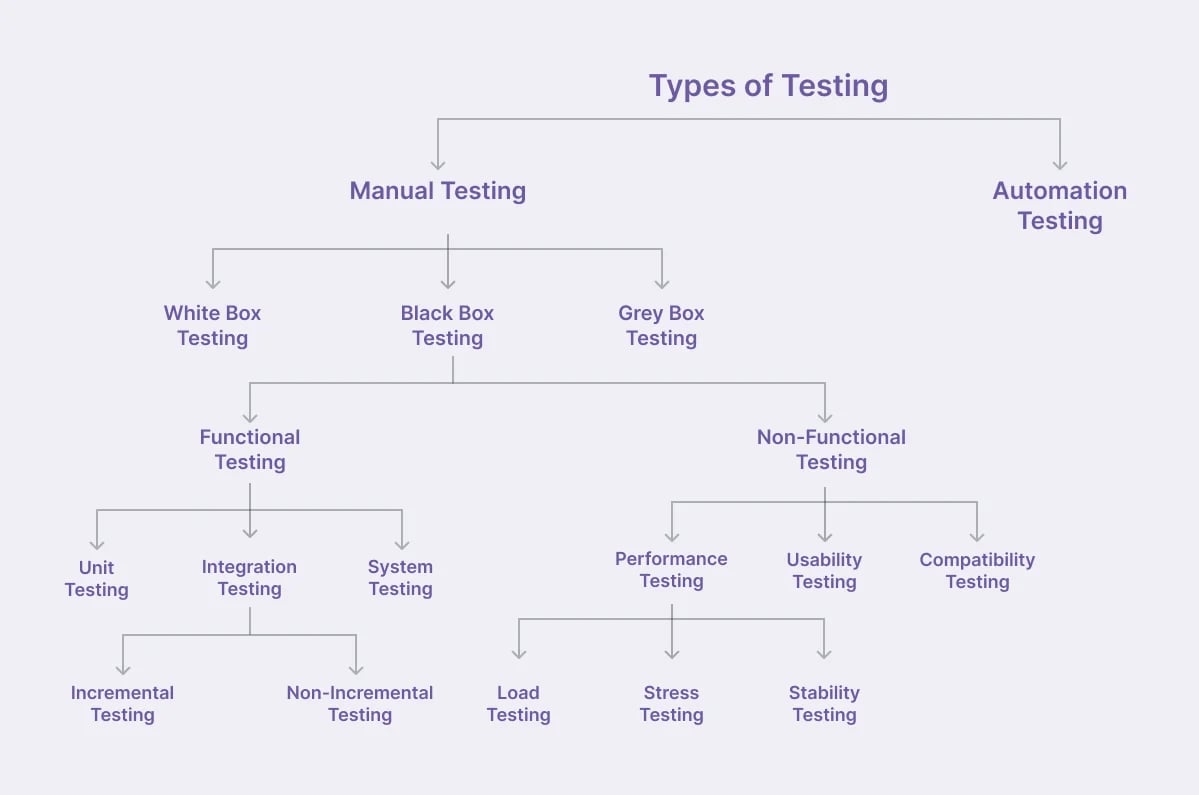
Write a brief summary of any 5 software testing styles.



**White box Testing:**

White Box Testing (also known as clear box, glass box, or structural testing) is a software testing methodology where the tester has full visibility into the internal workings of an application.

* Testers have access to the source code, allowing them to design test cases that exercise specific paths and conditions within the code.​
* Test cases are created based on the internal logic of the application, focusing on aspects like code paths, conditions, loops, and data flow.​
* Common techniques include statement coverage, branch coverage, path coverage, condition coverage, and data flow testing.

Advantages

* Allows for comprehensive testing of all code paths, leading to the identification of hidden errors.​
* Enables early detection of issues during the development phase, reducing the cost of fixing defects.​
* Helps in optimizing code by identifying redundant or inefficient code segments.​
* Facilitates the identification of security vulnerabilities within the code.

Disadvanges

* Requires a high level of expertise and understanding of the code, which can be time-consuming.​
* Can be resource-intensive due to the need for detailed analysis and test case creation.​
* May not effectively identify issues related to missing functionalities or requirements that are not part of the code.

**Black Box Testing:**

Black Box Testing is a software testing methodology where testers evaluate the functionality of an application without any knowledge of its internal code structure or implementation. Instead, they focus on inputs and outputs, ensuring the software behaves as expected according to its specifications.

Advantages

* Reflects how end-users will interact with the application.​
* Testers do not need to understand the code.​
* Suitable for testing complex systems where internal details are not accessible.

Disadvantages

* May not cover all possible paths or scenarios.​
* Test cases might overlap, leading to redundant testing.​
* Without knowledge of the internal code, some defects might go undetected.

Unit Testing:

Unit Testing is a fundamental software testing technique that involves verifying the functionality of individual units or components of a software application in isolation.

Advantages

* Identifies issues at an early stage, reducing the cost and effort of fixing bugs later in the development process.
* By ensuring that individual units function correctly, integration becomes smoother and more predictable.​
* Provides a safety net when making changes to the codebase, ensuring that existing functionality remains unaffected. ​
* Encourages developers to write cleaner, more modular code, leading to improved maintainability.

Disadvantages:

* Writing comprehensive unit tests can be time-consuming, especially for complex systems.
* Unit tests focus on individual components and may not detect issues arising from interactions between components.​
* As the codebase evolves, unit tests need to be updated to reflect changes, which can be resource-intensive.

Integration Testing:

Integration Testing is a software testing phase where individual modules or components, which have already passed unit testing, are combined and tested as a group to ensure they function together correctly.

Advantages

* Detects Interface Issues: Identifies problems in data flow and control between modules.
* Ensures Correct Module Interaction: Verifies that integrated components work together as expected.
* Improves System Reliability: Helps in building a robust system by addressing integration-related defects early.
* Facilitates Early Testing: Allows testing of modules before the entire system is complete, enabling quicker feedback.​

Disadvantages

* **Complex Fault Isolation**: Determining the exact source of defects can be difficult, especially in large systems.
* **Resource Intensive**: Requires significant time and resources to set up and execute tests.
* **Dependency Management**: Managing dependencies between modules can complicate the testing process.

**Regression Testing**

Regression Testing is a software testing practice aimed at ensuring that recent code changes—such as bug fixes, enhancements, or configuration updates—have not adversely affected the existing functionality of an application. The primary goal is to verify that previously working features continue to perform as expected after modifications.

Advantages:

* Ensures Stability: Confirms that new code changes do not disrupt existing functionality.​
* Identifies Side Effects: Detects unintended consequences of recent modifications.​
* Maintains Software Quality: Helps in delivering reliable and consistent software products.​
* Supports Agile Development: Facilitates continuous integration and delivery by ensuring ongoing software integrity.

Disadvantages:

* Time-Consuming: Executing comprehensive regression tests, especially without automation, can be labor-intensive and time-consuming. Even automated tests may take considerable time for large applications. ​
* Resource-Intensive: Regression testing requires significant resources, including skilled personnel and computing power, particularly when dealing with extensive test suites. ​
* Maintenance Overhead: As software evolves, maintaining and updating the regression test suite to reflect changes in functionality demands continuous effort and can become complex.
* False Positives and Negatives: Automated regression tests may produce false positives (indicating issues where there are none) or false negatives (failing to detect actual problems), leading to potential misinterpretations.