TDD and BDD

Q- Difference between TDD and BDD methodologies. Illustrate their unique approaches, benefits, and suitability for different software development contexts. Use visuals to enhance understanding.

Software teams use TDD (Test-Driven Development) and BDD (Behaviour-Driven Development) to improve software quality.

Both involve testing, but their approach, mindset, and usage are very different.

1)TDD – Test-Driven Development

TDD is a technique where **developers write tests first** and then write code to make those tests pass.

TDD Approach (Visual – Red → Green → Refactor)

1. Write a failing test (RED)

2. Write minimum code to pass (GREEN)

3. Refactor and clean code (REFACTOR)

TDD Example (Add to Cart logic)

Before coding, the developer writes tests:

- ✓ **Test 1:** When the user adds a product, the cart count becomes 1
- ✓ **Test 2:** When the same product is added again, the quantity increases
- ✓ **Test 3:** Total price updates correctly

Then the developer writes the code to satisfy these tests.

TDD Benefits

- Clean and reliable code
- Fewer bugs
- Safe refactoring
- High test coverage

Where TDD is suitable

- Backend logic

- Calculations & validations
- Financial systems (banking, payments)
- APIs & microservices

2)BDD – Behaviour-Driven Development

Behaviour-Driven Development (BDD) is a way of developing software where the main focus is on the behaviour of the application — how the system should behave from the user's point of view.

It uses a simple approach:

Given – When – Then

BDD is used by business teams, testers, and developers together.

BDD Approach

Given (starting condition)

When (user action)

Then (expected result)

BDD Behaviour Examples

1. Login Behaviour (Amazon Checkout Flow)

- ➢ Given the user is logged in
- ➢ When they search for a product
- ➢ And add it to the cart
- ➢ And make the payment
- ➢ Then an order should be created
- ➢ And the order should be delivered to the user

This describes the full behaviour of a customer who is already logged in.

2. Search Behaviour Without Logging In

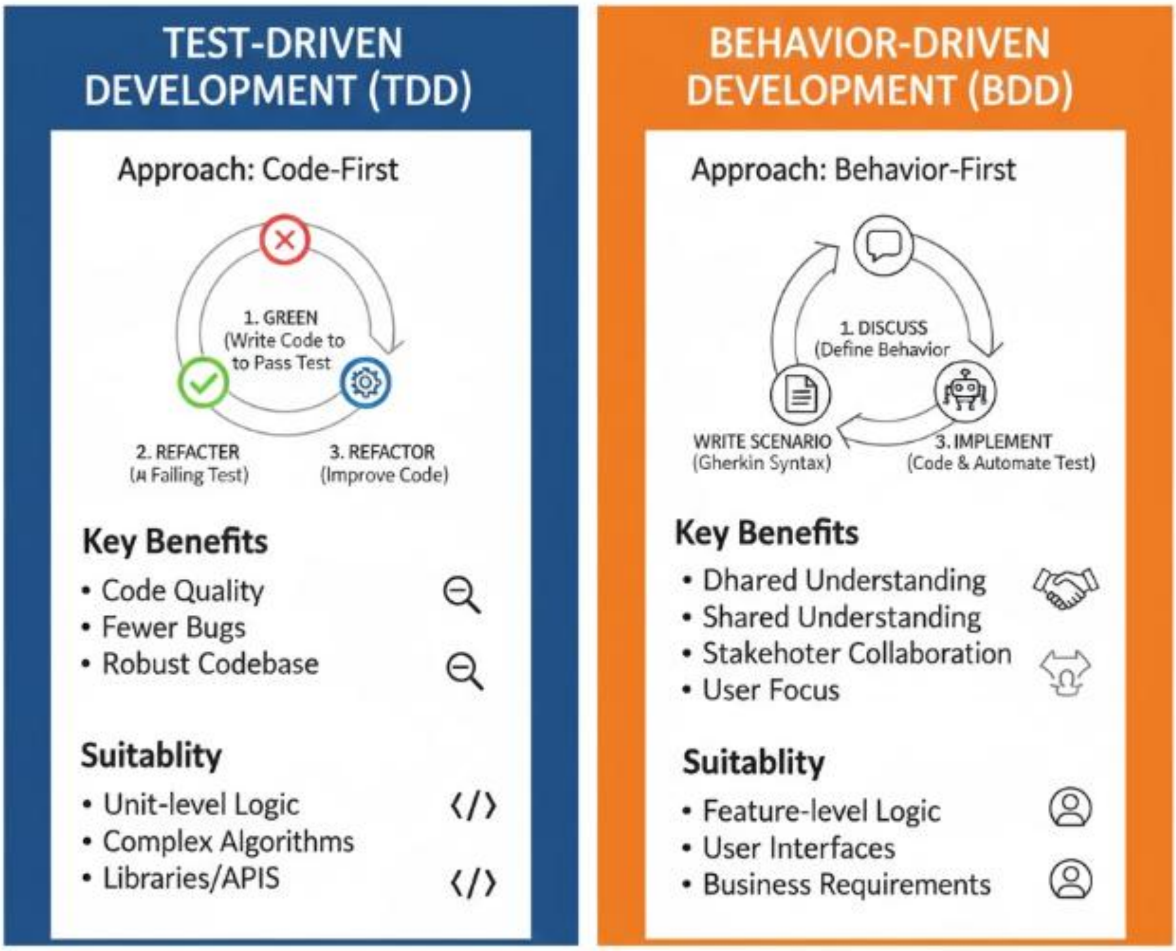
- ➢ Given the user is not logged in
- ➢ When they search for a product
- ➢ And add the product to the cart
- ➢ And click on "Proceed to Payment"
- ➢ Then the system should redirect the user to the Login screen

BDD Benefits

- Removes confusion between teams
- Requirements are written in plain English
- Easy for testers and business people to understand
- Behaviours become documentation

Where BDD is suitable

- User-facing features
- E-commerce flows (Search → Cart → Payment → Order)
- Login, Signup, Booking
- Any application where user behaviour matters



TEST-DRIVEN DEVELOPMENT (TDD)

Approach: Code-First

Key Benefits

- Code Quality
- Fewer Bugs
- Robust Codebase

Suitablity

- Unit-level Logic
- Complex Algorithms
- Libraries/APIS

BEHAVIOR-DRIVEN DEVELOPMENT (BDD)

Approach: Behavior-First

Key Benefits

- Dhared Understanding
- Shared Understanding
- Stakehoter Collaboration
- User Focus

Suitablity

- Feature-level Logic
- User Interfaces
- Business Requirements