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 functionality before we how if fosters software reliability

Solution: Test-Driven Development (TDD) Process

1. Write Test Cases
    - Write automated test cases based on desired functionality.
    - Tests focus on specific behaviors and edge cases.
2. Run Tests (Fail)
    - Run tests initially; they should fail as no code exists yet.
    - Failure confirms that tests are accurately detecting absence of functionality.
3. Write Code
    - Write code to pass failing tests.
    - Focus on simplicity and functionality to pass tests.
4. Run Tests (Pass)
    - Run tests again; they should pass after writing code.
    - Passing tests indicate successful implementation of functionality.
5. Refactor Code
    - Refactor code to improve design, performance, and readability.
    - Ensure tests still pass after refactoring.

Benefits of TDD:

- Bug Reduction: Catch bugs early in the development cycle, reducing debugging efforts later.
- Improved Code Quality: Encourages modular, well-structured code with clear functionality.
- Increased Productivity: Faster debugging and iteration cycles lead to faster development.
- Enhanced Reliability: Rigorous testing ensures software meets requirements and functions reliably.
- Customer Satisfaction: Deliver high-quality, bug-free software that meets customer expectations.

How TDD Fosters Software Reliability:

1. Prevents Regression: Tests catch regressions, ensuring new changes don't break existing functionality.

2. Ensures Functionality: Tests validate that each component and feature works as intended.
3. Encourages Confidence: Developers and stakeholders gain confidence in software reliability.
4. Continuous Validation: Ongoing tests validate software integrity throughout development and maintenance phases.

Assignment 2: Produce a comparative infographic of TDD, BDD and FDD methodologies development contexts. Use visuals to enhance understanding

Comparative Infographic: TDD vs BDD vs FDD

1. Test-Driven Development (TDD):
- Focus: Unit testing individual components.
- Process:
    - Write tests before code.
    - Red-Green-Refactor cycle.
    - Emphasizes small iterations and continuous testing.
2. Behavior-Driven Development (BDD):
- Focus: Collaboration between developers, testers, and stakeholders.
- Process:
    - Define behavior using Given-When-Then scenarios.
    - Uses natural language specifications (e.g., Gherkin).
    - Promotes understanding of user requirements and behavior.
3. Feature-Driven Development (FDD):
- Focus: Feature-centric development approach.
- Process:
    - Develop features incrementally.
    - Emphasizes domain modeling and design.
    - Iterative development with frequent builds and releases.

Key Differences:

- Testing Focus:
    - TDD focuses on unit testing code functionality.
    - BDD emphasizes testing behavior and user interactions.
    - FDD centers around developing and delivering features.

- Collaboration:
    - TDD involves developers primarily in writing and testing code.
    - BDD promotes collaboration between developers, testers, and stakeholders.
    - FDD encourages teamwork among cross-functional teams for feature delivery.
- Documentation:
    - TDD documentation includes test cases and code.
    - BDD documentation consists of behavior scenarios and user stories.
    - FDD documentation covers feature specifications and progress.

Benefits:

- TDD: Bug reduction, improved code quality, faster debugging.
- BDD: Enhanced collaboration, better understanding of requirements, user-centric development.
- FDD: Feature-focused development, iterative progress, adaptable to changing requirements.

Applicability:

- TDD: Ideal for small-scale projects, developer-centric teams.
- BDD: Suitable for projects with complex behavior, collaboration-driven environments.
- FDD: Best for large-scale projects, feature-based development, and cross-functional teams.