**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.

A software developer is building a mobile shopping app.

**TDD (Test-Driven Development):**

* **Icon:** A magnifying glass over code lines.
* **Focus:** Unit-level testing.
* **Process:**
  1. **Write failing test (Red):** A test case is written to check a specific functionality (e.g., adding items to cart). This test will initially fail because the code isn't written yet.
  2. **Write minimal code to pass (Green):** Developer writes just enough code to make the failing test pass. This ensures the core functionality works.
  3. **Refactor (Refactor):** The code is reviewed and improved for readability and maintainability without affecting functionality.
* **Benefits:**
  1. **Bug Reduction:** Early detection and isolation of errors.
  2. **Clean Code:** Focus on small, well-tested functions.
* **Suitable for:** Individual developers writing core functionalities or building unit tests for larger projects.

**BDD (Behaviour-Driven Development):**

* **Icon:** A group of people discussing features around a whiteboard.
* **Focus:** User behavior and acceptance criteria.
* **Process:**
  1. **Collaborate (Discuss):** Developers, stakeholders, and testers discuss user stories and desired functionalities (e.g., user adds items to cart and sees a running total). These discussions are captured in "gherkin" syntax, a plain-language format focusing on user behavior.
  2. **Develop Scenarios (Scenarios):** Specific scenarios are created based on the user stories, outlining expected system behavior under different conditions (e.g., adding multiple items, handling empty cart).
  3. **Automate Tests (Automate):** Automated tests are written based on the defined scenarios. These tests ensure the system delivers the agreed-upon functionalities from the user's perspective.
* **Benefits:**
  1. **Shared Understanding:** Clear communication between developers and non-technical stakeholders.
  2. **Focus on Value:** Ensures development aligns with user needs.
* **Suitable for:** Teams working on user-centric applications where clear communication and shared understanding are crucial.

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

* **Icon:** A puzzle with different pieces fitting together.
* **Focus:** Iterative development with feature-based decomposition.
* **Process:**
  1. **Model the Big Picture (Model):** A high-level model is created, outlining the overall system features and functionalities.
  2. **Feature Decomposition (Decompose):** Features are broken down into smaller, more manageable tasks (e.g., adding items to cart, displaying cart contents).
  3. **Iterative Development (Develop):** The development team works in short iterations, focusing on completing well-defined features within each iteration. This allows for continuous feedback and adaptation.
* **Benefits:**
  1. **Reduced Risk:** Manages project complexity through iterative development.
  2. **Improved Estimation:** More accurate estimations for feature development.
* **Suitable for:** Larger projects with complex features that require breaking down into smaller, manageable chunks.