1. **Differences in between Smoke & Sanity testing**

**A. Smoke testing**

The main purpose of smoke testing is to evaluate the application's general stability. It entails performing a cursory and superficial analysis of crucial functionality to make sure the key components are functional. Smoke testing is usually carried out following the delivery of a software build with the goal of locating any significant problems or show-stopper flaws that can hinder additional testing. It is often completed by the testing team and offers a high-level overview of the application's preparedness for more detailed testing phases. Automation may be used to increase efficiency.

**B. Sanity testing**

The main goal of sanity testing is to verifying particular functionalities or modules following recent upgrades, bug fixes. Sanity testing goes deeper into certain aspects of the program to make sure that recent changes haven't negatively impacted already-existing functionality, in contrast to Smoke Testing, which covers a wider range of applications. It is carried out as a focused check to ensure that the most recent modifications haven't created any new problems in connected or unconnected locations and also check the delivery of a new release or the application of bug fixes. Developers may be involved in Sanity Testing in addition to the testing team, particularly when changes need to be validated right away. Sanity testing, in general, offers a more thorough analysis of particular functionality, adding to the software's overall stability and dependability.

1. **Describe the Differences between Validation & Verification**

**A. Verification**

Verification is the process of assessing software to make sure it complies with the criteria and requirements that are set out. To ensure that the program is being constructed accurately in accordance with its planned design, this entails examining plans, document, and code. The goal of verification operations, which take place before the program is completely constructed during the development stage, is to find errors as soon as possible. Verification seeks to minimize the possibility of problems extending into subsequent phases of development by ensuring that the software satisfies quality standards and requirements defined by stakeholders through the use of reviews, inspections, and static analysis.

**B. Validation**

validation is the process of assessing software to make sure that it satisfies the requirements and expectations of its users. In order to confirm the software's usability, performance, and functioning, testing is required. After the software has been produced, validation activities take place during the testing phase with the aim of verifying that the program is suitable for its intended use. Validation employs several test methods such as functional, non-functional, and user acceptability testing (UAT) to verify that the program meets end-user requirements and delivers the intended functionality and performance. Validation seeks to confirm that the program satisfies the needs and expectations of the user, guaranteeing its usability, efficacy, and general quality.

1. **Explain about Agile Methodology.**

Agile methodology is an incremental, iterative approach to software development that prioritizes client satisfaction, adaptability, and teamwork. It was created as a substitute for conventional, linear development techniques like Waterfall with the intention of more effectively providing value to clients and adjusting to changing requirements.

1. **Explain about Epic and User Stories?**

**A. EPIC**

An epic is an extensive collection of work that can be divided into user stories or smaller tasks. Large and high-level features or functionality that may span several sprints or iterations are often represented by epics. They are frequently employed to record general user demands or business objectives that are difficult to accomplish in a single iteration.

With the help of epics, teams may prioritize and arrange their work so they can concentrate on adding value little by little while maintaining an overall perspective. Examples are "User Authentication System," and "Mobile App Redesign".

**B. USER STORIES**

Compared to epics, user stories are more manageable and smaller work units that are usually linked to a single sprint or iteration. They record certain user requirements, actions, or interactions with the software. This approach aids in making sure that providing value to the end-user always comes first. User stories provide a clear picture of what needs to be accomplished and why by serving as the framework for development tasks, conversations, and testing activities. Examples of user stories would be, "I want to see the payment gateway in the home page so that I can make payment easily."