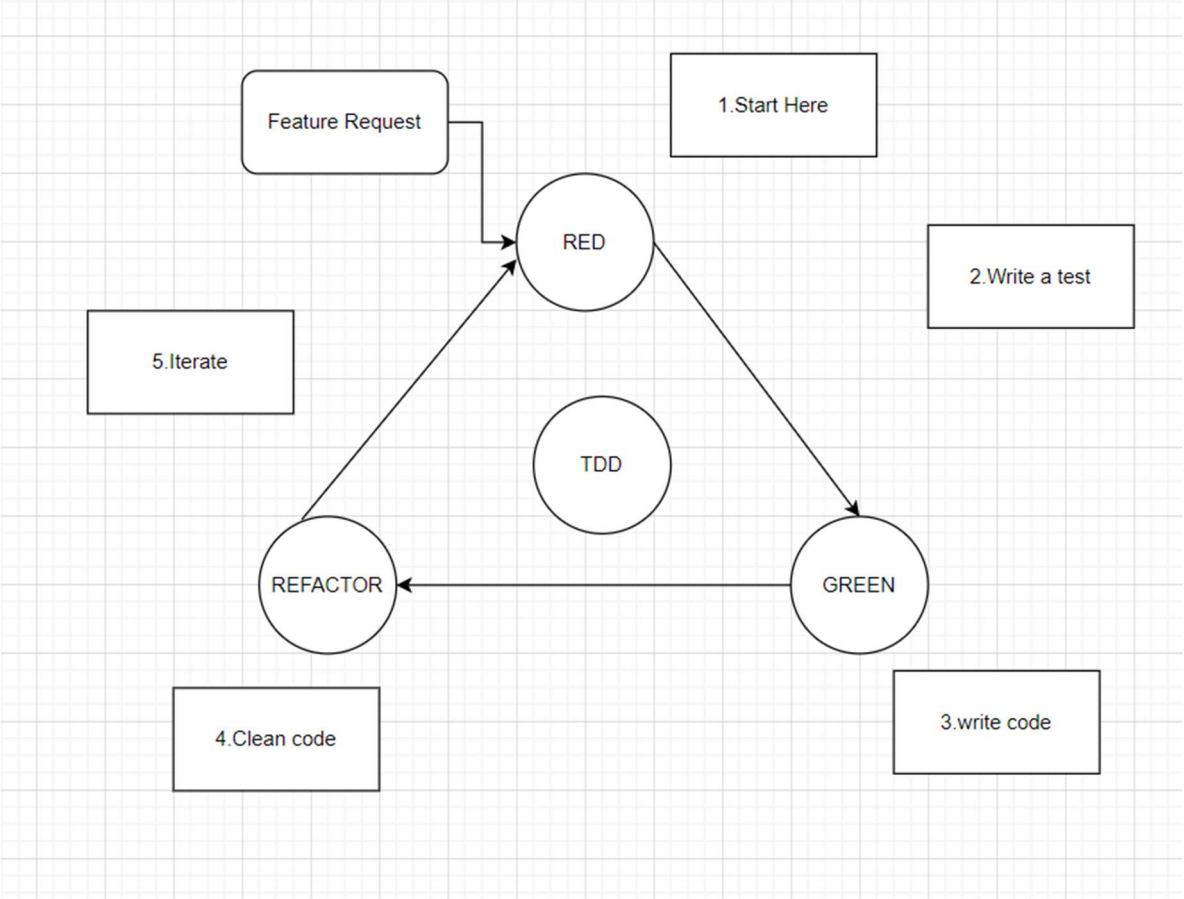
Assignment 1: Create an infographic illustrating the Test-Driven Development (TDD) process. Highlight steps like writing tests before code, benefits such as bug reduction, and how it fosters software reliability.

Infographic of Test-Driven Development (TDD) process.



TDD stands for Test-Driven Development. It's a software development approach where we write automated tests before we write the actual code.

Steps of TDD:

Step 1: Write a Test

Write a test for the next bit of functionality we want to add. Initially, this test will fail because the functionality doesn't exist yet.

Step 2: Run the Test

Run the test to see it fail. This confirms that the test is working correctly and that the feature is not yet implemented.

Step 3: Write the Code

Write the minimal amount of code necessary to make the test pass.

Step 4: Run the Test Again

Run all tests to see if the new code passes the test. If it doesn't, adjust the code until it does.

Step 5: Refactor:

Clean up your code while keeping the test passing. Improve the structure and efficiency without changing its behaviour.

Step 6: Repeat:

Continue the cycle with the next functionality.

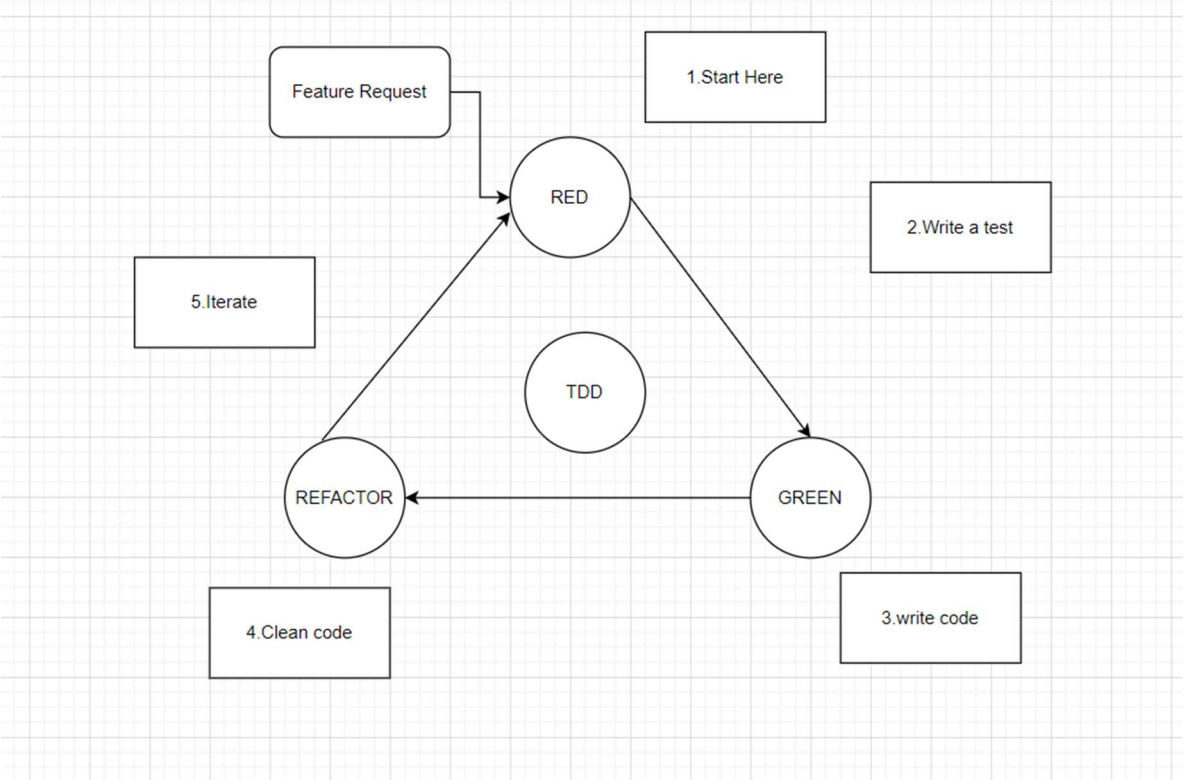
Benefits of Writing Tests Before Code

- **Early Bug Detection:** Identifies bugs at the earliest stage of development, reducing the cost and effort of fixing them later.
- **Clear Requirements and Design:** Forces a clear understanding of the requirements and the design before coding begins, leading to better-structured code.
- **Focused Development:** Ensures that development is guided by specific requirements, preventing feature creep and unnecessary functionality.
- **Improved Code Quality:** Encourages writing modular, testable code that is easier to maintain and extend.

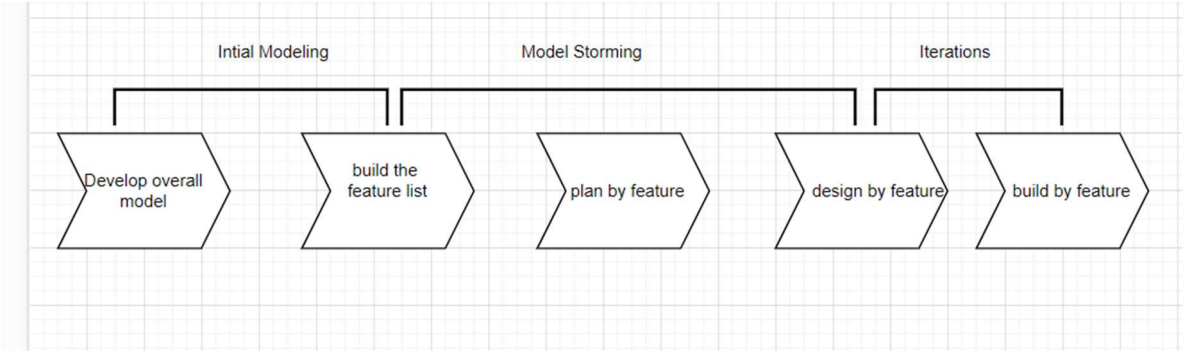
Assignment 2: Produce a comparative infographic of TDD, BDD, and FDD methodologies. Illustrate their unique approaches, benefits, and suitability for different software development contexts. Use visuals to enhance understanding.

infographic of TDD, BDD, and FDD methodologies

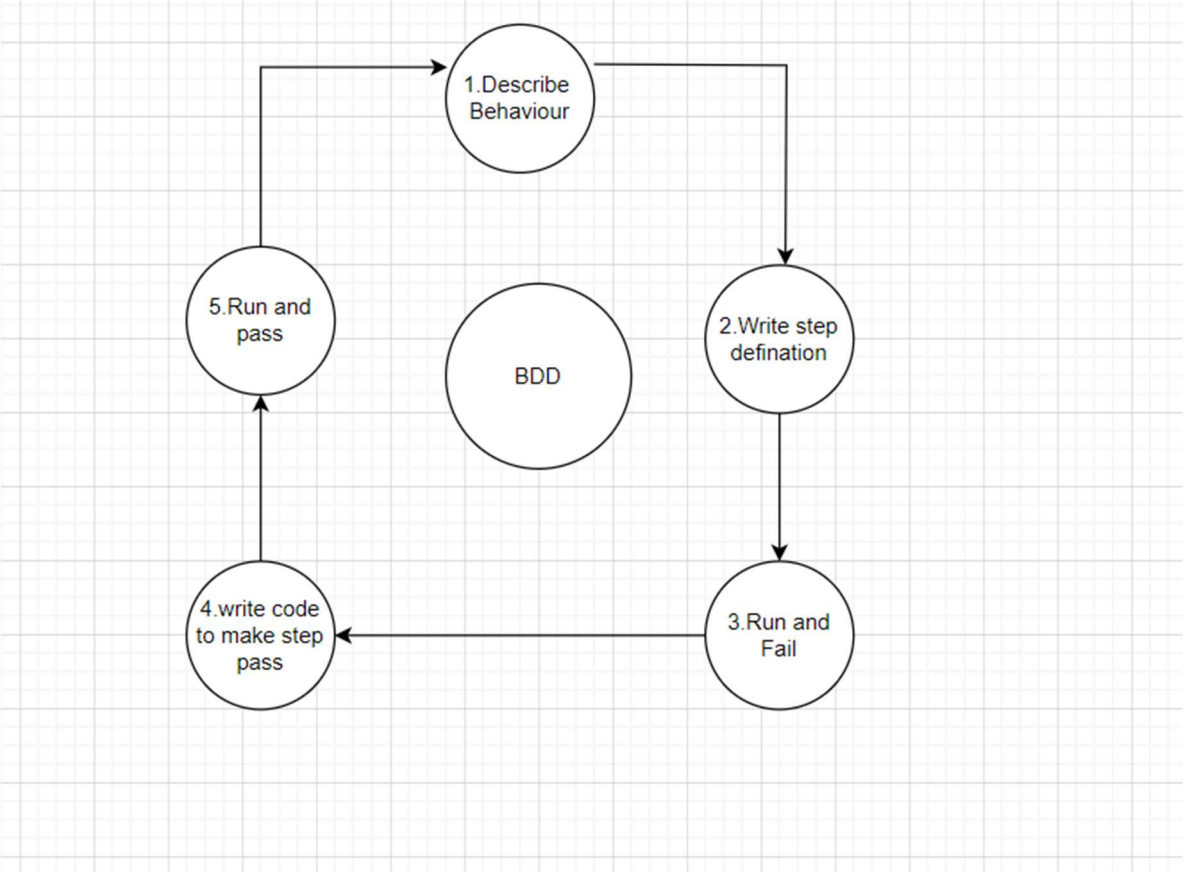
Test-Driven Development (TDD)



Feature Driven Development (FDD):



Behavior driven Development (BDD):



Their unique approaches, benefits, and suitability for different software development contexts

TDD (Test-Driven Development): A development process where tests are written before code.

BDD (Behavior-Driven Development): An extension of TDD that focuses on the behaviour of the software from the end-user perspective.

FDD (Feature-Driven Development): A client-centric, architecture-centric, and pragmatic software process that focuses on delivering tangible, working software repeatedly in a timely manner.

Unique Approaches:

1.Test-Driven Development (TDD):

- Write a Test: Define the expected outcome.
- Run the Test: Verify it fails since the code isn't implemented yet.
- 3.Write Code: Develop the minimal code to pass the test.

- 4.Run the Test Again: Ensure it passes.
- 5. Refactor: Optimise the code.

2.Feature Driven Development (FDD):

- 1.Develop an Overall Model: Outline the system.
- 2.Build a Feature List: Identify all features.
- 3.Plan by Feature: Organise development.
- 4. Design by Feature: Create detailed design.
- 5.Build by Feature: Implement and test.

3. Behavior driven Development (BDD):

- 1.Define User Stories: Describe functionality in user terms.
- 2. Write Scenarios: Use Given-When-Then format to outline behavior.
- 3.Implement Code: Develop based on scenarios.
- 4. Run Scenarios: Ensure code meets the behavior specifications.

Benefits:

TDD:

1. Early Bug Detection: Catches bugs early in development.
2. Improved Code Quality: Leads to cleaner, modular code.
3. Facilitates Refactoring: Safe to refactor without breaking functionality.

BDD:

1. Enhanced Collaboration: Encourages collaboration among all stakeholders.
2. Clear Requirements. Ensures everyone understands the requirements.
3. 3.User-Centric Development: Focuses on delivering what the user needs.

FDD:

1. Scalable Process: Suitable for large projects with many features.
2. 2.Client-Centric: Keeps the client involved and informed.
3. Continuous Progress: Regular feature delivery keeps the project on track.

Suitability for Different Contexts:

TDD:

Best For: Small to medium projects, codebase maintenance, projects needing high reliability.

BDD:

Best For: Projects with complex requirements, teams needing strong collaboration, user-focused applications.

FDD:

Best For: Large-scale projects, projects with many features, teams requiring clear structure and frequent delivery.