

### **Assignment 1:**

Create an infographic illustrating the Test-Driven Development (TDD) process. Highlight steps like writing tests before code, benefits such as bug reduction, and how it fosters software reliability.

Test-Driven Development (TDD) is a software development process that emphasizes writing tests before writing the actual code. The TDD process can be illustrated as a cyclical flow, with the following steps:

#### **Title: Test-Driven Development (TDD) Process**

##### **Step 1:** Write a failing test.

- Write a test case for a specific functionality that you want to implement.
- Ensure that the test case fails, as the functionality is not yet implemented.

##### **Step 2:** Write the minimum code to pass the test.

- Write the minimum amount of code required to pass the test.
- Ensure that the test case passes.

##### **Step 3:** Refactor the code.

- Refactor the code to improve its readability, maintainability, and performance.
- Ensure that the test case still passes.

##### **Step 4:** Repeat the process.

- Repeat steps 1-3 for the next functionality that you want to implement.
- Continue the cycle of writing tests, writing code, and refactoring.

#### **Benefits of TDD:**

- **Bug reduction:** Writing tests before code helps to identify and fix bugs early in the development process, reducing the overall number of bugs in the final product.
- **Software reliability:** TDD helps to ensure that the software is reliable and performs as expected, as each functionality is tested thoroughly.
- **Improved design:** Writing tests first forces developers to think about the design and architecture of the software, leading to better design decisions.
- **Faster development:** TDD can lead to faster development in the long run, as it reduces the time spent on debugging and testing.

#### **The infographic could also showcase how TDD fosters software reliability by:**

- Ensuring that all code paths are tested, reducing the risk of untested scenarios.
- Facilitating continuous integration and continuous deployment (CI/CD) practices, as the test suite provides a safety net for automated testing and deployment.
- Encouraging collaboration and communication among development teams, as tests serve as a common language for describing expected behaviour.

## **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.

### **Title: TDD vs BDD vs FDD Methodologies**

Here's a breakdown of the key information you could include in the infographic:

#### **1. Test-Driven Development (TDD):**

- Approach: Write tests before writing the production code.
- Focus: Unit testing and code implementation.
- Benefits: Early bug detection, regression prevention, modular design, improved code quality.
- Suitable for: Projects with well-defined requirements, where unit testing is crucial.

#### **2. Behaviour-Driven Development (BDD):**

- Approach: Define expected behaviour in plain language before writing tests and code.
- Focus: Acceptance testing and collaboration between stakeholders.
- Benefits: Improved communication, better understanding of requirements, living documentation.
- Suitable for: Projects with complex requirements or where collaboration is essential.

#### **3. Feature-Driven Development (FDD):**

- Approach: Develop features in a iterative, client-centric manner.
- Focus: Feature design, planning, and implementation.
- Benefits: Frequent client feedback, feature-focused development, adaptability.
- Suitable for: Projects with rapidly changing requirements or high user involvement.