Name: Subha Ranjan Sahoo

Batch: CPPE_Java Full

Assignment 1

Q. 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.

Answer:

Write Tests:

First, write tests for the functionality you want to add.

2. Run Tests (Fail):

Test your code. At first, expect it to fail because there's no code yet.

3. Write Code:

Write the code needed to pass the tests. Keep it simple and focused.

4. Run Tests (Pass):

Run the tests again. If they pass, your code works as expected.

5. Refactor Code:

Improve your code without changing what it does. This makes it cleaner and easier to understand.

Benefits of TDD:

1. Fewer Bugs:

Testing first helps catch and fix bugs early, making your code more reliable.

2. Faster Development:

TDD lets you develop one small piece at a time, speeding up the process.

3. Better Code Quality:

TDD encourages writing clean and reusable code, making it easier to maintain.

4. More Reliable Software:

With TDD, you can confidently make changes without breaking existing functionality, making your software more dependable.

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.

Answer:

Test-Driven Development (TDD)

Approach:

Write tests before writing code.

Tests focus on functionality and requirements.

Code is developed incrementally to pass tests.

Benefits:

Early bug detection.

Improved code quality.

Confidence in code changes.

Suitable for iterative development.

Suitability:

Agile and iterative projects.

Projects with well-defined requirements.

Projects where automated testing is essential.

Behavior-Driven Development (BDD)

Approach:

Define behaviour using user stories.

Write acceptance tests based on behaviour.

Implement code to fulfil behaviour.

Benefits:

Enhanced collaboration between teams.

Clear communication between stakeholders.

Focus on user needs and expectations.

Encourages a shared understanding of requirements.

Suitability:

Collaborative projects with non-technical stakeholders.

Projects with complex business logic.

Projects where understanding user behavior is critical.

Feature-Driven Development (FDD)

Approach:

Break down features into smaller tasks.

Develop features iteratively.

Focus on feature completion and delivery.

Benefits:

Emphasis on feature delivery.

Clear project structure and milestones.

Continuous feedback and adaptation.

Scalable for large and complex projects.

Suitability:

Large-scale projects with multiple teams.

Projects with evolving requirements.

Projects where feature delivery is paramount.