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

Aspect	TDD	BDD	FDD
Focus	Code correctness	User behavior	Feature delivery
Process Steps	Write Test Run Test (Fail) Write Code Run Test (Pass) Refractor	Define Behavior Write Examples (Tests) Implement Code Run Tests Refractor	Develop Model Build Feature List Plan by Feature Design by Feature Build by Feature
Advantages	Ensures code quality Facilitates refactoring Provides regression	Improves communication Ensures user requirements Provides understanding	Delivers features fast Clear structure Reduces risks
Disadvantages	Time-consuming Initial effort Tests for trivial code	Requires stakeholder buy-in Writing good examples Thorough collaboration	Complex to manage Needs upfront planning Not for small projects
Applicability	Ensuring code quality Preventing bugs Clear requirements	Enhancing communication Ensuring user satisfaction Frequent changes	Large projects Feature-rich apps Quick feature delivery

Test-Driven Development (TDD), Behavior-Driven Development (BDD), and Feature-Driven Development (FDD) are three distinct methodologies in software development. Each has its unique approaches, benefits, and suitability for different contexts. Here's a detailed comparison:

1. Test-Driven Development (TDD)

Approach:

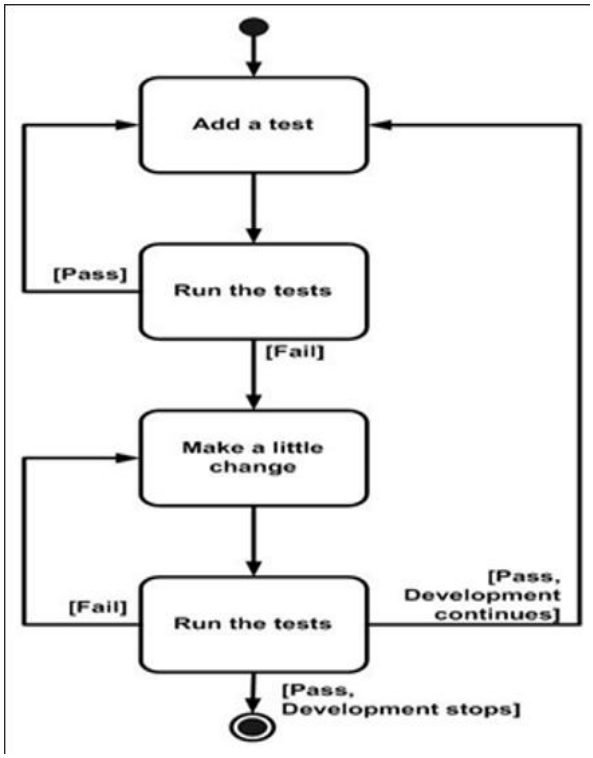
- TDD is a software development process where developers write automated test cases before writing the actual code.
- The process follows a repetitive cycle: Write a test → Run the test (it should fail) → Write code to make the test pass → Refactor the code → Repeat.

Benefits:

- Ensures that the code meets the specified requirements.
- Reduces bugs and errors in the later stages of development.
- Facilitates better design and more maintainable code.
- Promotes simple, clean, and bug-free code.

Suitability:

- Best suited for complex projects requiring high reliability and where the requirements are well understood.
- Useful in environments emphasizing automated testing and continuous integration.
- Ideal for projects where code quality and long-term maintenance are critical.



2. Behavior-Driven Development (BDD)

Approach:

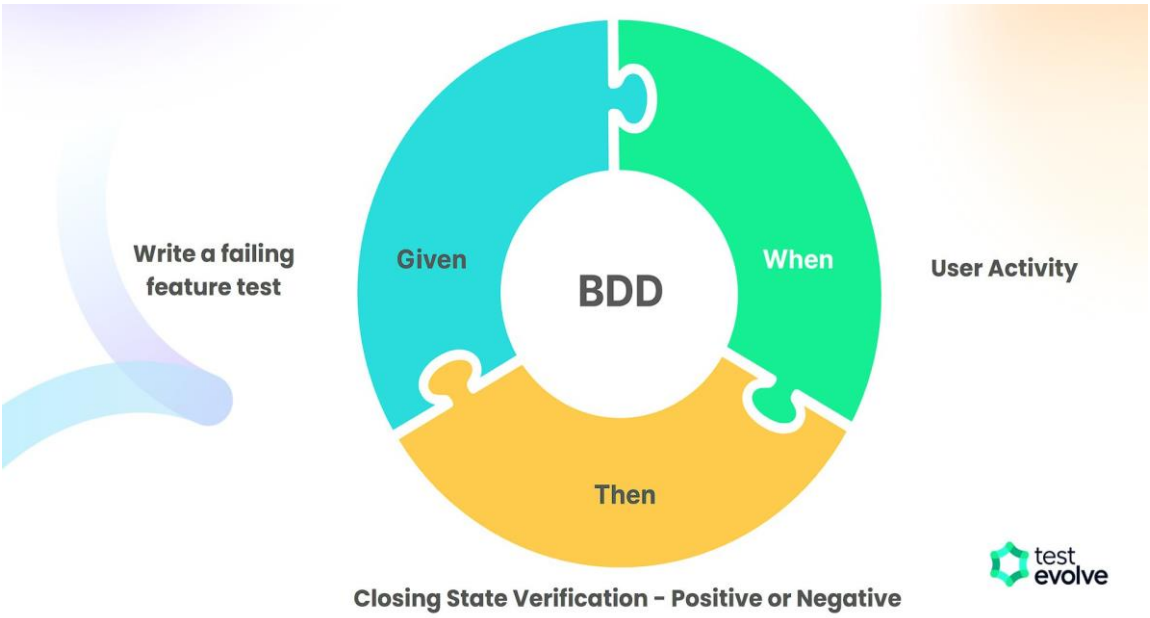
- BDD extends TDD by writing test cases in a natural language that non-technical stakeholders can understand.
- It focuses on the behavior of the application from the end-user's perspective.
- Test scenarios are written in the Given-When-Then format.

Benefits:

- Enhances collaboration between developers, QA, and non-technical stakeholders.
- Provides clear, understandable documentation of the system behavior.
- Ensures that all stakeholders have a shared understanding of the requirements.
- Encourages writing tests that reflect user stories and business language.

Suitability:

- Ideal for projects with significant stakeholder involvement and where requirements are likely to evolve.
- Useful in agile development environments where frequent feedback from stakeholders is necessary.
- Suitable for applications with complex business logic and where understanding user behavior is crucial.



3. Feature-Driven Development (FDD)

Approach:

- FDD is a model-driven, iterative, and incremental development process.
- It involves developing features, which are small, client-valued functions, within two-week iterations.
- The process includes five main activities: developing an overall model, building a feature list, planning by feature, designing by feature, and building by feature.

Benefits:

- Provides a structured, step-by-step approach to development.
- Focuses on delivering tangible, working software frequently.
- Ensures that development aligns with client-valued features.
- Facilitates progress tracking and management of large projects.

Suitability:

- Best suited for larger projects with clear, well-defined features.
- Useful in teams requiring a structured approach and where managing complexity is essential.
- Suitable for projects where regular delivery of client-valued features is a priority.