***Name: Tabammum Haque Pranty***

***SQA Assignment-2***

* **Testing Principles:** There are seven testing principles such as:
* **Testing Shows the presence of defects:** Testing can reveal the presence of defects, but it cannot prove that there are no defects. No matter how many tests you run, you can never guarantee that the software is 100% defect-free.

Example: After performing several modules of testing on a mobile app, you find several bugs and fix them. While the app seems stable, it doesn’t mean there are no hidden bugs left, there could still be some that are undiscovered due to incomplete test cases or scenarios.

* **Exhaustive Testing is Not Possible:** Testing every possible input and scenario for a system is practically impossible because there are infinite combinations of inputs, environments, and conditions.

Example: Consider a login page that accepts alphanumeric passwords. Testing every combination of letters, numbers, and special characters for every password length is infeasible.

* **Early testing:** Testing should start as early as possible in the software development life cycle (SDLC) to find defects early when they are easier to fix.

Example: In agile development, unit testing is performed during coding, and issues such as incorrect logic are identified before they can propagate to later stages like integration or system testing. Fixing an error in the design phase is much easier than fixing it after the product is released.

* **Defect Clustering:** A small number of modules usually contain the most defects. This follows the Pareto principle (80-20 rule), where 80% of the defects are often found in 20% of the code.

Example: While testing a shopping cart feature for an e-commerce website, you discover that most issues are concentrated around the payment processing system, even though it’s only a small part of the entire application. The rest of the application might be relatively bug-free.

* **Pesticide Paradox:** If the same tests are repeated over time, they will no longer find new bugs. To continue finding defects, test cases need to be regularly reviewed and revised.

Example: If a tester runs the same set of regression tests after every build, eventually, no new bugs are found. To overcome this, they can introduce new test cases or vary the conditions under which the tests are executed (e.g., different input data).

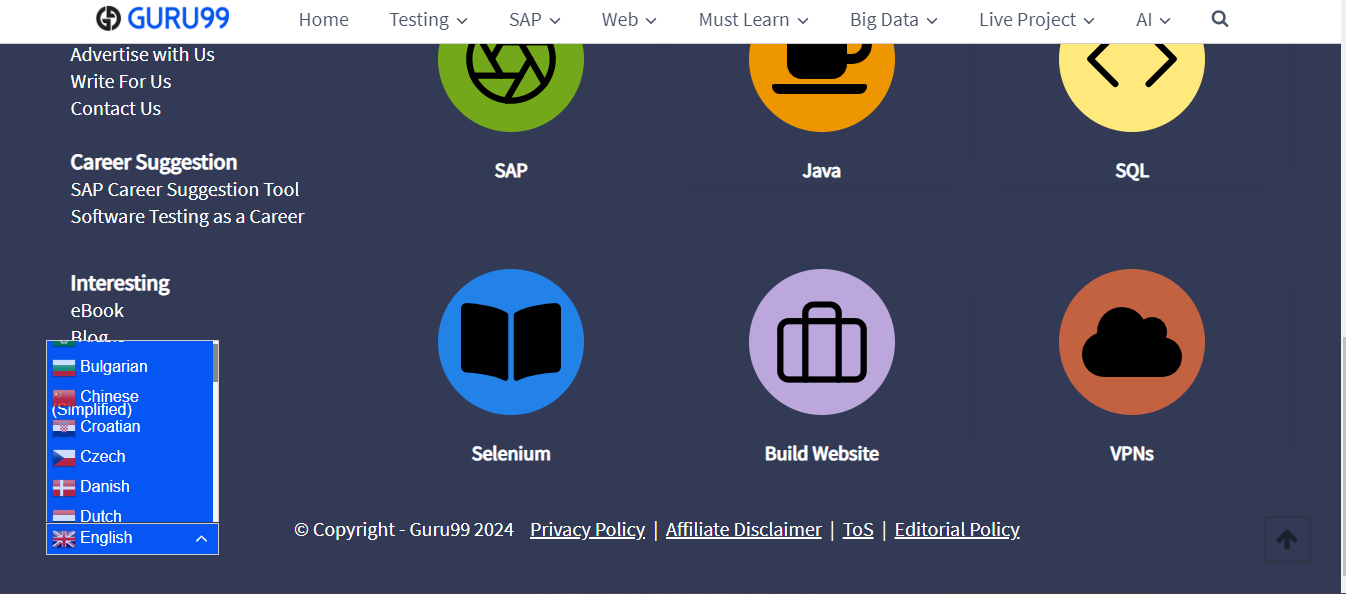
* **Testing is context-dependent:** The type of testing performed should depend on the context of the software. Different types of systems (e.g., web apps, financial systems) require different approaches to testing.

Example: A banking application might require rigorous security testing to ensure protection against cyber threats, whereas a gaming application might prioritize performance and usability testing to ensure smooth graphics and quick response times.

* **Absence of errors fallacy:** Even if the software is error-free, it does not mean it is useful or meets the users' needs. Software that solves the wrong problem or doesn’t satisfy user requirements is still a failure, even if it is defect-free.

Example: Suppose you’ve tested an e-learning platform extensively and found no bugs. However, the platform doesn’t allow teachers to upload large video files which is a critical requirement. In this case, the software is not defective but fails because it doesn't fulfill its intended purpose.

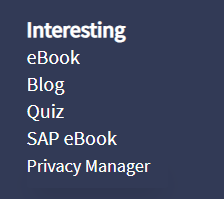
* **Bug Finding for a Website:**
* **Bug-1:**

****

Here is an issue in Chinese Simplified sentence. The simplified word should have been a space after Chinese word and before Croatian word.

URL: <https://www.guru99.com/>

* **Bug-2:**



In this picture, the Privacy Manager doesn’t work.

URL: <https://www.guru99.com/>