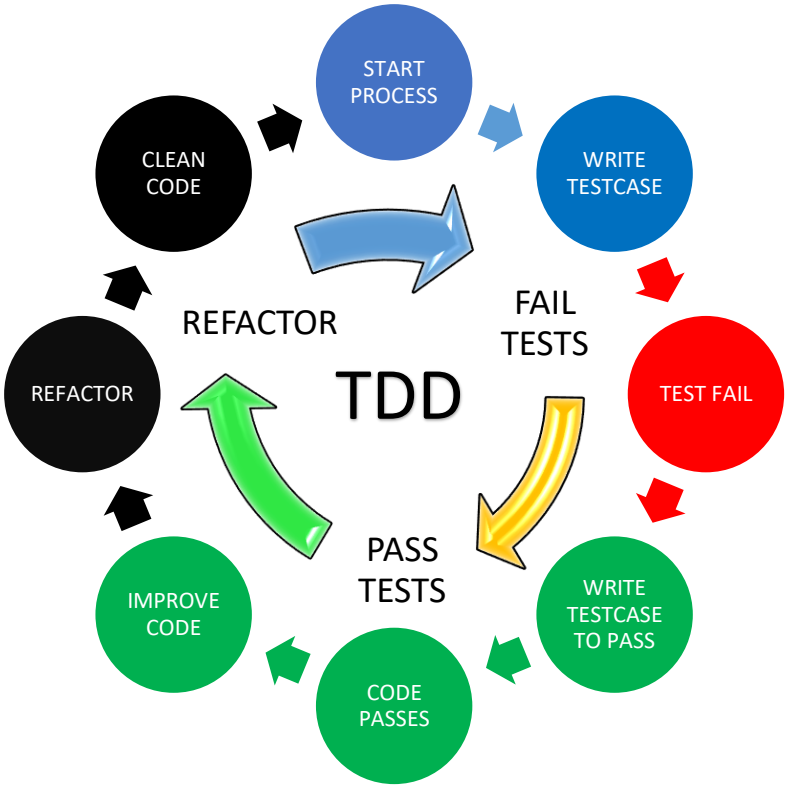
Assignment 1: Create an infographic illustrating the Test-Driven Development (TDD) process. Highlight steps like writing tests before code, benefits such as bug reduction, and how it fosters software reliability.

Test Driven Development (TDD):

- It's a very simple method in which we write test so that it fails then write test for passing it. After that we will refactor our code by making appropriate changes and cleaning up.

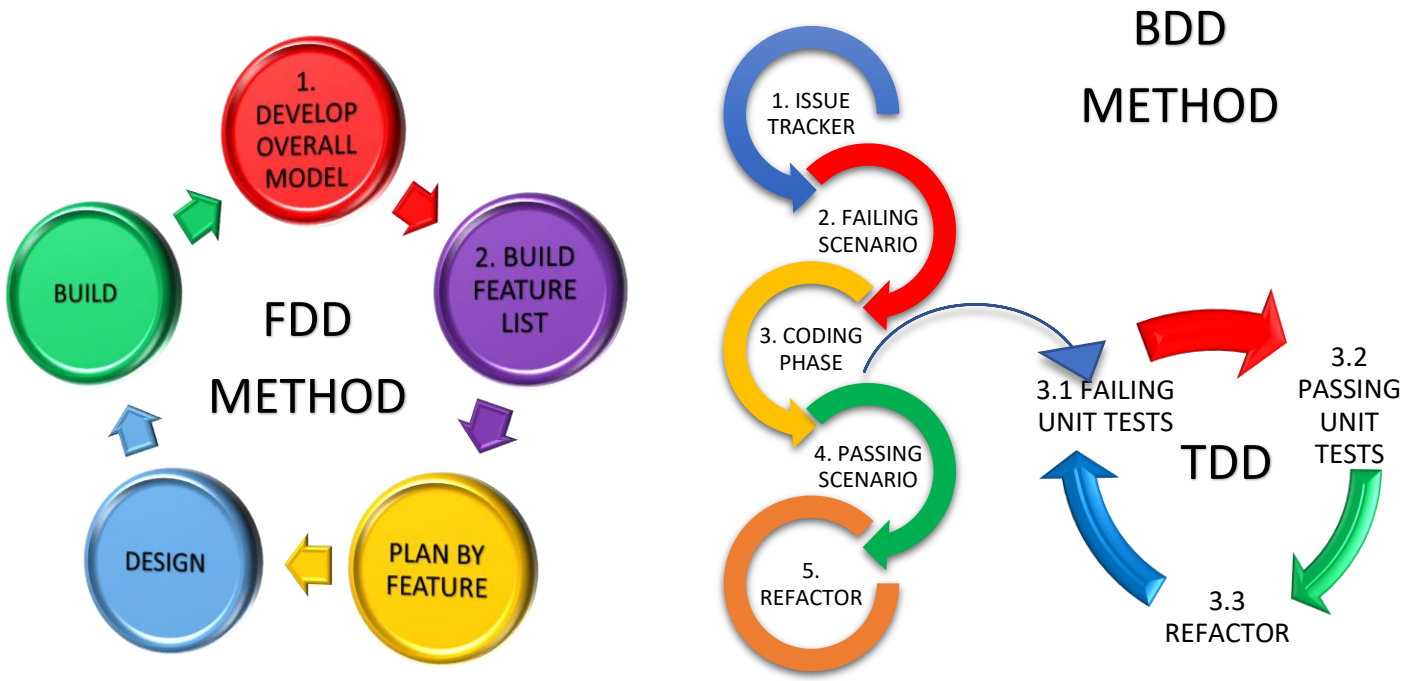


- ***Step involved in TDD:***
  - **Write a Test Case:** Define a test based on the expected functionality of the software.
  - **Run the Test (It Should Fail):** Execute the test case before implementing the actual functionality.
  - **Write code to pass the test:** Write minimal code necessary to pass the test case.
  - **Run the test:** The code must satisfy the condition/functionality.
  - **Refactor the code:** Optimize the code and handle relevant changes without changing the core behaviour.
  - **Repeat all test scenarios:** Continue adding new test cases and make sure to pass them and make a reliable codebase and clean up code.

- **Benefits of TDD:**
  - **Bug Reduction:** Detects and prevents bugs early in development. For Example, a function behaves incorrectly, the test fails immediately, in very early phase highlighting the issue.
  - **Improved Code Quality:** Continuous code Refactoring makes optimized and reliable code.
  - **Reliability of Final Software:** In TDD Process tests acts as documentation so software behaviour is as per expectation.

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.

Comparison Between FDD, TDD, FDD:



A diagramatic view for TDD, FDD and BDD

TDD:

Description:

- Write a test before implementing the code.
- Focuses on verifying small and specific functionality.
- Ensures code correctness and refactoring from start.

Benefits:

- Bug reduction
- early detection of bugs.
- Cleaner code.

Suitability:

- Ideal for small, iterative feature development.
- Well-suited for projects with complex logic.

BDD:

Description:

- Better Collaboration between developers, testers, and clients.
- Functionality Definition is clear.

Benefits:

- Better team communication.
- Ensures alignment with business requirements.
- No gap between Technical and non-technical stakeholders.

Suitability:

- Project where better collaboration needed.
- Suitable for applications with extensive user interactions.

FDD:

Description:

- Focuses on delivering features incrementally.
- Breaks the system into manageable features.

Benefits:

- Emphasizes functionality delivery and making prototype.
- Ensures progress tracking with clear milestones.

Suitability:

- Ideal for large-scale applications with multiple modules.
- Suitable for teams with well-defined processes and domain expertise.

Features	*TDD*	*BDD*	*FDD*
Focus	Code Correctness	Behaviour Alignment	Feature Delivery
Key Principle	Red-Green-Refactor Cycle	Collaboration And Shared Understanding	Feature-Centric Development, Short Iterations
Communication	Primarily Developer-Focused	Focus On Collaboration Between Developers, Testers, And Business Stakeholders	Emphasizes Clear Communication Within Development Teams
Tools	Unit Testing Frameworks (E.G., Junit)	BDD Frameworks (E.G., Cucumber, Specflow)	Project Management Tools, Version Control Systems
Key Benefit	*Bug Free Code*	*Clear Communication*	*Incremental Progress Tracking*