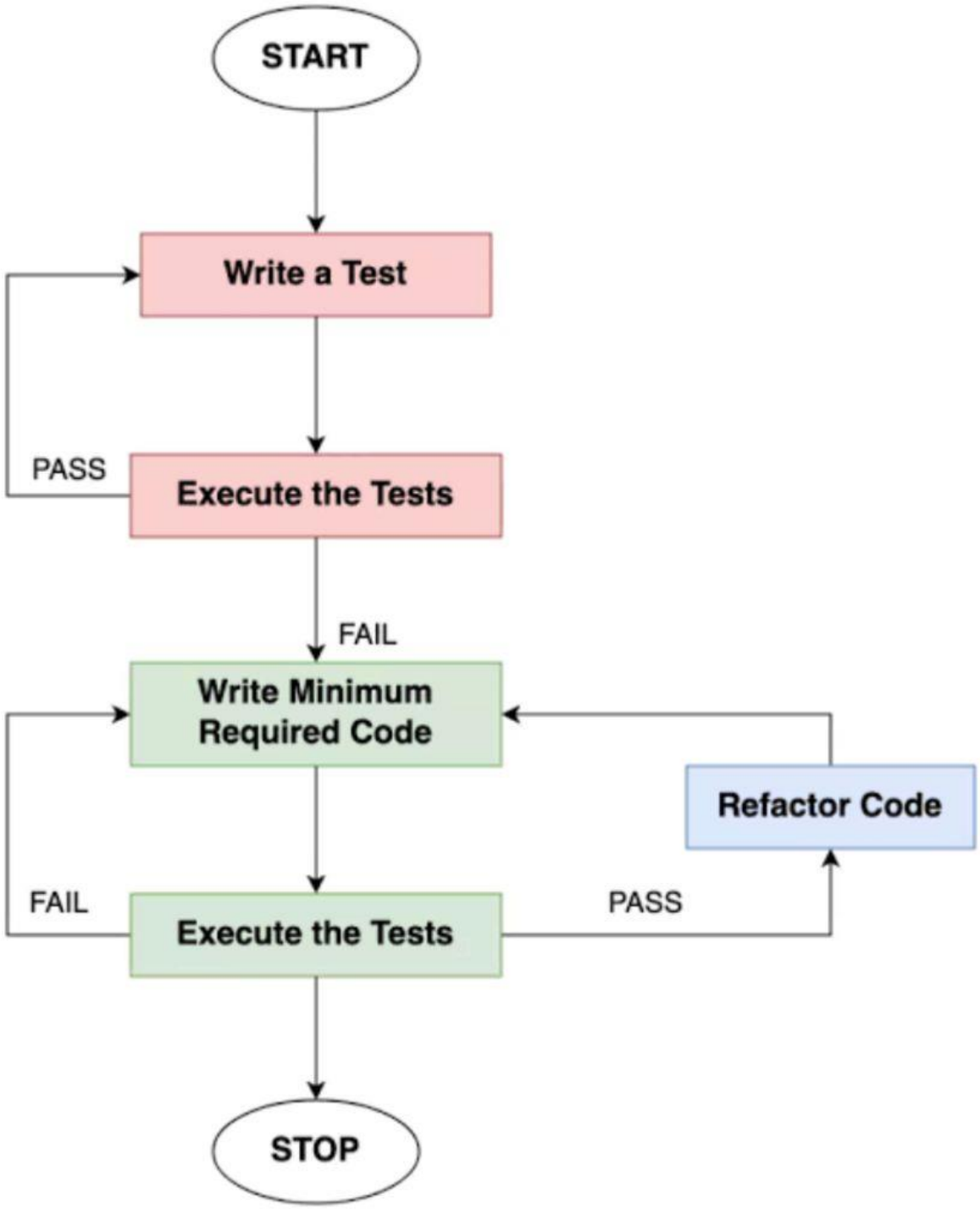
Modern Development Methodologies

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.

Test-Driven Development (TDD) in Action:



Start with Testing:

Write tests before writing any code.

Tests define what the code should do.

Write the Minimum Code:

Write the simplest code to make the test pass.

Focus on making the test pass, nothing more.

Refactor and Repeat:

Refactor code to improve its quality without changing functionality.

Repeat the process for each new feature or change.

Benefits of TDD:

Fewer Bugs: Catch issues early, reducing bugs in the final product.

Improved Code Quality: Encourages cleaner, more maintainable code.

Faster Development: Saves time in the long run with fewer bugs and easier maintenance.

Increased Confidence: Comprehensive tests provide confidence in code changes.

TDD is Reliable:

Continuous Testing: Ensures code is always tested, reducing the chance of regressions.

Early Issue Detection: Identifies problems before they become bigger issues.

Comprehensive Coverage: Tests cover all aspects of the code, enhancing reliability.

Conclusion:

TDD is a proven method for building reliable software by focusing on tests first.

It reduces bugs, improves code quality, and fosters 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.

Comparative Overview: TDD vs. BDD vs. FDD

Key Differences: TDD vs BDD

Parameters	TDD	BDD
Definition	TDD is a development technique that focuses more on the implementation of a feature	BDD is a development technique that focuses on the system's behavior
Participants	Developer	Developers, Customers, QAs
Language used	Written in a language similar to the one used for feature development (Eg. Java, Python, etc)	Simple English, (Gherkin)
Main Focus	Unit Tests	Understanding Requirements
Tools used	JDave, Cucumber, JBehave, SpecFlow, BeanSpec, Gherkin Concordian, FitNesse, Jest, Jasmine, Protractor	Gherkin, Dave, Cucumber, JBehave, Spec Flow, BeanSpec, Concordian, Cypress

Test-Driven Development (TDD):

Approach:

Write tests before coding.

Focus on small, iterative cycles.

Tests drive code design and implementation.

Benefits:

Early bug detection.

Improved code quality.

Faster development.

Enhanced confidence in changes.

Suitability:

Clear project specifications.

Small to medium-sized teams.

Projects requiring extensive test coverage.

Behavior-Driven Development (BDD):

Approach:

Focus on system behavior from user perspective.

Define requirements using a common language (e.g., Gherkin).

Foster collaboration among developers, testers, and stakeholders.

Benefits:

Improved communication.

Clearer understanding of requirements.

Better test coverage.

Enhanced collaboration across teams.

Suitability:

Projects with complex business logic.

Diverse skill sets within team.

Evolving project requirements.

Feature-Driven Development (FDD):

Approach:

Develop features incrementally.

Emphasize frequent releases.

Focus on domain object modeling and feature lists.

Benefits:

Rapid feature development.

Transparent project progress tracking.

Clear visibility into feature implementation.

Scalable for large projects.

Suitability:

Large, enterprise-level projects.

Specialized team roles.

Structured and disciplined project requirements