1. **REFACTORING:**

Refactoring is important in agile environment. It improves the internal structure of the code without changing the functionality.  By refactoring regularly, we update the code so that it reflects an improved understanding of the domain. This alignment enables easier modification. It improves the design the software. Refactoring is a way of sharing knowledge. After refactoring a piece of code, we gain a deeper understanding of what it does, even if you didn't originally write it. Spreading knowledge is crucial when you're [working as a team](https://www.coscreen.co/blog/even-10x-engineers-work-best-in-a-team/). Refactoring is an activity that benefits greatly from collaboration. If you do any amount of [pair programming](https://www.coscreen.co/blog/love-or-hate-pair-programming-here-to-stay/), you'll likely work on areas that either your pair or you don't fully understand. Refactoring this code together is a great way to gain a shared understanding, transfer knowledge, and develop a codebase that looks like it was written by a single person.

Refactoring does not mean that we are rewriting code, fixing bugs, improve observable aspects of software such as its interface. Refactoring in the absence of safeguards against introducing defects is risky. Safeguards include aids to regression testing including automated unit tests or automated acceptance tests, and aids to formal reasoning such as type systems. Refactoring prevents code rot, such as bad dependencies between classes, myriad patches, incorrect allocation of class responsibilities, and duplicate code, resulting in a more efficient code base. The time taken to refactor pays dividends, since it’s easier to clean code closer to when it was written rather than rush to fix problems later.

1. **TEST DRIVEN DEVELOPMENT**

Test-driven development (TDD) is a [software development process](https://en.wikipedia.org/wiki/Software_development_process) relying on software requirements being converted to [test cases](https://en.wikipedia.org/wiki/Test_case) before software is fully developed, and tracking all software development by repeatedly testing the software against all test cases. This is as opposed to software being developed first and test cases created later. It is software development approach in which test cases are developed to specify and validate what the code will do. In simple terms, test cases for each functionality are created and tested first and if the test fails then the new code is written in order to pass the test and making code simple and bug-free.

TDD framework instructs developers to write new code only if an automated test has failed. This avoids duplication of code. When a test fails, you have made progress because you know that you need to resolve the problem. TDD ensures that your system actually meets requirements defined for it. It helps to build your confidence about your system. In TDD more focus is on production code that verifies whether testing will work properly. In TDD, you achieve 100% coverage test. It shortens the programming feedback loop. It is detailed specification. Test-driven development is a process of modifying the code in order to pass a test designed previously.

1. **USER ACCEPTANCE TESTING**

**User Acceptance Testing (UAT)** is a type of testing performed by the end user or the client to verify/accept the software system before moving the software application to the production environment. UAT is done in the final phase of testing after functional, integration and system testing are done. The main **Purpose of UAT** is to validate end to end business flow. It does not focus on cosmetic errors, spelling mistakes or system testing. User Acceptance Testing is carried out in a separate testing environment with production-like data setup. It is kind of black box testing where two or more end-users will be involved.

**Need of User Acceptance Testing** arises once software has undergone Unit, Integration and System testing because developers might have built software based on requirements document by their own understanding and further required changes during development may not be effectively communicated to them, so for testing whether the final product is accepted by client/end-user, user acceptance testing is needed. The acceptance test activities are carried out in phases. Firstly, the basic tests are executed and if the test results are satisfactory then the execution of more complex scenarios are carried out.