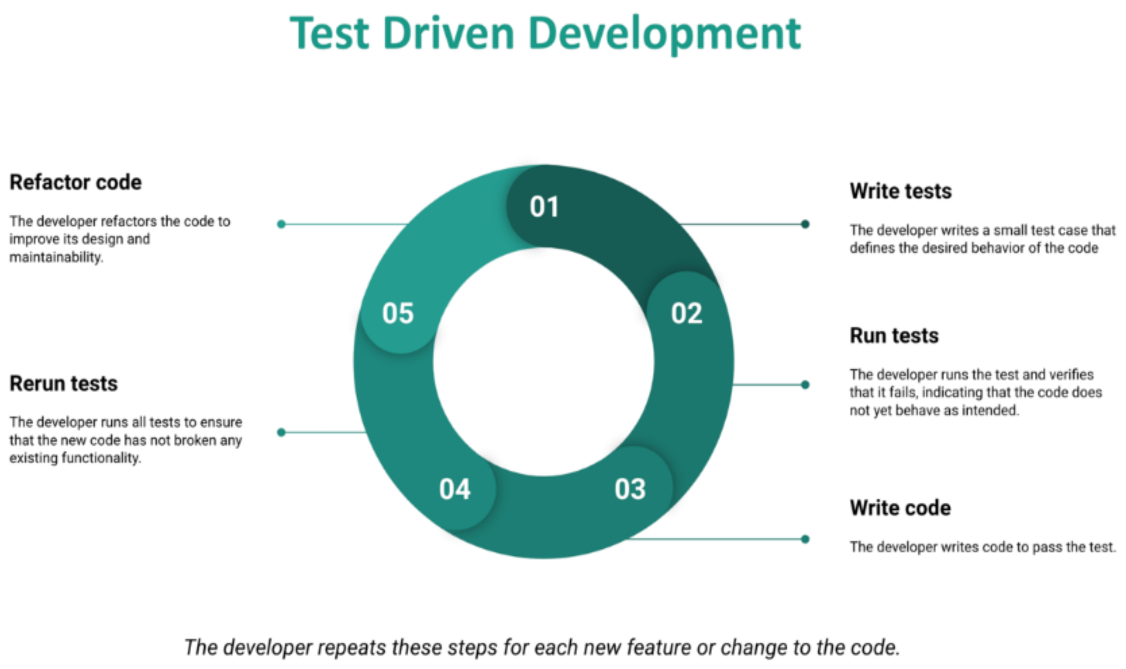
Day-3 (Modern Development Methodologies)

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.

Solution:

Test-driven development (TDD) is a software development process that involves writing tests for your code before you write the code. The main goal of TDD is to ensure that the code is **correct**, **reliable**, and **maintainable**. The TDD process involves writing a small test case that defines the desired behavior of the code, writing code to pass that test, and then refactoring the code to improve its design and maintainability. This cycle repeats with further improvements and new features until the product is complete.

Infographic representation of TDD:


Test Driven Development
Refactor code
The developer refactors the code to improve its design and maintainability.
Rerun tests
The developer runs all tests to ensure that the new code has not broken any existing functionality.
01
02
03
04
05
Write tests
The developer writes a small test case that defines the desired behavior of the code
Run tests
The developer runs the test and verifies that it fails, indicating that the code does not yet behave as intended.
Write code
The developer writes code to pass the test.
The developer repeats these steps for each new feature or change to the code.

TDD is an iterative process that involves the following sequence of steps:

Step 1. Write a test:

Description: start by writing a test for a feature or functionality that defines the expected behavior of a feature or functionality.

Purpose: To specify the requirements of the feature and ensure it meets the desired outcome.

Step 2. Run the test:

Description: Execute the test to verify that it fails because the feature has not been implemented yet.

Purpose: To confirm that the test is valid and the feature is not implemented yet.

Step 3. Write code-

Description: Write the minimum amount of code that is necessary to make the test pass.

Purpose: To implement the feature in a way that satisfies the test criteria, ensuring the desired behavior is achieved.

Step 4. Rerun test-

Description: Run the test again to verify that it now passes with the newly written code.

Purpose: To verify that the code implementation meets the requirements specified in the test.

Step 5. Refactor code:

Description: Refactor the code to improve its design, readability, and maintainability without changing its behavior.

Purpose: To ensure the code is clean, efficient, and easy to maintain, while preserving its functionality.

Benefits of TDD-

Bug Reduction: By writing tests before code, developers catch bugs early in the development process, reducing the chance of introducing defects.

Improved Code Design: TDD encourages modular, and well-structured code as developers focus on writing testable code that fulfills specific requirements.

Software Reliability: With a suite of automated tests, developers can confidently make changes, knowing that if a test fails, they have immediate feedback on what needs fixing.

TDD Fosters Software Reliability

Continuous Refactoring: Continuous refactoring in TDD keeps the codebase clean and easy to maintain contributing to software reliability.

Clear Documentation: Tests serve as living documentation, outlining the expected behavior of the code, which enhances software reliability.

Faster Debugging: TDD provides a structured approach to debugging, as failing tests point out the source of issues quickly, aiding in maintaining software reliability.

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.

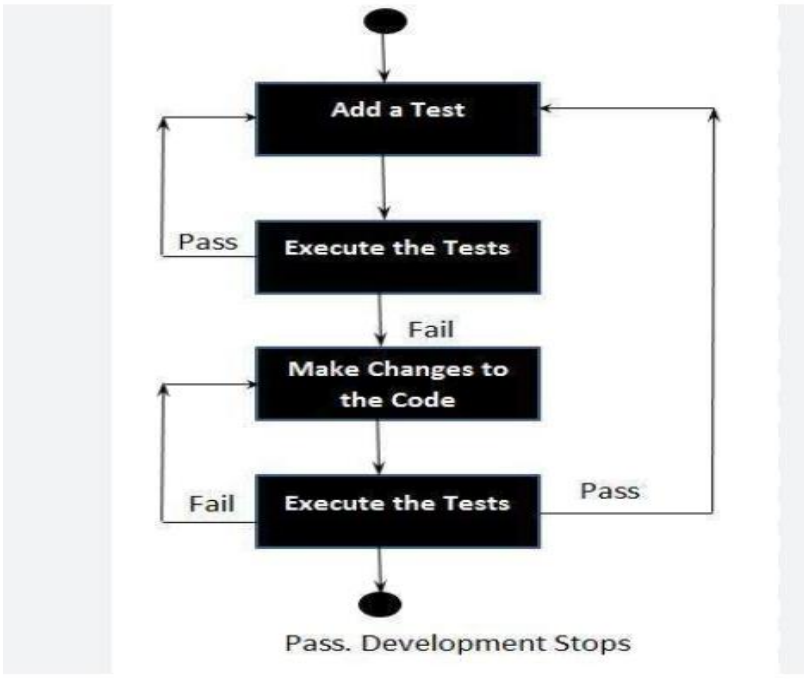
Solution:

Here is a comparison of **TDD**, **BDD**, and **FDD** methodologies:

1. **Test-Driven Development (TDD)**
    - focuses on writing tests before writing code
    - Tests are written from a developer's perspective to verify code correctness
    - Keeps code clean and manageable through iterative refactoring
    - Best for projects with clear requirements and a skilled technical team.

    Benefits:
    - Explores bugs or errors very early
    - Faster feedback
    - Enhances the productivity
    - Reduces the time required for rework

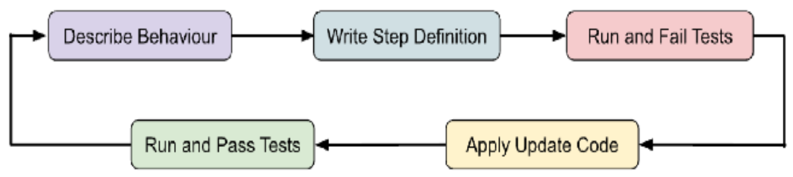
    Suitability:
    - Suitable for projects where code reliability and functionality are crucial
    - Ideal for agile development ecosystems where automated test scripts are written before functional pieces of code



2. **Behavior-Driven Development (BDD)**
    - Collaboration among developers, testers, and business stakeholders is key.
    - Scenarios are written using easy words to explain how the system should behave.
    - Tests are made from the scenarios to check if the system works the way users want.
    - Ideal for projects with changing needs and a diverse team with various skills.

    <u>Benefits:</u>
    - Helps reach a wider audience through the usage of non-technical language
    - BDD is a cost-effective technique
    - Improved collaboration between stakeholders
    - Alignment between development and business goals.

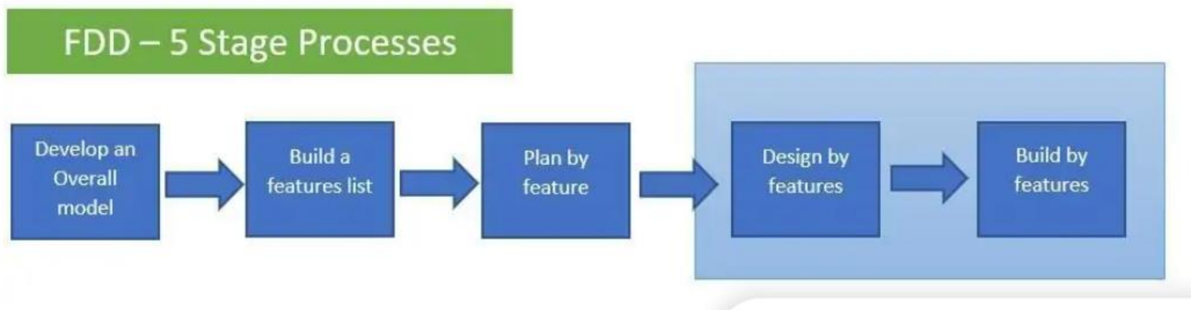
    <u>Suitability:</u>
    - Suitable for projects where collaboration between developers and business stakeholders is essential
    - Suitable for projects with complex business logic and involving multiple stakeholders.



3. **Feature-Driven Development (FDD)**
    - Iterative and incremental approach that focuses on delivering tangible, working software.
    - Starts with an overall model and then adds features one by one.
    - Emphasizes frequent code inspections and testing throughout the development process
    - Best for big, complicated projects with clear plans and a big team.

    <u>Benefits:</u>
    - Consists of a set of best practices that enhance the development process
    - Clear focus on features of product
    - Efficient resource allocation
    - Enhance client satisfaction

    <u>Suitability:</u>
    - Suitable for client centric projects
    - Suitable for projects where a structured approach to development is required


FDD – 5 Stage Processes
Develop an Overall model
Build a features list
Plan by feature
Design by features
Build by features