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.

Solution:

Infographic Title:

Comparing TDD, BDD, and FDD Methodologies

Sections and Content:

1. Introduction

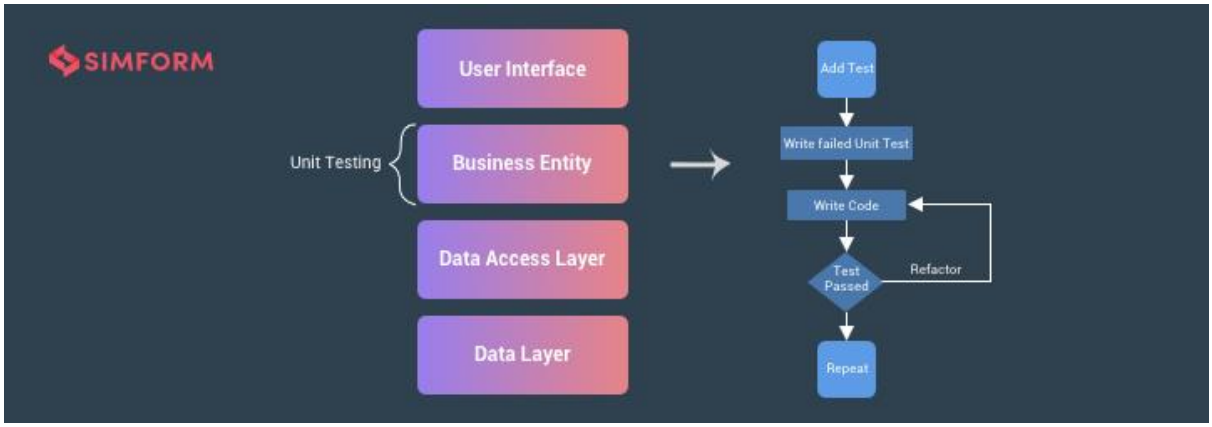
- **Title**: Understanding Development Methodologies
- **Description**: Briefly explain that TDD, BDD, and FDD are methodologies used to enhance software development processes by focusing on different aspects of development and testing.

2. Methodology Overviews

- **Title**: What Are TDD, BDD, and FDD?
- **Visual**: Use three distinct sections or columns for each methodology.

TDD (Test-Driven Development)

- **Definition**: Write tests before code to ensure functionality is verified continuously.
- **Process**:
    1. Write a test.
    2. Run the test (fail).
    3. Write code to pass the test.
    4. Run tests again (pass).
    5. Refactor the code.



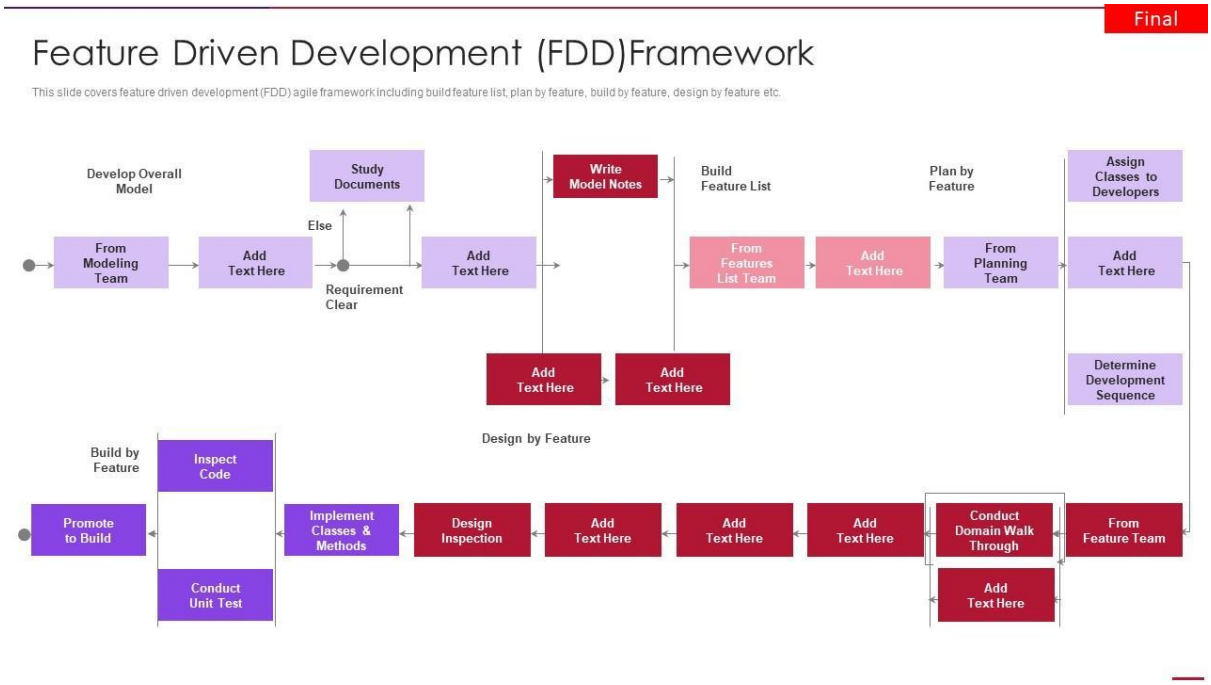
BDD (Behavior-Driven Development)

- **Definition**: Extends TDD by writing tests in a natural language that non-developers can understand, focusing on the behavior of the application.
- **Process**:
    1. Define scenarios in plain language.
    2. Write step definitions.
    3. Implement the underlying code.
    4. Run tests.
    5. Refactor.

FDD (Feature-Driven Development)

- **Definition**: An agile methodology focused on delivering tangible, working software repeatedly in the form of features.

- **Process**:
    1. Develop an overall model.
    2. Build a feature list.
    3. Plan by feature.
    4. Design by feature.
    5. Build by feature.



3. Comparative Table

- **Title**: TDD vs. BDD vs. FDD

Aspect	**TDD**	**BDD**	**FDD**
Focus	Code correctness	User behavior	Feature delivery
Language	Programming languages	Natural language	Feature list
Participants	Developers	Developers, Testers, Business Analysts	Developers, Project Managers
Benefits	Early bug detection, reliable code	Improved collaboration, clear requirements	Continuous delivery, manageable segments
Best For	Complex algorithms, backend systems	User-centric applications, web apps	Large projects, iterative delivery

4. Benefits and Suitability

- **Title**: When to Use Each Methodology

TDD

- **Benefits**:
    - **Early Bug Detection**: Bugs are found early in the development cycle.
    - **Reliable Code**: Ensures code works as expected.
    - **Continuous Integration**: Fits well with CI/CD practices.
- **Suitability**:
    - Ideal for complex algorithms, backend systems, and applications requiring high reliability.

BDD

- **Benefits**:
    - **Enhanced Collaboration**: Facilitates better communication among stakeholders.
    - **Clear Requirements**: Ensures requirements are understood and met.
    - **User-Centric**: Focuses on user behavior and experience.
- **Suitability**:
    - Suitable for web applications, user interfaces, and projects requiring clear requirements and collaboration.

FDD

- **Benefits**:
    - **Tangible Progress**: Delivers working software frequently.
    - **Manageable Segments**: Breaks down projects into smaller, manageable features.
    - **Clear Documentation**: Each feature is well-documented.
- **Suitability**:
    - Best for large projects, iterative development, and teams needing a structured approach.