## Further aspects in Software Testing Strategies.

As outlined in "[Software Testing Strategies - GeeksforGeeks](https://www.geeksforgeeks.org/software-testing-strategies/)," the realm of software testing encompasses a diverse array of strategies, each serving a specific purpose in ensuring the quality and reliability of software systems. Some of the most prominent strategies include:

1. Black Box Testing: This method focuses on the functionality of the software without delving into its internal structure. It is useful for evaluating user experiences and desired outcomes. Our technical practice testing with Postman on part 4 of this thesis is a good sample for how black box testing work. There was no touch in the project’s code, but with those implemented tasks, the API of ITD system would be clearer and more optimized.
2. White Box Testing: In contrast to black box testing, this technique examines the software's internal structure to analyses its logic, code routes, and data flows. It is critical for identifying vulnerabilities and potential problems in the code.
3. Unit Testing: Unit testing involves assessing individual components or functions within the software. It's a foundational practice that ensures each part of the software performs as intended.
4. Integration Testing: This strategy evaluates how different components or modules work together within the software. It's crucial for detecting issues that may arise when these parts interact. It is supposed to be the need to do task after writing the Unit testing case.
5. Functional Testing: Functional testing verifies that the software functions in accordance with specified requirements. It's used to validate if the software performs its intended tasks.
6. System Testing: This approach assesses the entire software system. It ensures that all components function seamlessly together and meet the overall objectives.
7. Acceptance Testing: Acceptance testing confirms that the software meets the criteria set by stakeholders and is ready for deployment.
8. Performance Testing (Scalability): This type of testing evaluates the software's performance in terms of speed, responsiveness, and scalability. It ensures that the software can handle expected workloads without performance degradation.
9. Regression Testing: Regression testing involves retesting the software after changes or updates to ensure that new modifications have not introduced new issues.
10. Security Testing: Security testing aims to identify vulnerabilities and weaknesses in the software that could be exploited by malicious actors. It's crucial for safeguarding data and user privacy.

These diverse testing strategies play a fundamental role in ensuring the functionality, reliability, and security of software applications. In future work, it will be important to consider how these strategies can be improved, automated, and further integrated into the software development process to improve software quality, reduces the risk of errors, and reduces testing time for a software development life cycle. Additionally, exploring emerging testing methods and tools will be essential to keep up with evolving software development technologies and practices.

Over the past decade, a wave of new programming languages has emerged with the primary aim of enhancing the coding process, making it more error-resistant. TypeScript stands as a prime example of this trend. It is a robust programming language that enables developers to define data types with precision before transpiling into the versatile JavaScript. TypeScript has become a valuable tool for programmers across the globe. Its adoption not only streamlines coding but also provides significant benefits for test engineers. By incorporating TypeScript into their skill set, test engineers can approach software development with a more sophisticated perspective, ultimately leading to the creation of easily maintainable and robust code for today's intricate software systems.

### Prior to Commencing Testing:

Based on the discussions at <https://www.geeksforgeeks.org/software-testing-strategies/>, a standard of software testing strategies should have:

1. Precise Requirement Specification: Clearly define software requirements, covering maintainability, risk assessment, and usability, to ensure error-free test outcomes.
2. Objective Clarity: Specify testing objectives, including effectiveness, failure identification, and defect cost, in a comprehensive test plan.
3. User Profiling: Identify user categories and create profiles to align testing with real-world needs.
4. Structured Test Plan: Develop a robust test plan for efficient and rapid-cycle testing. Our script test in chapter 4 is a good example, but test plan include many more script tests.
5. Self-testing Software: Create software capable of self-diagnosing errors, with automated and regression testing support.
6. Filter with Formal Reviews: Use technical reviews before testing to uncover overlooked errors, reducing testing time.
7. Technical Review Evaluation: Assess the quality and effectiveness of test strategy and cases, improving testing quality.
8. Continuous Improvement: Apply statistical process control to continually enhance software quality during development.



**Figure 78.** A standard of software testing strategies.

- source: <https://www.geeksforgeeks.org/software-testing-strategies/>

### Why we need to test the software?

Once again, let’s back deeper to the question of “Why we need to test the software?”

Software testing is a fundamental practice in the software development lifecycle, and it serves several crucial purposes:

Bug Identification: Testing helps in uncovering and identifying software defects, commonly known as bugs. By detecting and addressing these issues early, software quality is significantly improved.

Quality Assurance: Testing is an assurance of software quality. It ensures that the software behaves as intended, meets user requirements, and operates reliably.

Risk Mitigation: It reduces the risk of deploying faulty software. By testing, we can discover potential issues before they affect end-users or critical business operations.

User Satisfaction: Testing ensures that the software provides a smooth and error-free experience for end-users. Satisfied users are more likely to continue using the software.

Compliance: In certain industries, such as healthcare and finance, software must adhere to strict regulatory standards. Testing is essential to ensure compliance with these regulations.

Cost Reduction: While testing may seem like an additional expense, it can save significant costs in the long run by identifying and fixing problems early in the development process.

### What we need to improve for the disadvantage of Software Testing?

Improvement in software testing is a paramount concern, acknowledging that it consumes substantial time and resources. To optimize the testing workflow, we must prioritize the development of a structured test plan and methodical test scripts. This aligns the entire team toward cohesive, professional work, channeling collective energy and preventing the dissipation of effort on unproductive pursuits.

Effective planning necessitates meticulous research into software and testing tools. Their costs must be become more controllable for a project team.

In the contemporary landscape, software testing tools and frameworks, exemplified by Selenium or Jenkins[[1]](#footnote-1), are continually evolving, and growing in power. Equipping test engineers with the necessary skills is essential. Proficiency in these tools is crucial for success within the team. We can distill the importance of tools and skill acquisition based on the need for effective testing and resource optimization.

In the near future, based on the need of testing engineer, there would be more robust or modern tools for software testing. As a testing member of any organization, updating continuously useful latest tools knowledge is an essential skill.

Furthermore, testing management must be unforgettable, the management ought to ensure that the realism of the software based on the real-life interaction, environment, weather, … This process should also prioritize the test maintenance by continuously reviewing and updating test cases and plans as software evolves. Implement version control for test assets by for example Github, Gitlab, Azure DevOps, …

1. Jenkins is an open-source automation server that helps with building, testing, and deploying software. It provides a way to automate many tasks in the software development process, including code integration, testing, and even deployment. Jenkins is widely used in DevOps and continuous integration/continuous delivery (CI/CD) pipelines, where it automates the building and testing of software as changes are made by developers. [↑](#footnote-ref-1)