**Unit Testing**

Unit testing - software testing technique where individual components or functions of a program are tested independently to ensure they work as expected. The goal is to validate that each unit of the software performs correctly in isolation.

**Key Characteristics of Unit Testing:**

* Focuses on small components such as functions, methods, or classes.
* Does not depend on external systems like databases, APIs, or network connections.
* Usually automated and written by developers using testing frameworks.
* Runs quickly and provides instant feedback.

**Common Unit Testing Frameworks:**

* **Python**: unittest, pytest

**2. Integration Testing**

Integration testing - a type of software testing where multiple components or modules are tested together to ensure they work properly as a system. It focuses on theinteraction between different units of the application.

**Key Characteristics of Integration Testing:**

* Tests multiple units together (e.g., function calls, API connections, database interactions).
* Checks data flow between modules to ensure proper communication.
* Requires a test environment that simulates real-world scenarios.
* Can be automated or manual.

**Examples of Integration Testing:**

1.Testing API Integration:

import requests

def test\_api\_response():

response = requests.get("https://api.example.com/data")

assert response.status\_code == 200

assert "data" in response.json()

1. **Testing a Database Connection:**

import sqlite3

def test\_database\_connection():

conn = sqlite3.connect("test.db")

cursor = conn.cursor()

cursor.execute("SELECT \* FROM users")

assert cursor.fetchone() is not None

**Types of Integration Testing:**

1. **Top-Down Approach** → Test higher-level components first, then add lower ones.
2. **Bottom-Up Approach** → Test lower-level modules first and integrate step by step.
3. **Big Bang Approach** → Combine all modules at once and test together.
4. **Hybrid Approach** → Mix of top-down and bottom-up approaches.

**Development Environment (DEV)**

Purpose: Developers write and test individual units or modules.

Testing Types:

Unit Testing – Validate each function/method in isolation.

Integration Testing – Ensure different components interact correctly.

(Mostly automated; minimal manual testing is done.)

**Testing/QA/Staging Environment**

Purpose: Comprehensive testing of the application.

Testing Types:

Functional Testing – Ensure application features work as expected.

Regression Testing – Check if new changes break existing functionality.

System Testing – Validate the entire system against requirements.

Performance Testing – Evaluate system speed and stability.

Security Testing – Identify vulnerabilities and risks.

Smoke Testing – Basic validation after deployment.

Sanity Testing – Quick checks on bug fixes.

**Pre-Production / User Acceptance Testing (UAT)**

Purpose: Final verification before releasing to production.

Testing Types:

User Acceptance Testing (UAT) – Ensure it meets business requirements.

Performance Testing – Check scalability under expected load.

Security Testing – Final security audit before going live.

Smoke Testing – Last-minute validation before release.

**Production Environment (PROD)**

Purpose: Live environment used by real users.

Testing Types:

Production Monitoring – Observe system behavior in real-time.

Security Testing – Ongoing security assessments.

Performance Testing – Monitor performance under real-world conditions.

Smoke Testing – Quick checks after deployment.