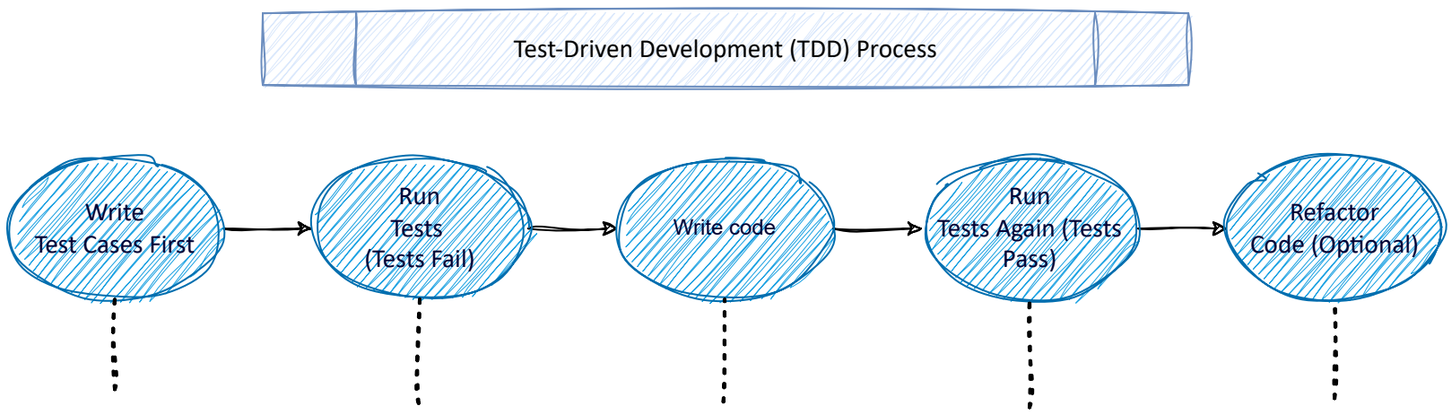
DAY - 3

Assignment 1:

Create an infographic illustrating the Test-Driven Development (TDD) process. Highlight steps like writing tests before code, benefits such as bug reduction, and how it fosters software reliability.



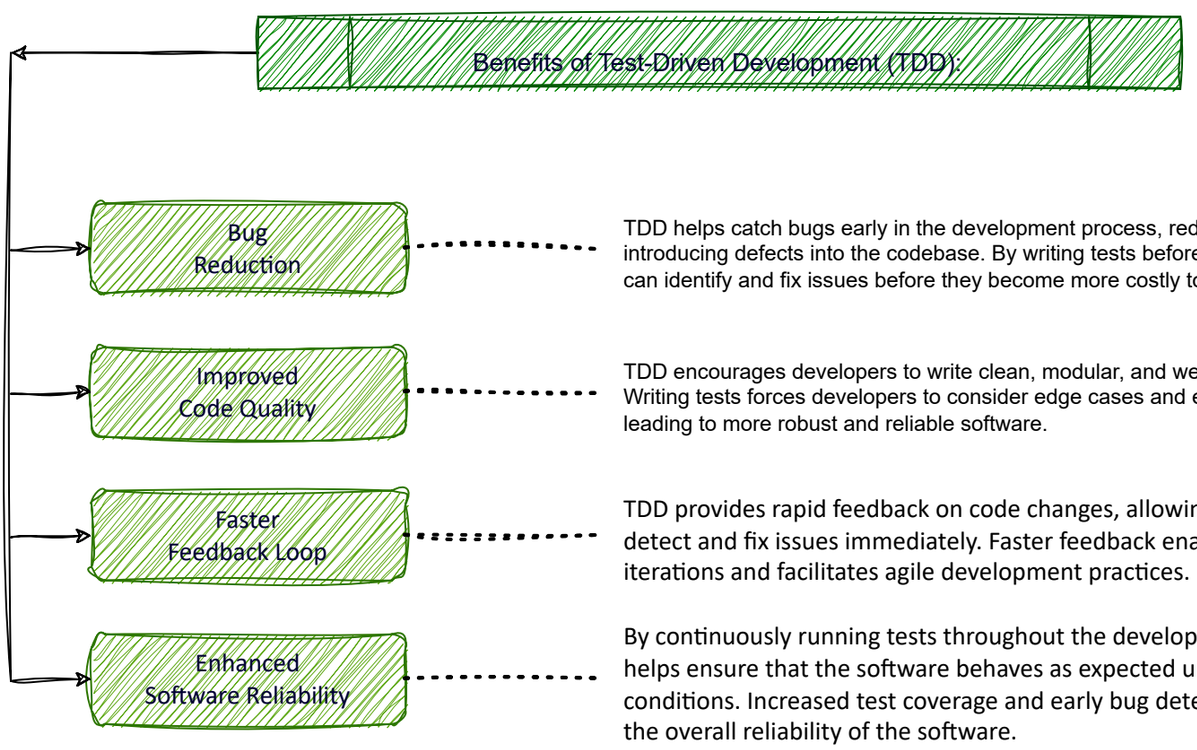
Developers start by writing test cases that define the expected behavior of the code. Test cases are written based on the requirements or specifications of the feature being implemented.

Developers run the tests, expecting them to fail initially since the corresponding code hasn't been written yet. Failing tests indicate that the expected behavior hasn't been implemented yet.

Developers write the minimum amount of code necessary to make the failing tests pass. Code is written incrementally, with each piece of functionality being added to fulfill the requirements specified by the tests.

After writing the code, developers rerun the tests to ensure that they now pass. Passing tests indicate that the code behaves as expected and meets the specified requirements.

Once tests pass, developers may refactor the code to improve its design, readability, or performance, refactoring ensures that the code remains maintainable and adheres to coding standards.



TDD helps catch bugs early in the development process, reducing the likelihood of introducing defects into the codebase. By writing tests before code, developers can identify and fix issues before they become more costly to address.

TDD encourages developers to write clean, modular, and well-structured code. Writing tests forces developers to consider edge cases and error conditions, leading to more robust and reliable software.

TDD provides rapid feedback on code changes, allowing developers to detect and fix issues immediately. Faster feedback enables quicker iterations and facilitates agile development practices.

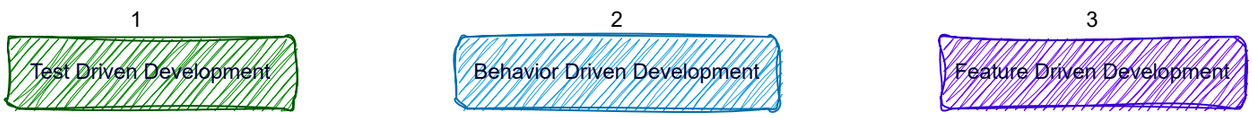
By continuously running tests throughout the development process, TDD helps ensure that the software behaves as expected under various conditions. Increased test coverage and early bug detection contribute to the overall reliability of the software.

DAY - 3

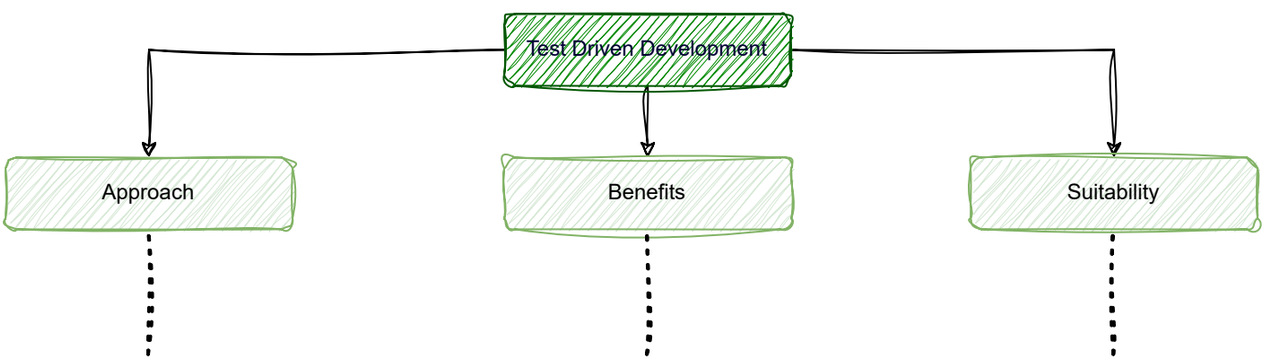
Assignment 2:

Produce a comparative infographic of TDD, BDD, and FDD methodologies. Illustrate their unique approaches, benefits, and suitability for different software development contexts. Use visuals to enhance understanding.

There are three comparative analysis of Software Development Methodologies they are



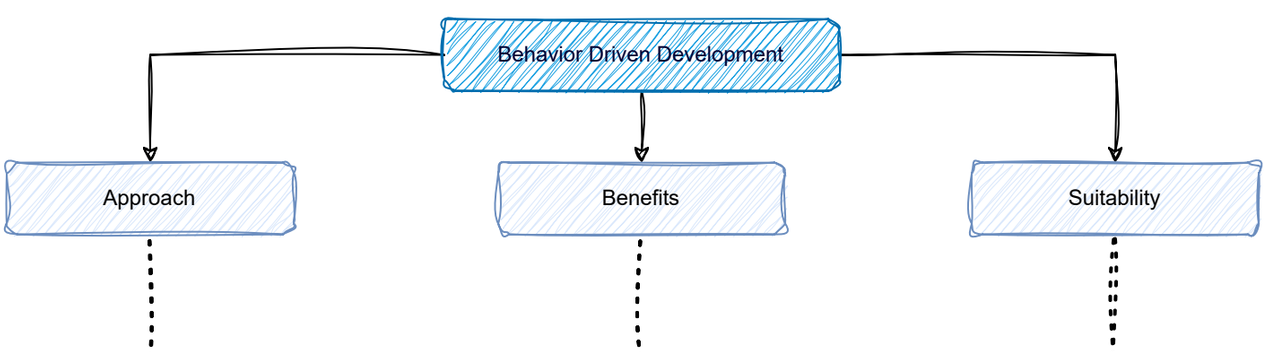
We can see the approach, benefits, and suitability for each of the driven development methodologies



Write tests before writing code. Focus on small, incremental development cycles. Tests serve as specifications for the code.

Early bug detection and reduction. Improved code quality and maintainability. Faster feedback loop and increased developer confidence.

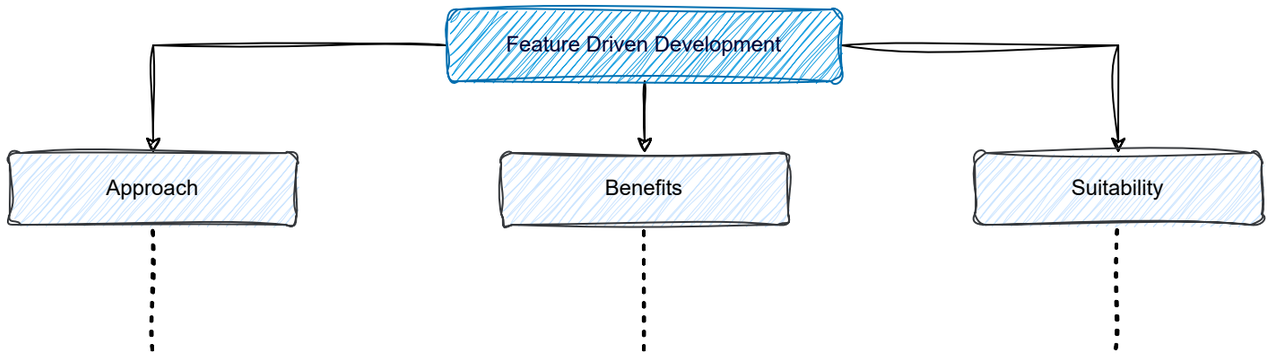
Ideal for projects with well-defined requirements. Suitable for projects where code quality and reliability are critical. Effective for teams practicing Agile or iterative development.



Define behavior using natural language specifications. Collaboration between developers, QA, and stakeholders. Automated scenarios serve as both documentation and tests.

Shared understanding between stakeholders. Aligns development with business goals. Encourages collaboration and communication.

Suitable for projects with complex or changing requirements. Ideal for teams with diverse skill sets and stakeholders. Effective for ensuring that software meets user expectations.



Focus on delivering features incrementally. Divide development into feature sets and iterations. Emphasize collaboration and progress tracking.

Clear focus on delivering tangible features. Efficient use of resources and time. Facilitates early and continuous delivery.

Ideal for largescale projects with multiple features. Suitable for teams with well-defined processes and roles. Effective for projects requiring frequent releases and customer feedback.