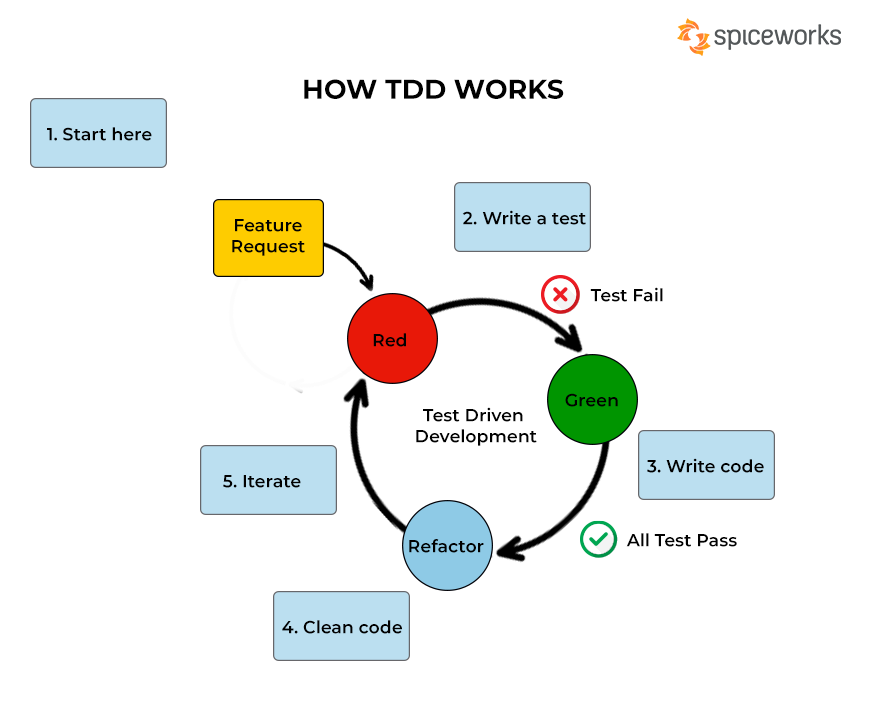
Day-3 (Assignment-2)

# **Q)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.**

## **Test-Driven Development [ TDD ]**

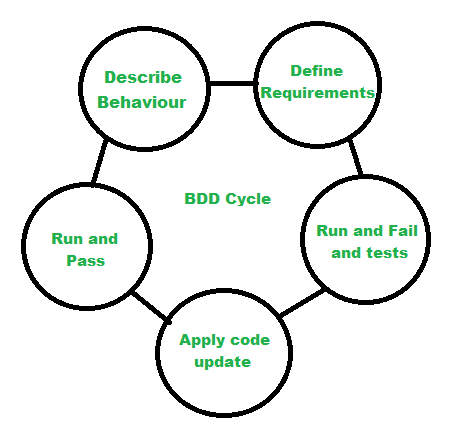
**Test Driven Development** is a software development methodology that emphasizes writing **tests before writing the actual code**. It ensures that code is always tested and functional, reducing bugs and improving code quality. In TDD, developers write **small, focused tests** that define the desired functionality, then write the minimum code necessary to pass these tests, and finally, refactor the code to improve structure and performance.



* **Approach:**
  + Write tests before writing the code.
  + Follow the Red-Green-Refactor cycle:
    - **Red:** Write a test that fails.
    - **Green:** Write the minimal code to pass the test.
    - **Refactor:** Optimize the code while keeping tests passing.
* **Benefits:**
  + Ensures high test coverage.
  + Encourages simple and modular code.
  + Facilitates early bug detection.
* **Suitability:**
  + Suitable for projects requiring high reliability and where code quality is paramount.
  + Ideal for complex systems with intricate logic.

## **Behavioral-Driven Development (BDD)**

**Behavior-Driven Development (BDD)** is an Agile software development methodology in which an application is documented and designed around the behavior a user expects to experience when interacting with it. By encouraging developers to focus only on the requested behaviors of an app or program, BDD helps to avoid bloat, excessive code, unnecessary features or lack of focus. This methodology combines, augments and refines the practices used in test-driven development ([TDD](https://www.techtarget.com/searchsoftwarequality/definition/test-driven-development)) and [acceptance testing](https://www.techtarget.com/searchsoftwarequality/definition/acceptance-test).



 **Approach:**

* Extend TDD by writing tests in a natural language style using Given-When-Then syntax.
* Collaboration between developers, testers, and non-technical stakeholders.
* Focus on the behavior of the application from the end-user perspective.

 **Benefits:**

* Improves communication between technical and non-technical team members.
* Ensures that development aligns with business requirements.
* Produces documentation that serves as both specifications and tests.

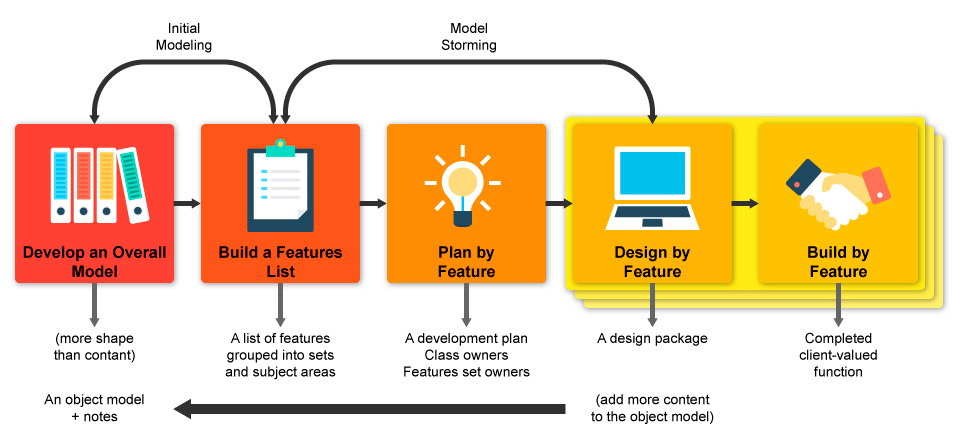
 **Suitability:**

* Suitable for projects where requirements can change frequently.
* Ideal for projects needing close collaboration between stakeholders.

## **Feature-Driven Development (FDD)**

**FDD** stands for **Feature-Driven Development**. It is an agile iterative and incremental model that focuses on progressing the features of the developing software. The main motive of feature-driven development is to provide timely updated and working software to the client. In FDD, reporting and progress tracking is necessary at all levels.

FDD was first applied in the year 1997 on a real-world application by **Jeff De Luca** for large software development with specific needs of 15-month and 50 persons and published as a discussion in book **Java Modeling in Color with UML** in the year 1999.



 **Approach:**

* Break down the project into features (small client-valued functions).
* Develop each feature in five steps: Develop Overall Model, Build Feature List, Plan by Feature, Design by Feature, Build by Feature.
* Emphasis on design and building features iteratively.

 **Benefits:**

* Clear visibility of progress through feature tracking.
* Encourages frequent, tangible results.
* Scalable for large teams and projects.

 **Suitability:**

* Suitable for large-scale projects with distinct, valuable features.
* Ideal for projects requiring incremental releases.

### **Comparative Table**

|  |  |  |  |
| --- | --- | --- | --- |
| Aspect | TDD | BDD | FDD |
| **Focus** | Code correctness through tests | Behavior and requirements of the system | Client-valued features |
| **Approach** | Red-Green-Refactor cycle | Given-When-Then scenarios | Feature breakdown and iterative design/build |
| **Benefits** | High test coverage, early bug detection | Improved communication, alignment with business needs | Frequent, tangible progress, scalability |
| **Suitability** | Complex systems, high reliability | Projects with changing requirements | Large-scale projects, incremental releases |