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.

Test-Driven Development (TDD)

Definition: TDD is a software development approach where tests are written before writing the actual code. The development process revolves around creating automated tests to validate code functionality.

Key Concepts:

1. **Test-First Approach:** Write a failing test that defines a new function or improvement.

2. **Code Implementation:** Write the minimum amount of code necessary to pass the test.

3. **Refactoring:** Improve the code without changing its functionality, ensuring it remains clean and maintainable.

Benefits:

- **Early Bug Detection:** Bugs are caught early in the development cycle, reducing the cost of fixing them.

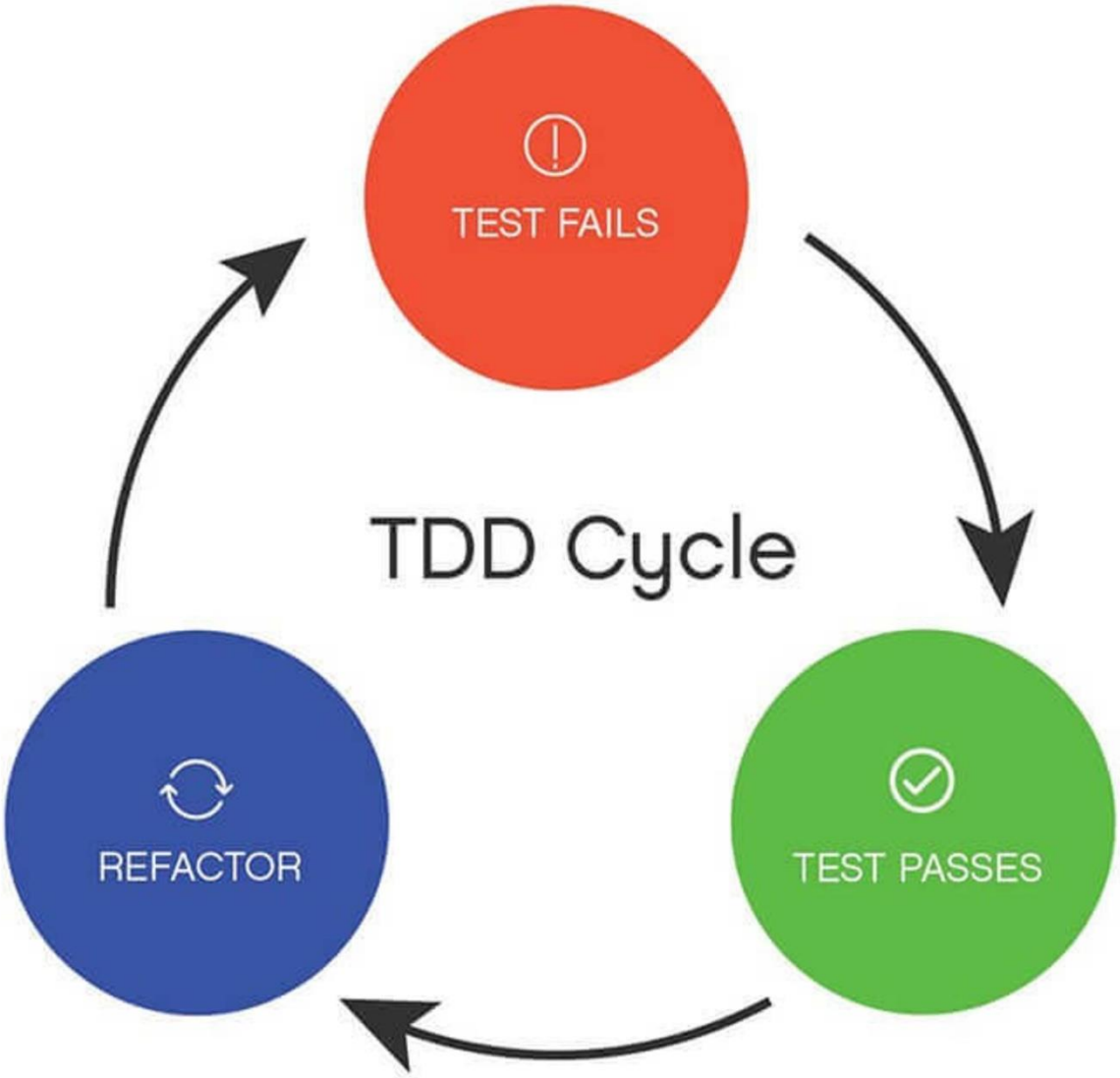
- **Code Reliability:** Codebase becomes more reliable as it's continuously tested against defined requirements.

- **Faster Development Cycles:** TDD can lead to faster development cycles due to less time spent on debugging.

Suitability:

- **Dynamic Requirements:** Best for projects that need to adapt to changing needs or requirements over time.

- **Small to Medium-Sized Teams:** Works well for teams where developers and testers can collaborate closely and communicate effectively.



Behavior-Driven Development (BDD)

Definition: BDD focuses on defining software behaviors using natural language specifications that are understandable by all stakeholders, including developers, testers, and business analysts.

Key Concepts:

1. **Given-When-Then Scenarios:** Define software behaviors using "Given" (preconditions), "When" (actions), and "Then" (expected outcomes) scenarios.

2. **Collaboration:** Encourages collaboration between cross-functional teams to ensure clear understanding of requirements.

3. **Test Automation:** Tests are automated to validate behaviors and ensure they are implemented correctly.

Benefits:

- **Improved Collaboration:** Facilitates communication and collaboration between technical and non-technical team members.

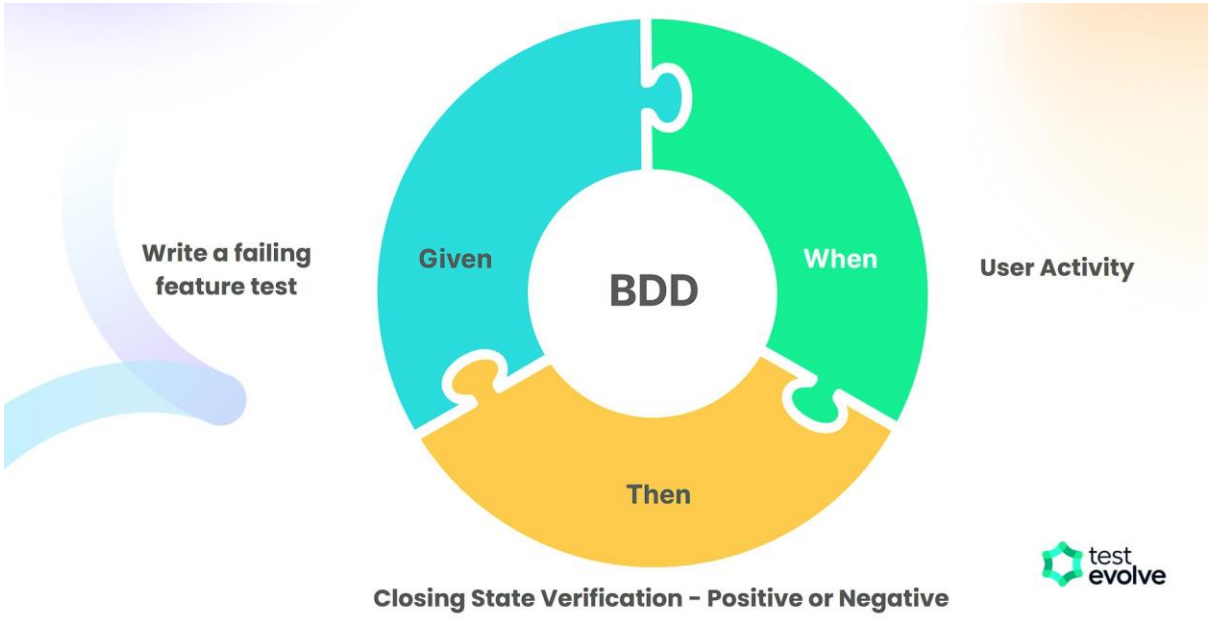
- **Clear Requirements Understanding:** Helps in creating a shared understanding of software behaviors and expected outcomes.

- **Reduced Rework:** By focusing on behaviors, BDD reduces the chances of implementing incorrect or unnecessary features.

Suitability:

- **Complex Business Logic:** Ideal for projects with complex business logic or requirements.

- **Cross-Functional Teams:** Suitable for teams where collaboration between developers, testers, and business stakeholders is essential.



Feature-Driven Development (FDD)

Definition: FDD is an iterative and incremental software development methodology that focuses on building features in short cycles. It emphasizes a feature-centric approach to development.

Key Concepts:

1. **Feature List:** Create a detailed list of all the features your software should have, based on what the project requires. This list acts as a roadmap for development, guiding the team on what needs to be built.

2. **Iterative Development:** Develop the features in small, manageable parts rather than all at once. This means working on one feature or a small group of features at a time, completing them, and then moving on to the next set of features. It's like building a house room by room instead of all at once.

3. **Inspection:** After completing a feature or a set of features, thoroughly review and inspect them to make sure they meet the quality standards set for the project. This includes testing, checking for bugs, and ensuring that the feature works as intended. Only after inspection and approval do you move on to developing the next feature.

Benefits:

- **Efficient Feature Delivery:** Enables efficient delivery of features based on priority and customer value.

- Project Visibility: Provides clear visibility into project progress through feature completion tracking.

- Scalability: Suitable for large-scale projects with multiple teams, as work is organized around features.

Suitability:

- **Well-Defined Features:** Best suited for projects where features are well-defined and can be developed independently.

- **Large-Scale Projects:** Effective for large-scale projects with multiple teams working on different features simultaneously.