**AIM: Explain the software development scenario using Scrum**

**Objectives:**

·         **Brief introduction about Scrum**

·         **Write one scenario**

**Scrum** is the type of **Agile framework**. It is a framework within which people can address complex adaptive problem while productivity and creativity of delivering product is at highest possible values. Scrum uses **Iterative process**.

**Silent features of Scrum are:**

·         Scrum is light-weighted framework

·         Scrum emphasizes self-organization

·         Scrum is simple to understand

·         Scrum framework help the team to work together

**Lifecycle of Scrum:**

**Sprint:**  
A Sprint is a time-box of one month or less. A new Sprint starts immediately after the completion of the previous Sprint.

**Release:**  
When the product is completed then it goes to the Release stage.

**Sprint Review:**  
If the product still have some non-achievable features then it will be checked in this stage and then the product is passed to the Sprint Retrospective stage.

**Sprint Retrospective:**  
In this stage quality or status of the product is checked.

**Product Backlog:**  
According to the prioritize features the product is organized.

**Sprint Backlog:**  
Sprint Backlog is divided into two parts Product assigned features to sprint and Sprint planning meeting.

**Case study#1: Distributed Scrum Project for Dutch Railways**

*Scrum provides a proven foundation for the execution of projects. However, in every project the Scrum process must be adapted to address specific needs and circumstances. How this is done is a large factor in the success or failure of a project. Scrum project, one which had already been scrapped once under a traditional approach, and which included developers in both India and the Netherlands. In the hope of helping readers run successful, large Scrum projects, we share here our lessons learned on: project startup, getting the right product owner, the importance of estimates, effective communication, testing and documentation.*

These are the most important lessons we learned:

·         It can be hard to find a product owner with both detailed knowledge of the requirements as well as the mandate to set priorities. Often it is unavoidable to fill the product owner role with multiple people, especially in large projects.

·         When meeting a deadline is important, it's important to make sure that the product backlog is complete and estimated. For requirements, any estimate is better than no estimate, even if little information is available. In combination with the team's velocity, this provides the necessary information for release planning.

·         Scrum is well-suited to execution with multiple distributed teams. Having each Scrum team contain resources both in the Netherlands and India was good for team spirit and forced us to work on effective communication. For communication, off-the-shelf hardware and free software make the cost of implementing this low.

·         It is useful to start a distributed project with an initial co-located session to reach agreement on team practices.

·         Work that doesn't fit well in a Scrum Sprint (i.e. chasing down key people, interfacing with other customer departments) can be handled more effectively by a separate team. This allows the feature teams to focus on building the software. Using a dedicated technical writer also helps in this respect, even if it does add communication overhead.

·         Although not needed for the software development process, extensive requirements documentation may still be required by the customer. In a Scrum project, however, this cannot replace the use of user stories. If both are used, the overhead of reconciling requirements in two places should be factored into planning.

·         Automated testing is vital to deliver software incrementally, unhindered by regression bugs. Before the project is over, the return on investment will outweigh the cost.