ID: 27381

ASSIGNMENT 2

Write a brief summary of any 5 Software Testing Styles.

**1. Unit Testing**

**➔ Purpose:**

* Verify that a single unit of code (function, method, object, etc.) behaves as expected.
* Units are the smallest testable parts of an application.

**➔ Process:**

1. Developer writes a test case for each function or method.
2. Use dummy data (or mocks) if the unit depends on external services.
3. Run tests after every code change.

**➔ Example:**

Suppose you have a function add (a, b) that returns the sum.

* A unit test would check if add (2,3) returns 5.

**➔ Advantages:**

* Early detection of bugs.
* Simplifies code refactoring.
* Documentation for how functions should behave.

**➔ Challenges:**

* Mocking external dependencies can be complex.
* Not effective for finding integration issues.

**➔ Tools:**

* **JUnit** (Java), **pytest** (Python), **NUnit** (.NET), **Jest** (JavaScript).

**2. Integration Testing**

**➔ Purpose:**

* Test the interaction between integrated modules to ensure they work together.

**➔ Process:**

1. Identify modules to integrate.
2. Create integration tests focusing on data flow between modules.
3. Use stubs or drivers if some parts are not ready.

**➔ Example:**

An app where the user registration form connects to a database:

* Integration testing would check if the form correctly saves user data into the database.

**➔ Advantages:**

* Catches issues with communication between components early.
* Ensures that combined modules produce expected outcomes.

**➔ Challenges:**

* Setting up the test environment can be complex.
* Debugging is harder because failures might involve multiple components.

**➔ Tools:**

* **Postman** (for API testing), **Selenium** (for UI flow integration), **JUnit with mocks**.

**3. System Testing**

**➔ Purpose:**

* Validate the complete, fully integrated software system against the specified requirements.

**➔ Process:**

1. Test entire workflows (end-to-end).
2. Conduct functional tests (does it work?) and non-functional tests (how well does it perform?).
3. Simulate production environment as much as possible.

**➔ Example:**

Testing an e-commerce system:

* Searching for products, adding to cart, applying coupons, making a payment, and getting an invoice.

**➔ Advantages:**

* Ensures the whole system works together correctly.
* Finds environment-related issues (e.g., database config, load handling).

**➔ Challenges:**

* Time-consuming and expensive.
* Requires a stable, production-like environment.

**➔ Tools:**

* **Selenium** (for UI automation), **LoadRunner** (performance), **OWASP ZAP** (security).

**4. Acceptance Testing**

**➔ Purpose:**

* Determine if the system meets business needs and is acceptable for delivery.

**➔ Process:**

1. Client or end-users define *Acceptance Criteria* (conditions the software must satisfy).
2. Testers create acceptance tests based on these conditions.
3. Testing is often done manually by business stakeholders.

**➔ Example:**

In a ride-sharing app, acceptance testing would include:

* Booking a ride, seeing driver details, and completing payment without technical issues.

**➔ Advantages:**

* Ensures real-world usability.
* Boosts client and user confidence.

**➔ Challenges:**

* Users might not know how to test technically.
* Managing expectations if acceptance criteria were unclear.

**➔ Tools:**

* **Cucumber** (BDD acceptance testing), **FitNesse**, **TestRail**.

**5. Regression Testing**

**➔ Purpose:**

* Ensure that a recent code change has not negatively impacted existing features.

**➔ Process:**

1. After every change (bug fix, feature addition), re-run previous test cases.
2. Prioritize test cases related to critical functionalities.
3. Automate repetitive tests where possible.

**➔ Example:**

Suppose a new payment gateway is added. Regression tests should:

* Check not only the new payment method but also verify that the old methods still work correctly.

**➔ Advantages:**

* Protects against unexpected bugs after updates.
* Essential for continuous integration/continuous delivery (CI/CD).

**➔ Challenges:**

* Regression test suites can become very large and time-consuming.
* Maintenance of test cases needed after system changes.

**➔ Tools:**

* **Selenium** (automated UI testing), **TestComplete**, **Appium** (mobile apps).

