**DAY 3 - ASSIGNMENT 1 MRIGANKA PATRA**

**Emp ID: 2415820**

**Email: mriganka.patra@cognizant.com**

**Tasks**

1. Describe the key skills and responsibilities of a Scrum Master.

**Key Skills**

* **Facilitation** — Helps Scrum events (Daily Scrum, Sprint Planning, Review, Retrospective) run smoothly.
* **Coaching & Mentoring** — Coaches the team on Agile principles and Scrum framework.
* **Conflict Resolution** — Mediates conflicts and removes obstacles to team progress.
* **Communication** — Acts as a bridge between Product Owner, Development Team, and Stakeholders.
* **Servant Leadership** — Focuses on enabling the team rather than directing it.
* **Change Management** — Supports organizational agility and helps teams embrace change.

**Responsibilities**

* **Guide the Team** — Ensures that Scrum practices are followed correctly.
* **Facilitate Scrum Events** — Plans and hosts Scrum ceremonies.
* **Remove Impediments** — Works to eliminate blockers preventing team progress.
* **Support the Product Owner** — Helps manage the Product Backlog effectively.
* **Promote Continuous Improvement** — Encourages reflection and iteration on processes.
* **Shield the Team** — Protects the team from external interruptions and distractions.

2. Compare Test-Driven Development (TDD) and Behavior-Driven Development (BDD): Explain the workflow for each. - Mention tools used (e.g.. JUnit, Cucumber).

|  |  |  |
| --- | --- | --- |
| **Aspect** | **Test-Driven Development (TDD)** | **Behavior-Driven Development (BDD)** |
| Focus | Code correctness & design | Business behavior & collaboration |
| Language | Technical (unit tests) | Business-readable (scenarios) |
| Workflow | Write failing unit test → Write code → Refactor → Repeat | Write failing scenario → Write step definitions → Implement code → Repeat |
| Collaboration | Primarily for developers | Involves developers, testers, and business stakeholders |
| Tools | JUnit, NUnit, TestNG | Cucumber, SpecFlow, Behave |
| Outcome | Well-tested code units | Verified user behavior and flow |

**TDD Workflow:**

1. Write a failing unit test (Red)
2. Write minimal code to pass the test (Green)
3. Refactor the code while keeping tests green
4. Repeat for each feature or unit

**BDD Workflow:**

1. Write a feature file with scenarios in Gherkin
2. Implement step definitions to automate scenarios
3. Write code to make steps pass
4. Refine feature and scenarios iteratively.

3. Choose a simple user story (e.g. 'User resets password). - Write one TDD-style unit test pseudocode. - Write one BDD-style scenario in Gherkin format.

4. Explain how TDD and BDD improve quality and collaboration.

**TDD:**

* **Improves Code Quality:** Forces developers to write tests first, which ensures code is well-designed and testable.
* **Reduces Defects:** Early detection of issues through automated unit tests.
* **Refactoring Safety:** Provides a safety net of tests to confidently refactor code.

**BDD:**

* **Enhances Collaboration:** Brings developers, testers, and business stakeholders together via common, business-readable language (Gherkin).
* **Clarifies Requirements:** Scenarios act as executable specifications, reducing misunderstandings.
* **Improves Test Coverage:** Scenarios cover end-to-end business flows and behavior.
* **Living Documentation:** Feature files serve as up-to-date documentation for the system.