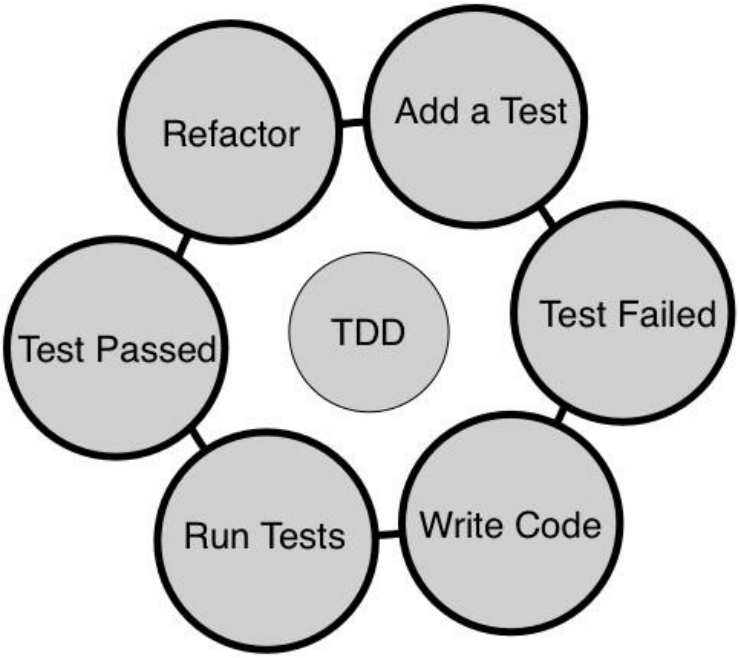
ASSIGNMENT-2

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) Process



1. **Add a Test**:
    - Begin by writing a test case that describes the specific functionality you want to implement.
    - This test case should initially fail because there's no corresponding code yet.

    Benefit:

    By writing tests first, you establish clear expectations for your code, ensuring that it meets the desired requirements.

2. **Test Failed**:
    - After adding the test, run all the existing test cases (including the new one).
    - Since there's no code to make the new test pass, it will fail at this stage.

Benefit:

Identifying failures early allows you to catch potential issues before they propagate further into your codebase.

3. **Write Code**:
   - Develop the minimum amount of code necessary to make the test case pass.
   - Keep the code simple and focused on the specific functionality being tested.

   Benefit:

   - TDD encourages incremental development, leading to more modular and maintainable code.
4. **Run Tests**:
   - Execute all the test cases again, including the new one.
   - If any test fails, revisit the code and make necessary adjustments.

   Benefit:

   Regularly running tests ensures that your code remains functional and aligned with requirements.

5. **Test Passed**:
   - Once all the tests pass, you've successfully implemented the desired functionality.
   - Congratulations! Your code now meets the specified requirements.

   Benefit:

   TDD provides confidence that your code behaves correctly, reducing the risk of introducing defects.

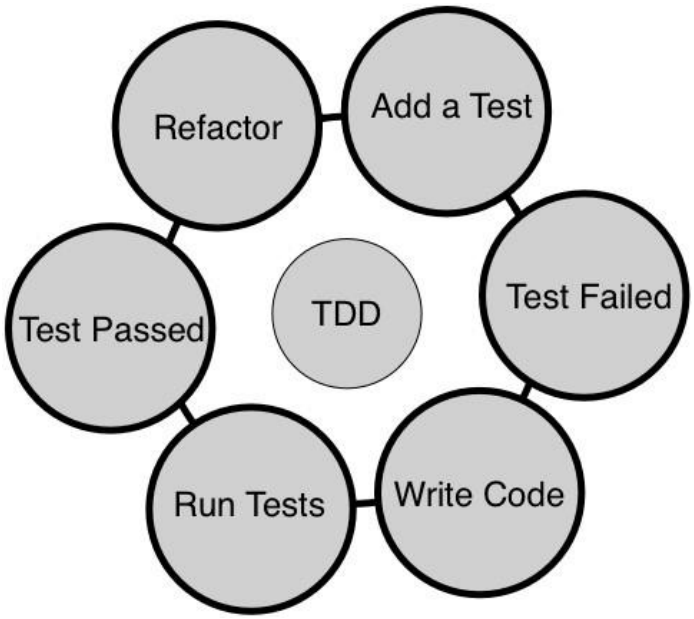
6. **Refactor (Optional)**:
   - After passing the tests, consider refactoring your code.
   - Refactoring involves improving the code's structure, readability, and efficiency without changing its behavior.

   Benefit:

   Refactoring maintains code quality, making it 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.

1. **Test-Driven Development (TDD):**



Approach:

In TDD, developers write tests before writing the actual code. The cycle typically involves three steps:

Red : write a failing test.

Green : write the minimum code to pass the test.

Refactor : improve the code without changing functionality.

Benefits:

- Ensures code correctness by focusing on test coverage.
- Early detection of defects.
- Drives better design decisions.
- Provides a safety net for code changes.
- Clean, refactored codebase.

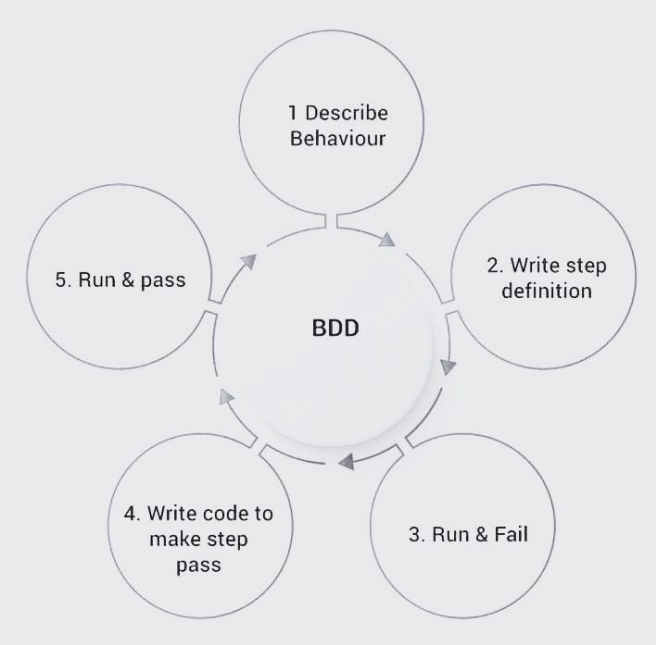
Suitability:

- Ideal for small, well-defined features.
- Commonly used in agile development.

Examples:

Financial systems, safety-critical applications.

2. Behaviour-Driven Development(BDD):



Behaviour-Driven Development (BDD):

Approach:

- BDD extends TDD by emphasizing behavior over implementation details.
- It uses a natural language syntax (Given-When-Then) to describe system behavior.

Benefits:

- Improved communication among stakeholders.
- Improves communication by using plain language.
- Clear understanding of requirements.
- Focuses on user needs.

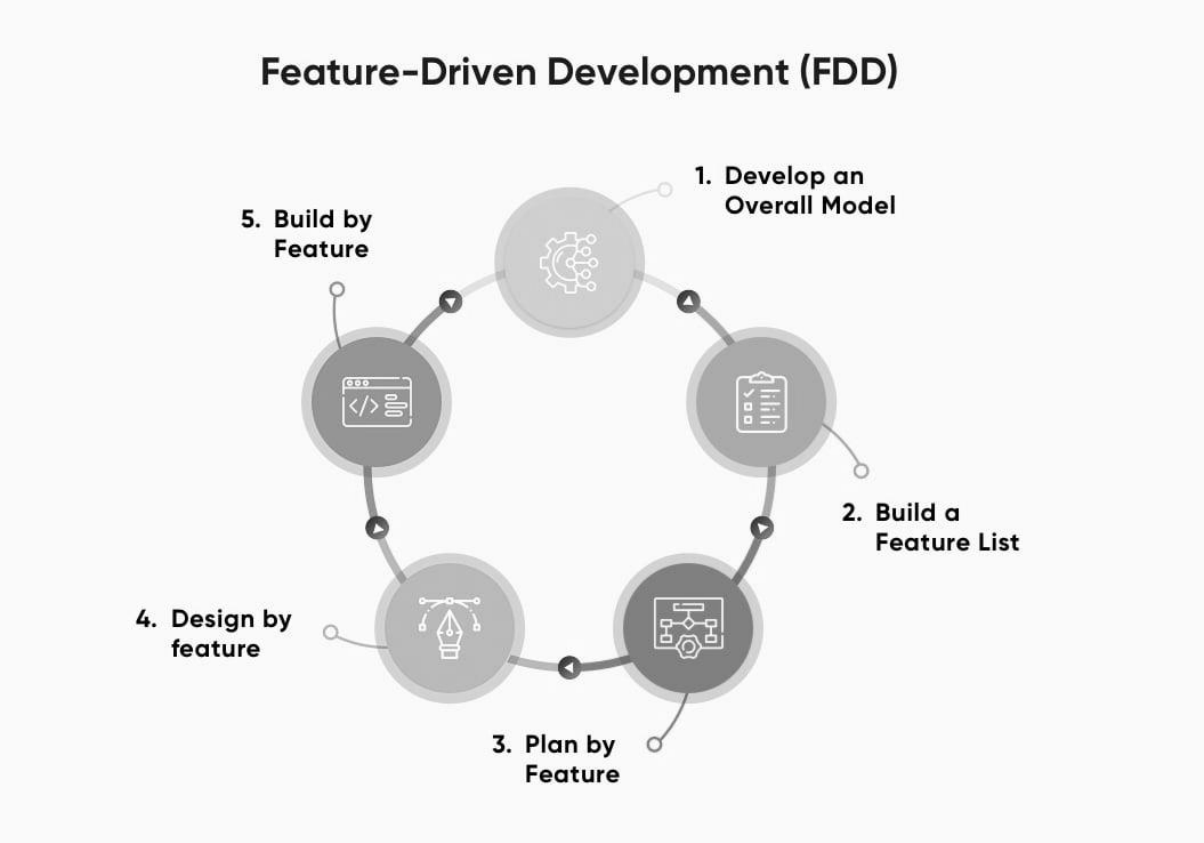
Suitability:

- Well-suited for complex systems with intricate interactions.
- Useful for acceptance testing and end-to-end scenarios.

Examples:

E-commerce platforms, customer-facing applications.

3.Feature-Driven Development (FDD):



Approach:

- FDD focuses on building features incrementally.
- It involves creating a feature list, designing features, and then implementing them.

Benefits:

- Provides a structured approach to feature development.
- Emphasizes domain modeling and design.
- Scales well for large projects.

Suitability:

- Best for large, team-based projects.
- Works well when features can be modularized.

Examples:

Enterprise systems, multi-feature software projects.