1. **What is the diff between the TDD and VDD?**

TDD (Test-Driven Development) and VDD (Validation-Driven Development) are both software development methodologies that emphasize testing and validation, but they have different approaches and focuses.

**Test-Driven Development (TDD)**:

***Process***: In TDD, developers write tests before they write the corresponding code. The cycle typically follows these steps: write a test, run the test (which fails initially because the code isn't implemented yet), write the minimum amount of code necessary to pass the test, run the test again (which should pass now), and finally, refactor the code if needed while ensuring that all tests still pass.

***Focus***: TDD focuses on the development process and design of the software. It helps in driving the software design by continuously testing small units of code, ensuring that they meet the desired functionality before moving on to the next part of the code.

***Benefits:*** TDD can lead to well-structured, modular code with high test coverage. It helps in catching bugs early in the development process and encourages a rapid feedback loop, allowing developers to iterate quickly.

Validation-Driven Development (VDD):

***Process***: VDD, on the other hand, focuses on validating the software against the requirements and user expectations. It involves writing tests or validation criteria based on the software requirements or user stories. These tests are used to validate the software once it's fully developed.

***Focus***: VDD focuses on the end product and ensuring that it meets the specified requirements and user expectations. It involves validating the entire system or software application rather than individual units of code.

***Benefits***: VDD helps in ensuring that the final product aligns with the intended purpose and requirements. It provides a systematic way to validate the software, reducing the likelihood of delivering a product with unmet requirements or unexpected behavior.

In summary, TDD is primarily concerned with the development process and focuses on testing individual units of code, while VDD is concerned with validating the final product against the specified requirements and user expectations. Both methodologies are important in software development, and they can be used in conjunction to ensure both the development process and the final product are of high quality.

2. **Which are the Different agile ceremonies that you are part of** ?

In Agile software development, there are several ceremonies or meetings that are commonly practiced to ensure effective communication, collaboration, and transparency within the development team. While I'm not a direct participant in these ceremonies, I can certainly provide information about them:

Sprint Planning: At the beginning of each sprint, the team meets to plan the work to be done during the sprint. They decide which user stories or tasks to work on and how much work they can commit to completing.

Daily Standup (Daily Scrum): The team holds a brief daily meeting to discuss progress, plans for the day, and any obstacles they're facing. Each team member answers three questions: What did I do yesterday? What will I do today? Are there any obstacles in my way?

***Sprint Review***: At the end of a sprint, the team conducts a review meeting to demonstrate the work completed during the sprint to stakeholders and receive feedback. This helps in gathering input for future sprints.

***Sprint Retrospective***: Also held at the end of a sprint, the retrospective is a meeting where the team reflects on their performance during the sprint. They discuss what went well, what didn't, and how they can improve their processes and teamwork for the next sprint.

***Backlog Refinement (Grooming)***: This is an ongoing process where the team reviews and refines the items in the product backlog. It involves breaking down large user stories, adding details, and estimating the effort required to complete them.

***Release Planning***: A higher-level planning meeting where the team discusses and plans for upcoming releases. It involves aligning the development work with the overall project goals and timelines.

***Product Owner Meetings***: The Product Owner, a key role in Agile, often has several meetings such as stakeholder meetings, requirements discussions, and feedback sessions to ensure that the product backlog is well-maintained and reflects the priorities of the stakeholders.

***Big Room Planning:*** This is a larger-scale planning event where multiple teams get together to plan their work collectively, especially in large Agile frameworks like SAFe (Scaled Agile Framework).

Remember that the exact ceremonies and their names can vary based on the specific Agile framework being used (e.g., Scrum, Kanban, XP) and the organization's adaptations of these frameworks to suit their needs.

**3. What is the difference between sprint backlog & product backlog ?**

In Agile development, both Sprint Backlog and Product Backlog are artifacts used for managing work, but they serve different purposes and have distinct characteristics:

**Product Backlog:**

* **Definition:** The Product Backlog is a prioritized list of all desired work on the project. It contains features, enhancements, bug fixes, and other types of work that provide value to the product.
* **Ownership:** The Product Backlog is owned and managed by the Product Owner, who is responsible for ensuring that it reflects the stakeholders' priorities and contains the most valuable items at the top.
* **Scope:** It encompasses all the work that is anticipated to be done on the project, both short-term and long-term. Items at the top are usually more detailed and closer to implementation, while items at the bottom can be broader and less defined.
* **Prioritization:** Items in the Product Backlog are prioritized based on their value to the product and the needs of the users and stakeholders. The Product Owner continually refines and reprioritizes the backlog based on changing requirements and feedback.

**Sprint Backlog:**

* **Definition:** The Sprint Backlog is a subset of the Product Backlog. It includes the list of tasks, user stories, and other work items selected by the development team for implementation during a specific sprint.
* **Ownership:** The Sprint Backlog is owned by the development team. They decide which items from the Product Backlog to include in the Sprint Backlog based on their capacity and understanding of the work.
* **Scope:** It contains the work that the team commits to completing during the upcoming sprint. These items are broken down into smaller tasks and are typically more detailed than the items in the Product Backlog.
* **Time Frame:** The Sprint Backlog covers the work planned for a single sprint, which is a time-boxed iteration usually lasting between one to four weeks.

In summary, the Product Backlog is the overall to-do list for the project, while the Sprint Backlog is a subset of the Product Backlog containing the tasks to be completed within a specific sprint. The Sprint Backlog is a dynamic document that evolves as the team works through the sprint, while the Product Backlog is continuously refined and reprioritized based on changing requirements and stakeholder feedback.

**4.What is smoke test ?**

A smoke test is a type of software testing that is conducted to check whether the basic functionalities of a software application are working properly. The term "smoke testing" comes from the electronics industry, where a device would be turned on and observed for smoke. If it passed this test, it was deemed safe to run further tests.

In the context of software development, a smoke test is usually performed after a new build or version of the software is deployed. It is a preliminary test to reveal simple failures severe enough to reject a prospective software release. The purpose is to determine whether the application is stable enough for more detailed testing. During a smoke test, testers verify that the major components of the software work without encountering critical errors.

If the smoke test fails, it indicates that the build is too unstable for further testing, and developers need to fix the critical issues before more comprehensive testing can proceed. If the smoke test passes, more in-depth testing such as functional testing, integration testing, and regression testing can be carried out to ensure the software's quality and reliability.

A smoke test is also commonly referred to as a "build verification test" or "build acceptance test." These terms are often used interchangeably to describe the preliminary testing conducted on a software build to ensure that it is stable enough for more comprehensive testing.

**5.What is sanity testing?**

Sanity testing, also known as sanity check or build verification testing, is a subset of regression testing. It is a quick round of testing to ensure that the most critical functionalities of an application or software are working as expected after a minor change or a bug fix.

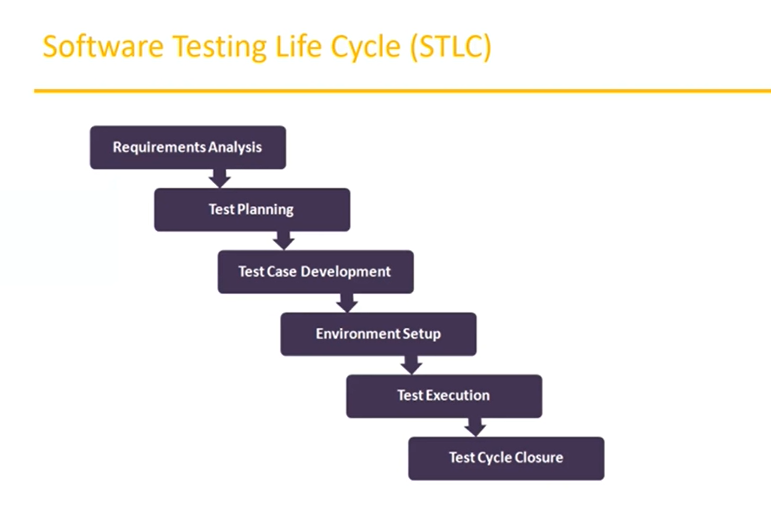
The primary goal of sanity testing is not to perform exhaustive testing but to verify that the key functionalities of the software are still operational after a specific change has been made or after a new build has been deployed. It helps in ensuring that the recent modifications haven't adversely affected the core features or functionalities of the system.

Key points about sanity testing:

* **Focused Testing:** It focuses on specific areas or functionalities rather than conducting comprehensive testing.
* **Quick and Selective:** It's rapid and targeted, aiming to cover critical functionalities.
* **Limited Scope:** It doesn’t cover all aspects of the system; instead, it concentrates on the most crucial parts.
* **Non-Scripted Approach:** Test cases might not be pre-scripted; it's more ad-hoc to verify core functionalities.
* **Decision-making:** It assists in making a quick decision about whether the software is stable enough for further, more detailed testing.

For example, if a bug was fixed in the login functionality of a website, a sanity test would ensure that users can log in successfully and access their accounts after the fix without extensively testing every feature of the website.

Sanity testing helps in identifying major issues quickly, allowing teams to decide whether to proceed with more comprehensive testing or development activities based on the results obtained from this rapid assessment.

*   
  **Black-box testing in software testing**: In black-box testing, the system is tested only in terms of its external behaviors; it does not consider how the software functions on the inside. This is the only limitation of the black-box test. It is used in Acceptance Testing and System Testing.
* **White-box testing in software testing:** A white-box test is a method of testing a program that takes into account its internal workings as part of its review. It is used in integration testing and unit testing.
* **Grey-box testing in software testing:** A Gray Box Testing technique can be characterized as a combination of a black box as well as a white box testing technique used in the software testing process. Using this technique, you can test a software product or application with a partial understanding of its internal structure.