Name: Tanmay Shailendra Patil

Batch: CPPE_Java Full Stack

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) Process Infographic

1. Write Test Cases:

- Define tests based on requirements and desired behavior.
- Write tests before writing any code.

2. Run Tests:

- Execute tests to ensure they fail initially (as no code has been written yet).

3. Write Code:

- Develop code to pass the failing tests.
- Focus on writing only the code necessary to make the test pass.

4. Run Tests Again:

- Execute tests to check if the newly written code passes.
- If tests fail, refine code until they pass.

5. Refactor Code:

- Improve code structure without changing its functionality.
- Ensure code remains clean and maintainable.

Benefits of TDD:

- **Bug Reduction:** By writing tests before code, potential bugs are identified and fixed early in the development process.
- **Improved Software Reliability:** Ensures that the codebase is continuously validated, leading to more reliable software.
- **Clear Requirements:** Helps clarify requirements and desired behavior before implementation.

- **Encourages Modular Design:** Promotes modular and loosely coupled code, enhancing maintainability and scalability.
- **Faster Development:** Reduces debugging time and speeds up development by catching errors early.

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.

Comparative Infographic: TDD vs BDD vs FDD

1. Test-Driven Development (TDD):

- **Approach:**
- Write tests before writing code.
- Focus on small, incremental development cycles.
- **Benefits:**
- Early bug detection.
- Improved code quality.
- Clear requirements.
- **Suitability:**
- Suitable for projects where requirements are well-defined.
- Effective for unit testing and code validation.

2. Behavior-Driven Development (BDD):

- **Approach:**
- Focuses on behavior and interactions.
- Uses natural language to describe scenarios.
- **Benefits:**
- Collaboration between developers and stakeholders.
- Enhanced understanding of user needs.

- **Suitability:**
- Ideal for projects with complex business logic.
- Promotes shared understanding of requirements.

3. Feature-Driven Development (FDD):

- **Approach:**
- Emphasizes iterative and feature-based development.
- Divides project into small, manageable features.
- **Benefits:**
- Efficient management of large projects.
- Focus on delivering tangible features.
- **Suitability:**
- Well-suited for large-scale projects with multiple teams.
- Provides clear progress tracking and accountability.