**Write a brief Summary of Any 5 Software Testing Styles**

**1. Unit Testing**

**Unit Testing** is the process of testing individual components or functions of a software application in isolation from the rest of the system. The main goal is to ensure that each small unit of code behaves exactly as expected. Unit tests are typically written and executed by developers during the coding phase.

**Example:**  
Imagine a banking app with a function called calculateLoanEMI(principal, rate, time). A unit test would verify if the function correctly calculates the EMI for different inputs. For instance, if principal = 100,000, rate = 10%, and time = 5 years, the function should return the correct EMI value. The focus is only on the calculation logic, not on how the data is retrieved or displayed.

**Benefits:**

* Quick identification of bugs
* Encourages better code design
* Simplifies debugging and maintenance

**Common Tools:** JUnit (Java), NUnit (.NET), PyTest (Python)

**2. Integration Testing**

**Integration Testing** checks how different modules or components of an application interact with each other after unit testing is complete. While units might work perfectly alone, integration testing ensures they collaborate correctly when combined.

**Example:**  
In a flight booking system, one module handles payment, and another manages ticket generation. Integration testing would check if, after a successful payment, the system correctly generates and sends the ticket to the customer. Even if both modules work independently, bugs might arise when they are integrated, such as incorrect ticket details or failure in email delivery.

**Approaches:**

* **Top-Down Integration:** Testing starts from the top module and moves downward.
* **Bottom-Up Integration:** Testing starts from the lower modules and moves upward.
* **Big Bang Integration:** All modules are tested together after integration.

**Benefits:**

* Early detection of interface issues
* Improves overall system reliability

**Common Tools:** Postman (API testing), Selenium, Junit

**3. System Testing**

**System Testing** is a complete, end-to-end test of the entire software system as a whole. It checks whether the application meets all specified requirements, both functional and non-functional (performance, security, usability).

**Example:**  
For an online shopping platform, system testing would involve simulating the entire process — from logging in, browsing products, adding items to the cart, making payments, and receiving order confirmations. It would also involve checking if features like discount codes, payment gateways, and customer reviews are working properly together.

**Types of System Testing:**

* Functional Testing
* Performance Testing
* Security Testing
* Usability Testing

**Benefits:**

* Verifies the system's complete behavior
* Catches critical defects before release

**Common Tools:** Selenium, LoadRunner, QTP/UFT

**4. Acceptance Testing**

**Acceptance Testing** determines whether the software meets business requirements and is ready for delivery to the client or market. It is often conducted by the client, product owners, or end-users.

**Example:**  
Suppose a company develops a mobile app for food delivery. During acceptance testing, real users might place test orders, make payments, track deliveries, and provide feedback. If users find the app easy to navigate and the deliveries happen as expected, the product "passes" acceptance testing and is considered ready for launch.

**Types:**

* **Alpha Testing:** Done by internal employees.
* **Beta Testing:** Done by real users in a real environment.

**Benefits:**

* Ensures the product meets customer expectations
* Identifies usability or business flow issues early

**Common Tools:** TestRail (for managing acceptance tests), UserTesting

**5. Regression Testing**

**Regression Testing** is performed to ensure that new code changes (like bug fixes or new features) have not adversely affected the existing functionality. It re-runs previously executed tests to detect any unintended side effects.

**Example:**  
After adding a new "promo code" feature to a ticket booking app, regression testing would check if the basic booking process, payment gateway, and ticket generation still work without errors. Even a small change in code can potentially break unrelated parts of the application.

**Benefits:**

* Maintains system stability after updates
* Detects unexpected bugs caused by changes
* Saves time and effort with automated test cases

**Common Tools:** Selenium, Katalon Studio, TestComplete

**Conclusion**

Each testing style serves a specific and crucial purpose:

* **Unit testing** ensures individual parts are correct.
* **Integration testing** ensures components interact properly.
* **System testing** validates the system's overall functionality.
* **Acceptance testing** ensures business goals are met.
* **Regression testing** ensures stability after changes.

Together, these testing styles form a strong foundation for delivering high-quality, reliable, and user-friendly software. Effective testing not only reduces bugs and improves performance but also boosts customer satisfaction and trust in the final product.