**DDD – Domain-Driven Design**

**Domain-Driven Design** focuses on modeling software based on the core business domain and its logic. It emphasizes collaboration between technical and domain experts to create a shared understanding. The goal is to align the software design closely with business needs, using concepts like entities, value objects, aggregates, and bounded contexts.

**TDD – Test-Driven Development**

**Test-Driven Development** is a development approach where tests are written **before** the actual code. The cycle follows Red-Green-Refactor: write a failing test (Red), write just enough code to pass it (Green), then refactor for improvement. TDD helps ensure code correctness, improve design, and reduce bugs early in the development process.

**BDD – Behavior-Driven Development**

**Behavior-Driven Development** extends TDD by focusing on the expected behavior of the application from the user’s perspective. It uses natural language (like “Given-When-Then” format to describe test scenarios, making it easier for developers, testers, and business stakeholders to collaborate. BDD ensures that development aligns with business goals and user expectations.

**FDD – Feature-Driven Development**

**Feature-Driven Development** is a model-driven, short-iteration process that focuses on building and delivering **features**—small, client-valued functions. It involves defining an overall model, building feature lists, and then planning and developing by feature. FDD is suitable for large teams and emphasizes incremental progress, good design, and frequent delivery.