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. Introduction

Brief Overview: Introduce the three methodologies – Test-Driven Development (TDD), Behavior-Driven Development (BDD), and Feature-Driven Development (FDD).

2. Methodology Breakdown

TDD (Test-Driven Development)

Approach:

- Write tests before writing the actual code.
- Follow the Red-Green- Refactor cycle.

Benefits:

- Early bug detection.
- High code quality.
- Regression protection

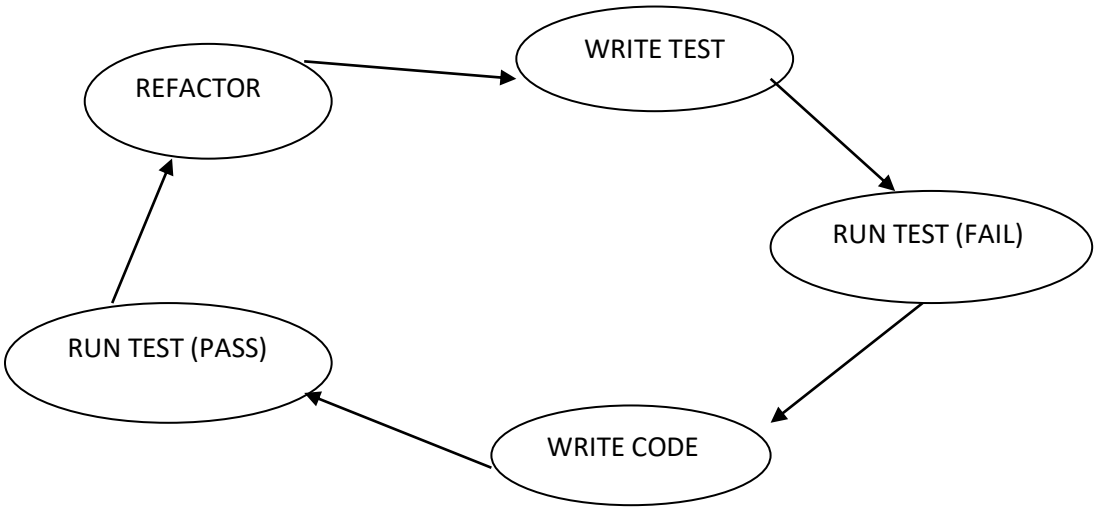
Suitable For:

- Projects requiring high reliability.
- Developers comfortable with automated testing.

Visual:

A cycle diagram showing: Write Test -> Run Test (Fail) -> Write Code -> Run Test (Pass) -> Refactor.

VISUAL DIGRAM:



BDD (Behavior-Driven Development)

Approach:

- Write tests based on user stories and behavior scenarios.
- Use Gherkin language for creating test cases.

Benefits:

- Enhanced collaboration between non-technical stakeholders and developers.
- Clear documentation of requirements.
- Ensures software meets business needs.

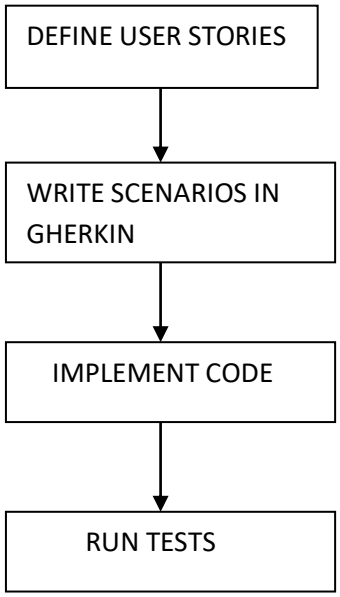
Suitable For:

- Projects needing strong communication between business and technical teams.
- Ensuring clear understanding of requirements.

Visual:

A flowchart showing: Define User Stories -> Write Scenarios in Gherkin -> Implement Code -> Run Tests.

FLOW CHART :



FDD (Feature-Driven Development)

Approach:

- Develop features based on client-valued functionality.
- Follow the five-step process: Develop Overall Model -> Build Feature List -> Plan by Feature -> Design by Feature -> Build by Feature.

Benefits:

- Clear focus on delivering tangible features.
- Scalability for larger teams.
- Regular progress tracking.

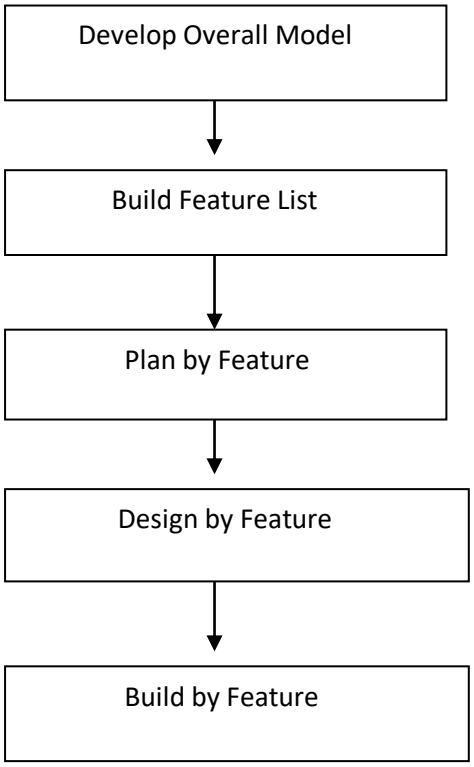
Suitable For:

- Large, complex projects.
- Teams requiring regular client feedback and iteration.

Visual:

A process diagram showing the five steps of FDD.

Process Diagram of FDD:



3. Use Cases

TDD: Best for projects needing high code quality and reliability, such as financial software or APIs.

BDD: Ideal for projects where clear communication between stakeholders is crucial, like e-commerce platforms or customer-facing applications.

FDD: Suitable for large-scale projects with a need for structured and iterative feature delivery, such as enterprise software or large IT systems.