Infographic Title: The Test-Driven Development (TDD) Process

1. Write Test: Developers start by writing a failing test based on the requirements of the feature or functionality they are implementing.

2. Code Implementation: Write the simplest code to make the failing test pass. This step focuses solely on making the test pass, not on writing perfect or complete code.

3. Refactor Code: Once the test passes, refactor the code to improve its structure, readability, and efficiency while ensuring all tests still pass.

4. Repeat: Repeat the process for each new feature or functionality, cycling through writing failing tests, coding, and refactoring.

Benefits of TDD:

1. Bug Reduction: By writing tests before implementing code, developers catch bugs early in the development process, reducing the likelihood of bugs reaching production.

2. Improved Reliability: TDD fosters software reliability by ensuring that each piece of code is thoroughly tested before it is considered complete, leading to more robust and stable software.

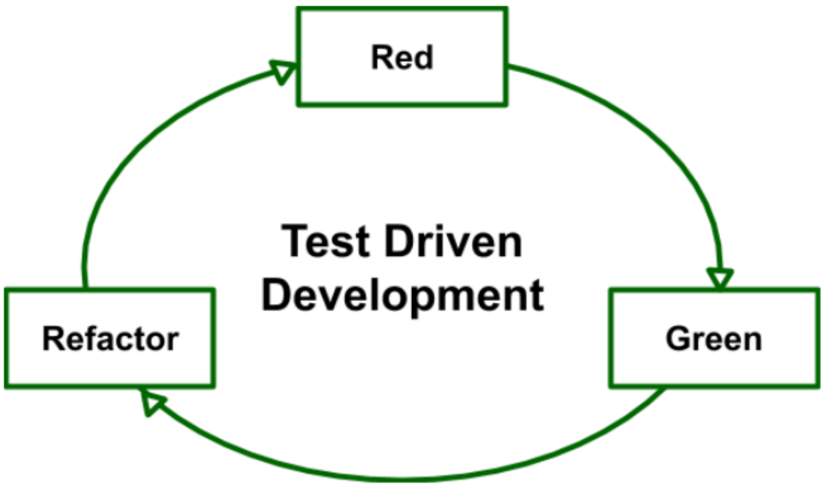
3. Better Design: TDD encourages developers to write modular, loosely-coupled code, leading to better overall design and easier maintenance.

4. Increased Confidence: Developers have greater confidence in their codebase as it is continuously validated by a comprehensive suite of automated tests.

5. Faster Development: While TDD may seem to slow down initial development, it often results in faster overall development time due to reduced debugging and rework.

Infographic of TDD, BDD, and FDD methodologies:

➢ TDD (Test-Driven Development)



- Approach: Write automated tests before writing code

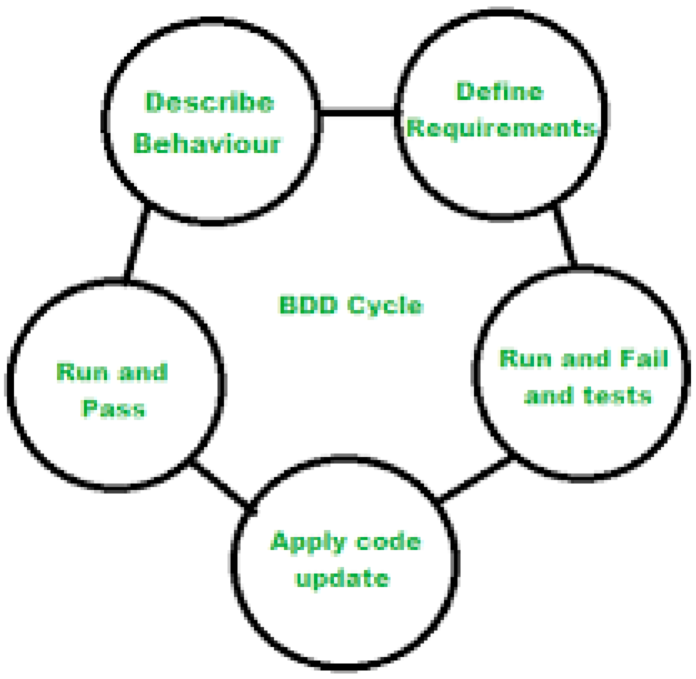
- Benefits:

- Ensures code quality and reliability
- Reduces bugs and debugging time
- Promotes refactoring and clean code

- Suitable for:

- Small to medium-sized projects
- Teams with experienced developers
- Projects requiring high code quality

➢ BDD (Behavior-Driven Development)



- **Approach:** Define behavior through scenarios and examples

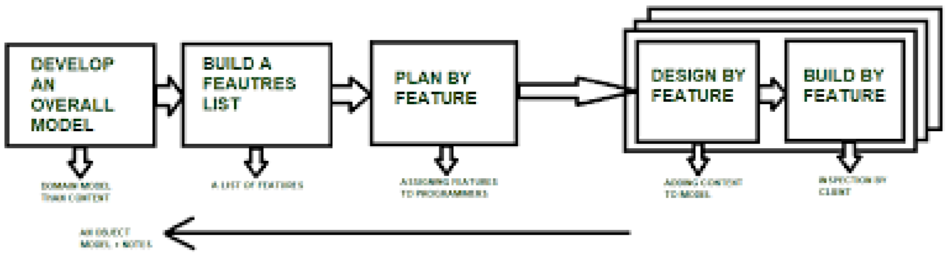
- **Benefits:**

- Improves communication and collaboration
- Reduces misunderstandings and misinterpretations
- Focuses on user needs and acceptance criteria

- **Suitable for:**

- Agile teams and projects
- Projects with complex business logic
- Teams with diverse skill levels

➢ FDD (Feature-Driven Development)



- Approach: Develop features incrementally and iteratively

- Benefits:

- Prioritizes features based on business value
- Reduces complexity and technical debt
- Improves team collaboration and productivity

- Suitable for:

- Large and complex projects
- Teams with varying skill levels
- Projects requiring rapid delivery