### **TDD – Test-Driven Development**

* **Development Approach:**  
  Code is developed by first writing automated unit tests. Focuses on small, independently testable components.
* **Workflow:**  
  **Write Test → Test Fails → Write Minimum Code → Test Passes → Refactor → Repeat**
* **Core Advantages:**
  + Catches bugs early in development
  + Promotes clean, modular, high-quality code
  + Provides a strong foundation for safe refactoring
* **Ideal For:**  
  Systems with heavy backend logic, technical development teams, API and library development

### **BDD – Behavior-Driven Development**

* **Development Approach:**  
  Begins with defining system behavior in plain language using structured formats like **Given–When–Then**. Encourages collaboration across technical and non-technical roles.
* **Workflow:**  
  **Describe Behavior → Automate Test Scenario → Write Code → Validate Behavior → Refactor**
* **Core Advantages:**
  + Improves cross-team communication
  + Ensures shared understanding of requirements
  + Keeps development aligned with user needs and business goals
* **Ideal For:**  
  Agile teams, user-focused products, collaborative project environments

### **FDD – Feature-Driven Development**

* **Development Approach:**  
  Focuses on building features based on a detailed domain model. Work is planned and delivered feature-by-feature.
* **Workflow:**  
  **Create Feature List → Plan by Feature → Design by Feature → Build Feature → Iterate**
* **Core Advantages:**
  + Well-structured and scalable in large teams
  + Delivers tangible value consistently
  + Keeps development aligned with business priorities
* **Ideal For:**  
  Enterprise-level systems with predefined feature requirements and large development teams