about figuring out the maximum load the system can handle before it starts slowing down, crashing, or becoming unresponsive.
* This type of testing is critical for applications expected to have many users at the same time, like e-commerce platforms during sales, online ticket booking systems, or banking apps.
* In load testing, testers simulate hundreds or thousands of virtual users performing actions simultaneously to observe how the system reacts. It helps in identifying bottlenecks like slow database queries, memory leaks, or network issues.
* Load testing tools like Apache JMeter and Locust are widely used to perform such simulations.
* By doing proper load testing, developers can ensure the system remains stable, responsive, and available even under peak usage conditions. It also helps businesses avoid outages that can lead to loss of revenue and customer trust.

**Performance Testing**

* Performance testing is a broader concept that includes load testing but also covers many other aspects of an application’s responsiveness and stability. It focuses on how quickly a system responds to user actions, how efficiently it uses system resources, and how it behaves under different workloads over time.
* In performance testing, testers measure various metrics like response time, throughput, resource utilization (CPU, memory, disk), and system scalability.
* An application might pass all functional tests, but if it loads slowly or crashes under pressure, users will not be happy. That is why performance testing is crucial.
* It includes stress testing, which pushes the system beyond its expected limits to see how it breaks, and endurance testing, which keeps the system running under load for an extended period to check for memory leaks or performance degradation.
* Popular tools for performance testing include JMeter, Gatling, and k6. A well-performing application leads to better user experience, higher user satisfaction, and better chances of success in the competitive market.
* For example, performance testing is critical in platforms which use the Unreal Engine as it relies heavily on high spec systems.