DAY-2 SDLC

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.

Infographic Title: Understanding Test-Driven Development (TDD)

Section 1: What is TDD?

- **Definition:** Test-Driven Development (TDD) is a software development process where tests are written before code to define desired functionality.

Section 2: TDD Process

1. **Write a Test**
    - Create a test for a new feature or functionality.
    - The test should initially fail, as the feature is not yet implemented.
    - **Visual:** Icon of a pencil and paper with a failed test mark.
2. **Write the Minimum Code**
    - Write just enough code to make the test pass.
    - Focus on functionality, not optimization.
    - **Visual:** Icon of a coder at a computer.
3. **Run the Tests**
    - Execute all tests to ensure the new code causes the test to pass.
    - **Visual:** Icon of a play button or a test execution process.
4. **Refactor the Code**
    - Improve the code's structure without changing its behavior.
    - Ensure the test still passes after refactoring.
    - **Visual:** Icon of a gear and wrench.
5. **Repeat**
    - Continue the cycle with the next feature or functionality.
    - **Visual:** Circular arrow indicating repetition.

Section 3: Benefits of TDD

- **Bug Reduction:** Early detection and fixing of bugs.
  - **Visual:** Bug icon with a slash through it.
- **Improved Code Quality:** Encourages clean, modular, and maintainable code.
  - **Visual:** Ribbon or badge indicating quality.
- **Reliability:** Ensures new changes don't break existing functionality.
  - **Visual:** Shield or lock icon for reliability.
- **Faster Development:** Reduces time spent on debugging later.
  - **Visual:** Stopwatch or clock icon.

Section 4: TDD Fosters Software Reliability

- **Continuous Testing:** Ensures the software behaves as expected at all times.
  - **Visual:** Chain or continuous loop icon.
- **Documentation:** Tests serve as documentation for the codebase.
  - **Visual:** Book or document icon.
- **Confidence:** Developers can make changes with confidence, knowing tests will catch regressions.
  - **Visual:** Thumbs up or checkmark icon.

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.

Infographic Title: Comparing TDD, BDD, and FDD

Section 1: Definitions

- **TDD (Test-Driven Development):**
  - **Definition:** Write tests before writing the code to ensure functionality.
  - **Visual:** Test icon with a forward arrow to code.
- **BDD (Behavior-Driven Development):**
  - **Definition:** Extends TDD by writing tests in a natural language that non-developers can understand.

    - **Visual:** Speech bubble or dialogue icon.
- **FDD (Feature-Driven Development):**
    - **Definition:** Focuses on building and delivering features in an iterative cycle.
    - **Visual:** Feature or checklist icon.

Section 2: Unique Approaches

- **TDD:**
    - Write failing test -> Write code to pass test -> Refactor code.
    - **Visual:** Circular flowchart with steps.
- **BDD:**
    - Define behavior in Gherkin language -> Write tests -> Implement code to pass tests.
    - **Visual:** Script or story icon with arrows to tests and code.
- **FDD:**
    - Develop an overall model -> Build a feature list -> Plan by feature -> Design by feature -> Build by feature.
    - **Visual:** Step-by-step ladder or hierarchy chart.

Section 3: Benefits

- **TDD:**
    - **Bug Reduction**
    - **Code Quality**
    - **Reliability**
    - **Visual:** Icons for bug, quality badge, and reliability shield.
- **BDD:**
    - **Improved Communication:** Clear communication between developers, testers, and business stakeholders.
    - **Better Understanding:** Natural language tests make requirements clearer.
    - **Visual:** Icons for communication (speech bubbles) and understanding (light bulb).
- **FDD:**
    - **Scalability:** Suitable for large projects.
    - **Predictability:** Regular, predictable delivery of features.

    - **Visual:** Icons for growth (graph) and predictability (calendar).

Section 4: Suitability

- **TDD:**
    - Best for developers who need a clear specification and like iterative testing.
    - **Visual:** Developer icon with a checklist.
- **BDD:**
    - Ideal for teams with strong collaboration between technical and non-technical members.
    - **Visual:** Team or collaboration icon.
- **FDD:**
    - Suitable for large-scale projects with clear feature requirements.
    - **Visual:** Large project or skyscraper icon.

Conclusion: Choosing the Right Methodology

- **TDD:** Choose for bug-free, reliable code with rigorous testing.
- **BDD:** Choose for enhanced communication and clear requirements.
- **FDD:** Choose for managing large projects with continuous feature delivery.