Test Driven Development

Training Details

Training Time	:	3 Days
Capacity	:	12
Prerequisites	:	There are no prerequisites for this course.

About Training

About Training

Test-Driven Development (TDD) is a design engineering process that relies on a very short development cycle. A TDD approach to software development requires a thorough review of the requirements or design before any functional code is written. The development process is started by writing the test case, then the code to pass the test and then refactoring until completion.

- Benefits of a TDD approach to software engineering include:
- Faster feedback
- Higher Acceptance
- Avoids scope creep and over engineering
- Customer centric and iterative
- Leads to modular, flexible, maintainable code

In this course, you will learn about unit tests, user stories, design, refactoring, frameworks, and how to apply them to existing solutions. In addition, this course explores the implications of code dependencies, fluid requirements, and early detection of issues. This course demonstrates the skills developers and teams need for building quality applications sustainably, with quality, for the life of the code base

What You'll Learn

- Unit Testing principles and practices
- Role of unit testing in software development and testing
- How to write effective unit testing

- Properties of effective unit tests
- How to use mock objects to isolate the “system under test”
- Effective refactoring of the code base
- Benefits of the test-first and test-driven development
- Techniques and practices to aid in the successful adoption of test-driven development
- Using acceptance testing and behavior-driven development to further advance test-driven development

Who Should Attend

- Software developers and programmers
- Agile practitioners
- Quality assurance professionals
- Software testers
- Product owners
- Project managers
- IT managers
- Software engineers

Outline

1. Agile Overview

- What is Agile Software Development
- Components of Agile
- The Role of TDD in Agile Development
- Benefits of Agile Development
- Becoming Agile

2. Principles of Agile Development

- Design Principles Overview
- Coding Principles

3. Unit Testing

- Unit Test Fundamentals
- Advanced Unit Testing
- Frameworks
- Test Runners
- Advanced Test Attributes

4. Test Driven Development

- TDD Rhythm
- Sustainable TDD
- Supporting Practices
- Pair Programming
- Pairing Techniques
- Eight Wastes of Software Development

- Test Automation

5. Refactoring
   - Why Refactor?
   - Refactoring Methods
   - Refactoring Cycle
6. Pair Programming
   - Pair Programming
   - Advantages of Pairing
   - Pairing Techniques
7. Acceptance Testing and BDD
   - Acceptance Testing
   - Best Practices for Effective Testing
   - BDD vs. ATDD
8. Principles and Benefits
   - Consequences of no Testing
   - TDD Solutions
9. Unit Test Examples
   - Queues and Stacks
   - Unit Test Examples
   - Advanced Refactoring
10. Testing for Non-Developers
    - FIT and Fitnesse
    - Additional TDD Considerations