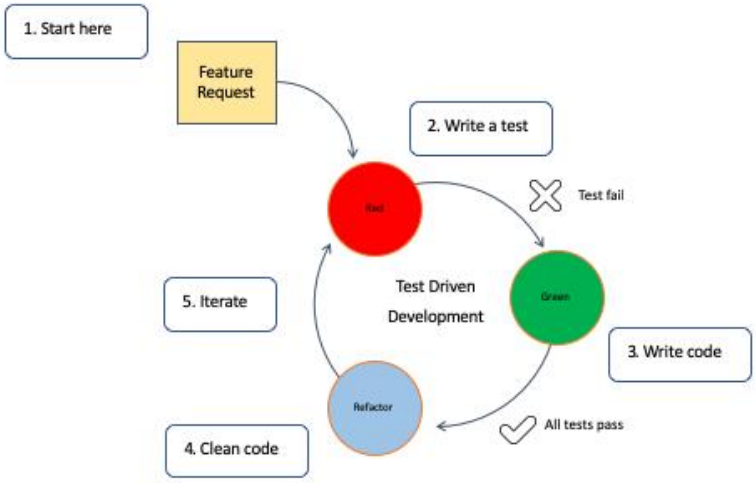
Modern Development Methodologies

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 defined as an iterative methodology that prioritizes the creation of and checking against test cases at every stage of software development, by converting each component of the application into a test case before it is built and then testing and tracking the component repeatedly.

1. **Red –** Create a test case and make it fail.
2. **Green –** Make the test case pass by any means.
3. **Refactor –** Change the code to remove duplicate/redundancy.

Step 1: Write Tests

Illustrate a developer writing unit test for a specific functionality. Include examples like JUnit for Java or Pytest for Python.

Step 2: Run Tests

Show running the tests to demonstrate that they fail initially (red phase).

Step 3: Write Code

Illustrate the developer writing code to pass the failing tests. Show code snippets alongside the tests.

Step 4: Run Tests Again

Demonstrate running the tests again, showing that they pass (green phase).

Step 5: Refactor Code

Depict the developer refactoring the code to improve readability and maintainability. Show before-and-after code snippets.

Benefits of Bug Reduction

Bug prevention and debugging are the key aspects of TDD as they focus on the fact that software must be free from errors so that when the software is launched after development the users do not get any problems due to the software. TDD provides fruitful assistance for bug prevention as stated earlier with the help of test cases because TDD is the tool that can provide the sample inputs for the software, and it is a tool on which almost all the developers and programmers in the software industry trust.
ERRORS can be fixed only when we have some sample inputs for the code and these sample inputs are provided by TDD.

Test-Driven Development (TDD) contributes to software reliability by detecting bugs early, improving code quality, enabling regression testing, supporting confident refactoring, integrating well with CI/CD practices, and reducing technical debt. These factors collectively lead to more reliable and maintainable software products.

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.

TDD	BDD	FDD
Illustrate the process of writing tests before writing the actual code. Emphasize the iterative nature of TDD, where developers write tests, implement code, and refactor continuously. **Unique Approaches** Visual: Developer writing tests first, then writing code to pass those tests, and finally refactoring. Text: "Write tests before code, ensure code meets specifications, refactor for clean design." **Benefits** Visual: Graph showing reduced bugs over time, clean code structure, and faster development cycles. Text: "Bug reduction, improved code quality, faster iterations." **Suitability for Different Contexts** Visual: Software projects with a focus on code quality, continuous integration, and iterative development. Text: "Suitable for projects emphasizing code quality, CI/CD,iterativedevelopment."	Explain that BDD focuses on defining software behaviour using natural language specifications. Highlight the collaboration between developers, QA, and business stakeholders in writing and verifying behaviour specifications. **Unique Approaches** Visual: Collaboration between developers, QA, and business stakeholders to define behaviour using natural language scenarios (e.g., Gherkin syntax). Text: "Define behaviour through scenarios, ensure alignment with business requirements, automate tests based on scenarios." **Benefits** Visual: Collaboration diagram showing stakeholders working together,clear communication, and reduced misinterpretation requirements. Text:"Enhanced collaboration, clear requirements, reduced miscommunication." **Suitability for Different Contexts** Visual: Projects with complex business requirements, collaboration between technical and non-technical stakeholders, and a need for clear behavior specifications. Text: "Ideal for projects with complex business requirements, collaboration between stakeholders, clear behavior specifications."	Describe FDD as an iterative and incremental approach that emphasizes building features based on client requirements. Highlight the feature-centric nature of FDD, where development is organized around specific features or functionalities. **Unique Approaches** Visual:Feature-centric development, with a focus on identifying, designing, and implementing features iteratively. Text: "Identify features, design feature sets, build and integrate features, repeat for incremental development." **Benefits** Visual: Feature delivery timeline, showing incremental feature releases and client satisfaction. Text: "Incremental feature delivery, client satisfaction, flexible development." **Suitability for Different Contexts** Visual: Projects with well-defined features, client-driven development, and a focus on incremental delivery. Text: "Best for projects with well-defined features, client-driven development, incremental delivery."