SDLC Viva-Voice

1)what is the agile manifesto how it controls the product development?

A) The Agile Manifesto is a simple set of principles created in 2001 to improve the way software is developed.It focuses on teamwork, flexibility, and delivering useful software quickly. Instead of strictly following heavy processes and long documentation, it values people, communication, working software, customer collaboration, and the ability to adapt to change. In product development, it controls the process by breaking the work into small parts, delivering working features in short cycles, collecting feedback from customers regularly, and making improvements step by step. This approach helps teams respond quickly to changing requirements and ensures the final product truly meets customer needs rather than just following a fixed plan.

2)Explain and analyse the differnet between burn-down and burn-up?

A) A burn-down chart and a burn-up chart are both Agile tools used to track project progress, but they show information in different ways. A burn-down chart shows how much work is remaining over time. The line starts at the total amount of work and moves downward as the team completes tasks. It helps teams see whether they are on track to finish the sprint or project on time. If the line is not going down as expected, it clearly shows delay or slow progress.

B) A burn-up chart, on the other hand, shows how much work has been completed over time. It usually has two lines: one for total work and one for completed work. The completed work line moves upward as tasks are finished. It is more useful when the project scope changes, because if new work is added, the total work line increases and makes the change clearly visible.

3)How do you handle change in requriements in agile projects?

A) In Agile projects, changes in requirements are expected and welcomed because customer needs can evolve over time. When a change comes, the first step is to discuss it clearly with the customer or product owner to understand the exact need and impact. Then the new requirement is added to the product backlog and prioritized based on business value. The team does not disturb the current sprint once it has started, unless the change is very critical. Instead, the new change is planned for the next sprint during sprint planning. The team also estimates the effort required and adjusts timelines if needed. Continuous communication, regular feedback, and short development cycles help manage changes smoothly without affecting the overall quality of the product.

4)How do you handle a sistuation where testing is not compleated within script?

A) If testing is not completed within the sprint, first I analyze the reason clearly, whether it is due to late requirement changes, development delays, environment issues, unclear test cases, or underestimation of effort. Then I communicate transparently with the team and the product owner during the daily stand-up or review meeting so everyone understands the situation. In Agile, incomplete testing work is not forcefully closed; instead, the remaining testing tasks are moved back to the product backlog and prioritized for the next sprint. If the issue is frequent, I work on improving estimation, breaking tasks into smaller parts, starting testing early in the sprint, and increasing collaboration between developers and testers. The main focus is to maintain quality rather than rushing testing just to meet the sprint deadline.

5)How do you estimate tickets?

A) In Agile, tickets are usually estimated based on effort, complexity, and uncertainty rather than exact hours. First, the team clearly understands the requirement by discussing it with the product owner. Then we break the ticket into smaller tasks if needed. After that, we estimate using story points, which measure how difficult the task is compared to other tasks. We consider factors like technical complexity, dependencies, risks, and testing effort. Estimation is usually done using team discussion methods like planning poker so everyone shares their view and we agree on a final number. Over time, the team uses past sprint performance, called velocity, to improve estimation accuracy and plan work more realistically.

6)what do you do if when your automation suite is failed?

A) If my automation suite fails, first I do not panic and immediately check the test report and logs to understand the exact reason for failure. I verify whether the failure is due to a real application defect, test script issue, environment problem, test data issue, or timing and synchronization problem. If it is a genuine bug in the application, I reproduce it manually and report it clearly with proper evidence. If the failure is due to flaky tests, locator changes, or synchronization issues, I fix the script by improving waits, updating locators, or stabilizing the test logic. I also check whether the environment was stable during execution, such as server downtime or network issues. After fixing the issue, I re-run the failed tests to confirm stability. Finally, I analyze patterns of failure to improve framework reliability and prevent similar issues in the future.