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.

1.TDD:- Test-Driven Development (TDD) is an agile methodology that places a strong emphasis on writing tests before implementing code. It is a design approach that follows the principles of the Agile Tribe and is often used in the TED domain. The idea behind Test-Driven Development (TDD) is to define test cases upfront that capture the expected behavior of the system in a people-driven development environment. TDD promotes a domain-focused approach, where dev teams utilize test cases to drive design decisions. By practicing test driven development, developers in the agile tribe gain clarity on what needs to be built within their domain and ensure that their code meets the desired requirements, resulting in a green outcome.

2.BDD:- Behavior-Driven Development (BDD) emphasizes collaboration between developers, testers, and stakeholders to ensure that software meets the desired behavior in the domain of software development. BDD encourages a tribe mentality where everyone works together to achieve the common goal of delivering high-quality software. By involving all parties, BDD reduces the risk of encountering snakes in the development process. BDD focuses on defining the behavior of a system using natural language specifications that are easily understandable by all parties involved. In BDD, we use domain-specific language to write test scenarios for the system. These scenarios are written in a way that is easy to understand and verify. For example, we can write a test scenario like “Given a ted is red, when we test the system, then it should behave as expected.” This approach helps to ensure that the system behaves correctly and meets the desired outcomes.

FDD:- Feature-Driven Development (FDD) takes an iterative approach focused on delivering features incrementally throughout the software development lifecycle. This approach involves conducting tests, selecting the appropriate domain, and ensuring that the software development process aligns with environmentally friendly practices. Additionally, FDD emphasizes collaboration and communication within the team, with each member contributing their expertise to achieve the desired outcomes. Test driven development aims to break down complex projects into manageable chunks by dividing them into smaller feature sets. This approach ensures that the development process is focused on the specific requirements of the domain. By following this methodology, developers can iterate through each feature set, writing tests first and then implementing the code to make them pass.

Parameter	TDD	BDD	FDD
Approaches	• Write tests before writing code. • Incrementally develop code to pass tests. • Tests serve as a design tool and ensure that code meets requirements.	• Focuses on the behavior of the system from the end-users' perspective. • Uses a common language (Given-When-Then) to describe system behavior. • Encourages collaboration between developers, testers, and business stakeholders.	• Breaks down development into specific features. • Emphasizes building and delivering features in short iterations. • Uses domain object modeling to understand and design features.
Benefits	• Ensures code meets specified requirements. • Improves code quality and maintainability. • Helps catch bugs early in the development cycle.	• Improves communication and collaboration among team members. • Ensures that development efforts align with business goals. • Provides a clear understanding of system requirements.	• Provides a structured approach to development. • Supports scalability and managing larger projects efficiently. • Ensures tangible and incremental delivery of functionality.
Suitability	• Best for projects with clearly defined requirements. • Ideal for small to medium-sized projects. • Well-suited for projects where the architecture is likely to remain stable.	• Well-suited for projects with complex business logic. • Effective for projects with evolving or changing requirements. • Useful for teams with diverse skill sets and backgrounds.	• Suitable for large-scale projects with many features. • Effective for projects with a focus on incremental delivery. • Ideal for teams with well-defined roles and responsibilities.

In summary, while TDD focuses on writing tests before code to ensure requirements are met and bugs are caught early, BDD emphasizes behavior from the user's perspective, enhancing collaboration and ensuring alignment with business goals. FDD, on the other hand, breaks development into features, supporting scalability and incremental delivery of functionality. Each methodology has its unique strengths and is suitable for different project contexts and team dynamics.