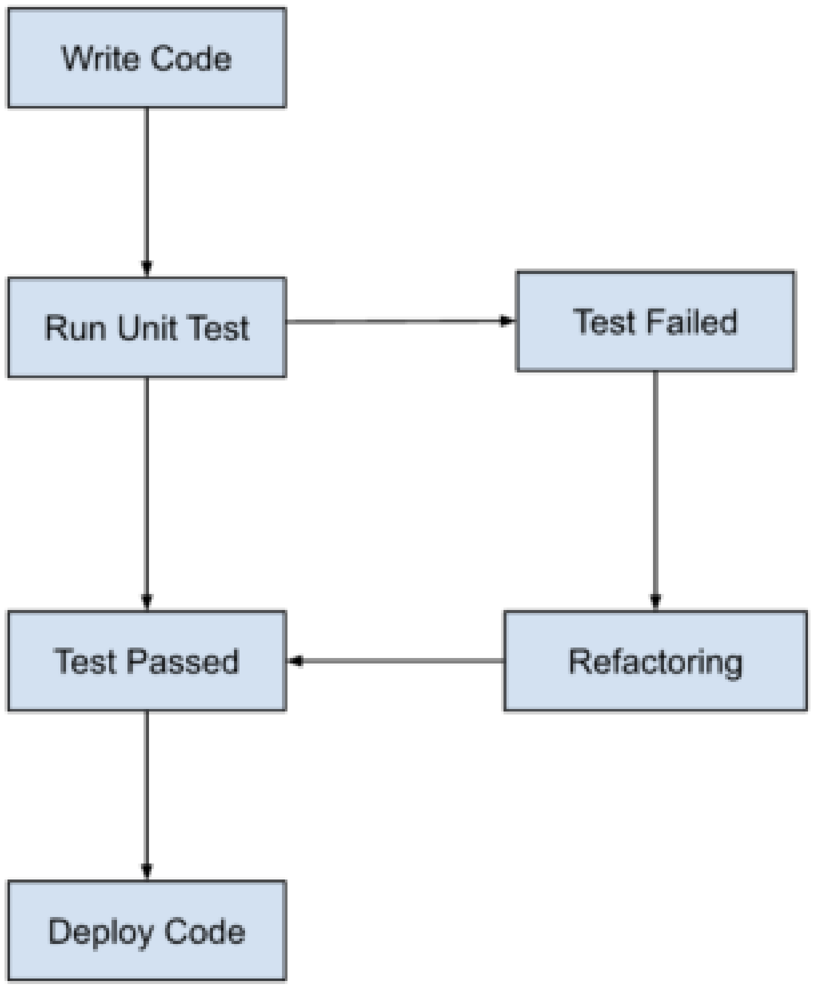
Assignment 1: Create an infographic illustrating the Test-Driven Development (TDD) process. Highlight steps like writing tests before code, benefits such as bug reduction, and how it fosters software reliability.



Create precise tests: Developers need to create exact unit tests to verify the functionality of specific features. They must ensure that the test compiles so that it can execute. In most cases, the test is bound to fail. This is a meaningful failure as developers create compact tests based on their assumptions of how the feature will behave.

Correcting the Code: Once a test fails, developers must make the minimal changes required to update the code to run successfully when re-executed.

Refactor the Code: Once the test runs successfully, check for redundancy or any possible code optimizations to enhance overall performance. Ensure that refactoring does not affect the external behavior of the program.

Benefits of Test Driven Development (TDD)

-Fosters the creation of optimized code.

-It helps developers better analyze and understand client requirements and request clarity when not adequately defined.

-Adding and testing new functionalities become much easier in the latter stages of development.

-Test coverage under TDD is much higher compared to conventional development models. The TDD focuses on creating tests for each functionality right from the beginning.

-It enhances the productivity of the developer and leads to the development of a codebase that is flexible and easy to maintain.

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.

Aspect	**TDD**	**BDD**	**FDD**
Focus	code quality through testing	User behavior and acceptance criteria	Feature delivery and iterative development
Testing Approach	Test code first, then implement	Behavioral scenarios based on user stories	Feature validation through acceptance tests
Tooling	Testing frameworks (JUnit, etc.)	BDD frameworks (Cucumber, SpecFlow)	Not framework-dependent
Documentation	Code-focused	Behavior-focused	Feature-centric
Communication	Within dev team	Across stakeholders	Within dev team, domain experts
Suitability	Agile environments, code quality focus	Collaborative projects, clarity	Medium to large-scale projects
Adaptability	Excellent for isolated units	Good for user stories, scenarios	Efficient with structured features