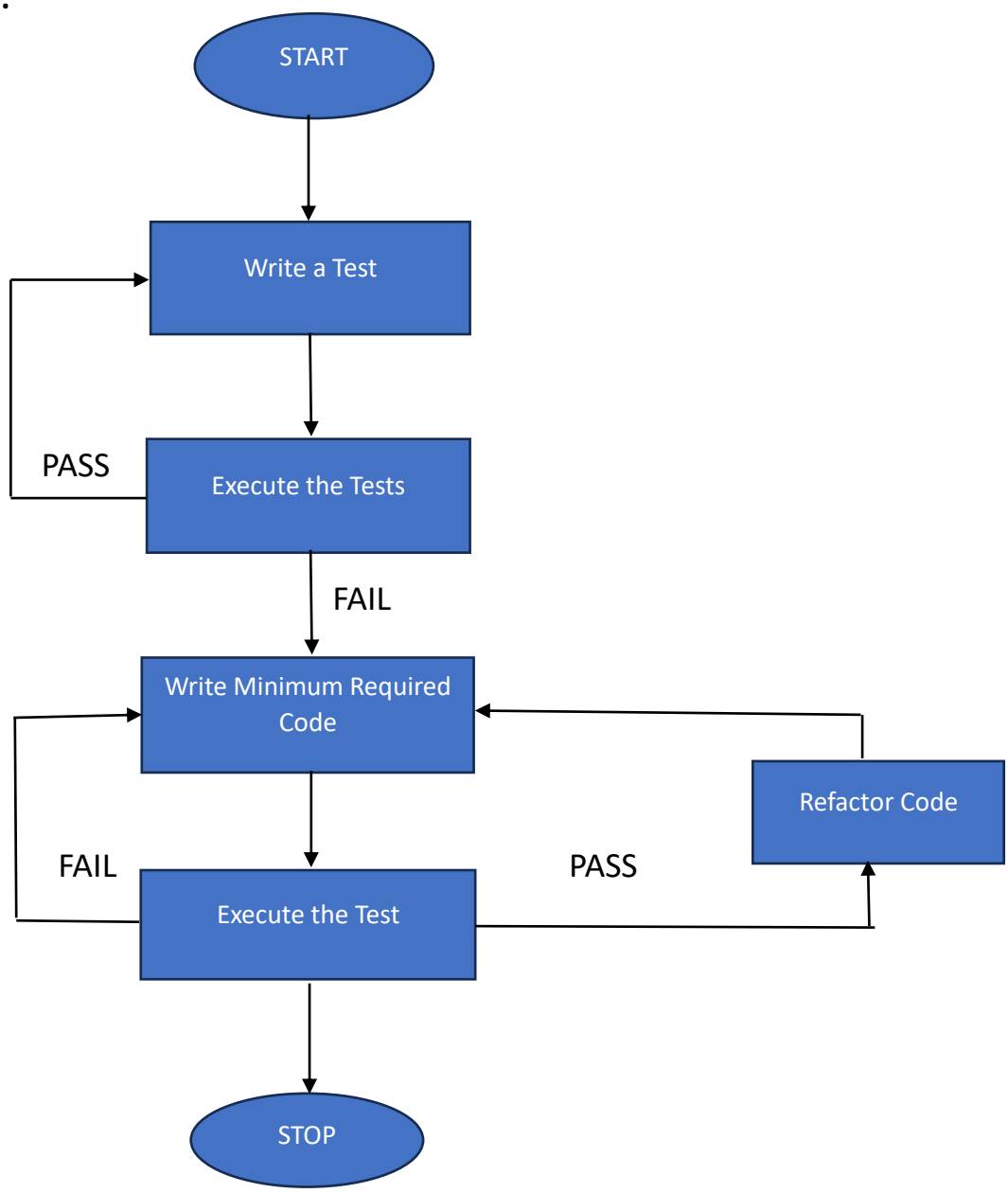
Assignment- Day-3

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:



Steps like writing tests before code

1.Write a Test: Begin by writing tests that describe the desired behavior of a small part of your software.

2. Execute the Tests (Expect Failures): Execute the tests to ensure they fail. This confirms that your initial tests are valid and the feature isn't accidentally working.

3. Write Minimum Required Code: Implement the simplest code to make the failing tests pass. This ensures you are only adding code that's necessary.

4. Execute the Tests Again (Expect Pass): Execute the tests again. They should now pass, demonstrating that your new code works as expected.

5. Refactor Code: Clean up and optimize the code without changing its functionality. Refactoring helps maintain a clean and maintainable codebase.

6. Repeat: Cycle through steps 1-5 for each new feature or change.

Benefits of TDD:

Bug Reduction: Catch bugs early in the development cycle, reducing the cost and effort of fixing them later.

Improved Code Quality: Encourages writing modular, well-structured, and maintainable code.

Faster Development: Efficiently iterate on features and confidently refactor existing code without fear of breaking functionality.

Software Reliability: Ensures that the software behaves as intended by validating each aspect through automated tests.

How TDD Fosters Software Reliability:

Complete Test Coverage: Every piece of code is tested, minimizing the chance of unexpected behavior.

Continuous Validation: Constantly validates the correctness of the software, resulting in a more reliable product.

Encourages Modularity: Promotes writing code in smaller, manageable units, enhancing reliability and ease of maintenance.

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:

Test-Driven Development (TDD)

Approach:

- Write tests before writing code.
- Focus on small units (functions or classes).

Benefits:

- Early bug detection.
- Improved code quality.
- Incremental development.

Suitability:

- Ideal for Agile environments.
- Best for applications with well-defined requirements.

Behavior-Driven Development (BDD)

Approach:

- Define behavior using natural language.
- Focus on user scenarios and acceptance criteria.

Benefits:

- Encourages collaboration between developers, testers, and non-technical stakeholders.
- Enhances communication and understanding of requirements.

Suitability:

- Well-suited for complex systems with evolving requirements.
- Useful for projects emphasizing business value and user interaction.

Feature-Driven Development (FDD)

Approach:

- Break down development into features.
- Emphasizes domain object modeling and iterative development.

Benefits:

- Clear and structured development process.
- Efficient for managing large teams and projects.

Suitability:

- Effective for enterprise-level projects.
- Works well with distributed development teams.

Aspect	**TDD**	**BDD**	**FDD**
Focus	Write tests before code to drive development	Describe system behavior in readable format.	Deliver features incrementally, focusing on design.
Collaboration	Developers and testers collaborate for thorough testing.	Collaboration among developers, testers, and business stakeholders.	Collaboration among team members for feature delivery.
Language	Programming language testing frameworks.	Natural language syntax.	May vary
Iterative Development	Yes	Yes	Yes
Emphasis on Behavior	Less emphasis	High emphasis	Moderate emphasis
Domain Modeling	Optional	May involve	Strong emphasis

Design by Feature	Optional	May follow	Yes
Stakeholder Engagement	Less direct engagement	Direct engagement	Direct engagement throughout.

Use visuals: