Software Testing

1) What is SDLC?

Ans: SDLC Stands for a Software Development Life Cycle. It is a methodology with well-defined stages that enables developers to create high-quality software with given budget and timeline.

2) What is software testing?

Ans: Software Testing is Method to check the Correctness, Completeness and Quality of the Developed Software.

3) What is agile methodology?

Ans: Agile Methodology is a combination of Iterative and Incremental model. It is mainly focus on a customer satisfaction and Rapid delivery of a product.

4) What is SRS?

Ans: SRS Stands for Software Requirement Specification. It is a document that capture complete description about how system is expected to perform.

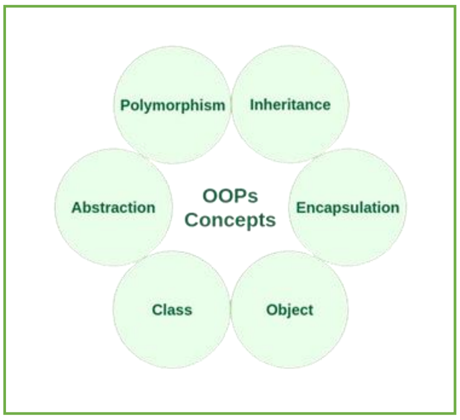
5) What is oops?

Ans: Oops Stands for Object oriented programming. It is mainly use for structure of software program simple code.

6) Write Basic Concepts of oops?

Ans: Below are basic concept of oops.

1. Class :
   Class is a collection of data member and member functions.
2. Object :
   Objects are the basic unit of OOPS representing real-life entities.
3. Encapsulation :
   Encapsulation is a means of binding data variables and methods together in a class. Only objects of the class can then be allowed to access these entities.
4. Inheritance :
   Inheritance is the process by which one class inherits the functions and properties of another class.
   Types of Inheritance:
   - Single Level Inheritance
   - Multi Level Inheritance
   - Multiple Inheritance
   - Hierarchical Inheritance
   - Hybrid Inheritance
5. Polymorphism :
   In a polymorphism means same name but different forms. It's a Compile time and Run time Polymorphism its called method overloading and method overriding.
6. Abstraction :
   Abstraction means showing only the relevant details to the end-user and hiding the irrelevant features that serve as a distraction.



7) What is object?

Ans: Object is a real-world entity. It is create memory for a class. It is access by its properties called data member and member function.

8) What is class?

Ans: Class is a collection of a data member and member function. Class is a blueprint or a template to describe the property and behaviour of the object.

9) What is encapsulation?

Ans: A wrapping up a data and function in a single unit is called encapsulation. In this important information inside an object and selected information is exposed.

10) What is inheritance?

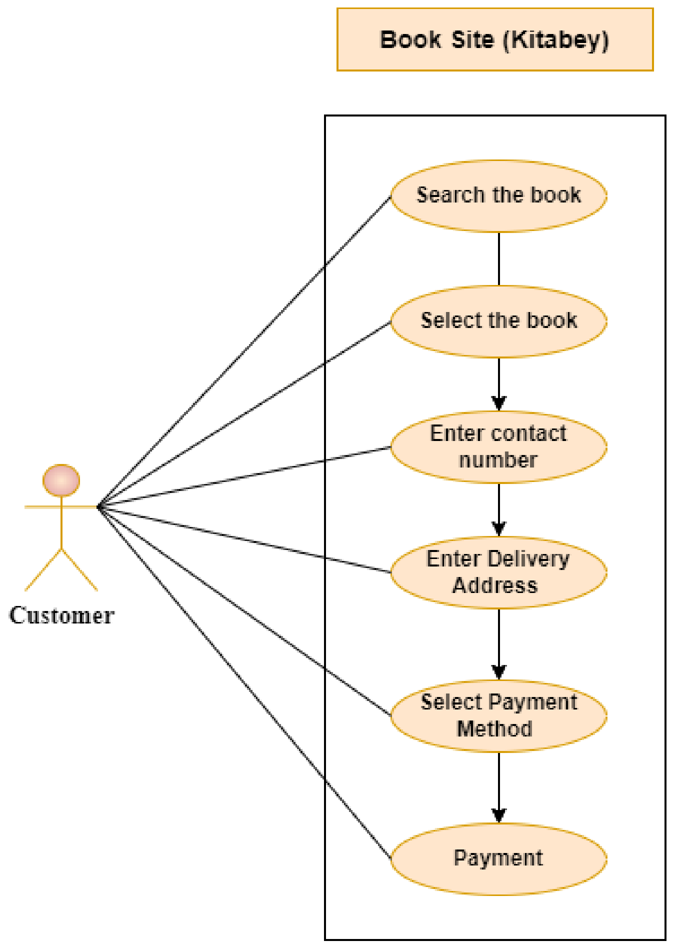
Ans: It is inherit the property of a parent class to child class and continue this cycle that’s call inheritance.

11) What is polymorphism?

Ans: In a polymorphism means same name but different forms. It’s a Compile time and Run time Polymorphism its called method overloading and method overriding.

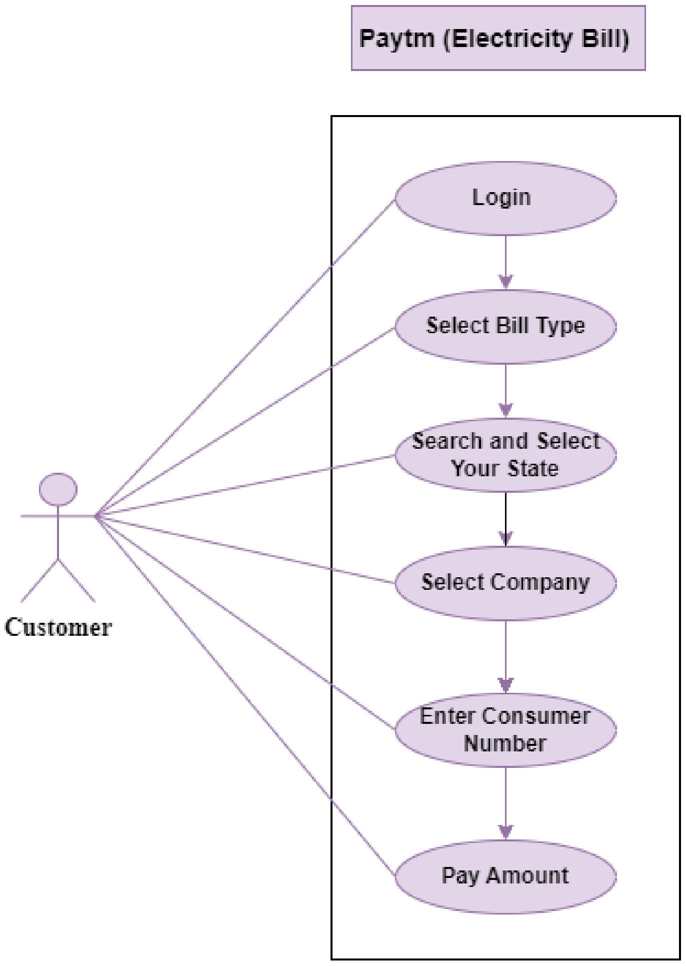
12) Draw Usecase on Online book shopping (Kitabey)?

Ans:



13) Draw Usecase on online bill payment system (Paytm)?

Ans:



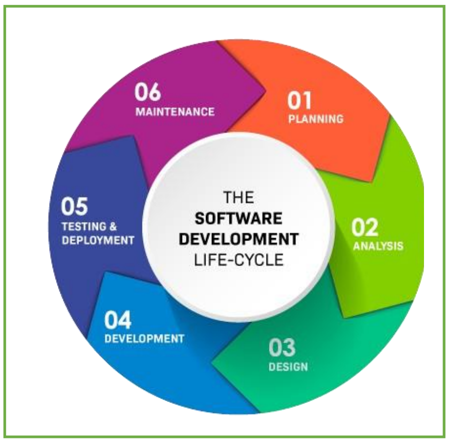
14) Write SDLC phases with basic introduction?

Ans: Follow this Phase to build a Software.

I. Planning:
   - We always do the plan before we start the new thing and we plan at that ariase the issues.
   - In this phase we gathering the requirement of the user and at that time phase many things like Lack of Clarity, Requirement Confusions etc.

II. Analysis:
   - In this phase defines the problem that the customer is trying to solve.
   - Ideally, this document states in a clear and precise fashion what is to be built.
   - It also Connected with the SRS. It make a fully document of the software. In this document describe "what we are maiking ? " Or " how it is making" in detail.

III. Design:
   - In this phase we make a design a architecture of the software. Requirement define in SRS and after check multiple design for the product architecture.
   - The requirement document must guide this decision process.



- Example: When we search on browser and open any website or app at that time we can see the first that's all are design accroding to the software requitements.

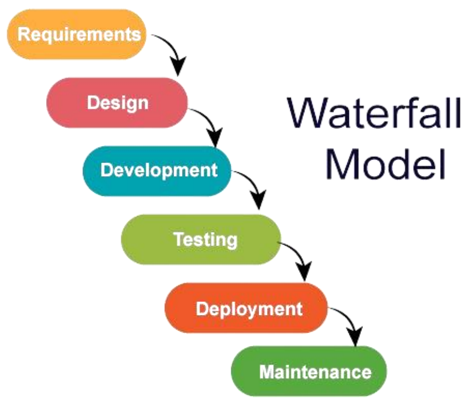
IV. Development:
- Given the architecture document from the design phase and the requirement document from the analysis phase, the team should build exactly what has been requested, though there is still room for innovation and flexibility.
- The implementation phase deals with issues of quality, performance, baselines, libraries, and debugging.

V. Testing & Deployment:
- The testing phase is a separate phase which is performed by a different team after the implementation is completed.
- Documentation also provides information about how to use the product.
- These bugs are then sent back to the developers for rectification. Once the required fixes are implemented, the software re-enters the testing phase for validation.
- After crafting a product with precision, it's time to present it to the users by pushing to the production environment.

VI. Maintanance:
- Maintenance is the process of changing a system after it has been deployed.
  Corrective maintenance: identifying and repairing defects
  Adaptive maintenance: adapting the existing solution to the new platforms.
  Perfective Maintenance: implementing the new requirements
- In a spiral lifecycle, everything after the delivery and deployment of the first prototype can be considered "maintenance"!
- The maintenance phase also considers long-term strategies, for instance, upgrading or replacing the software. This decision depends on the software's lifecycle and technological progress. Similar to a homeowner contemplating a renovation or selling their house, the software might require a complete revamp or phase-out to stay relevant and valuable.

15) Explain Phases of the waterfall model?

Ans:



I. Requirement Gathering and Analysis:
- The aim of this phase is to understand the exact requirements of the customer and to document them properly.
- Both the customer and the software developer work together so as to document all the functions, performance, and interfacing requirement of the software.
- It describes the "what" of the system to be produced and not "how."
- In this phase, a large document called Software Requirement Specification (SRS) document is created which contained a detailed description of what the system will do in the common language.

II. System Design:
- This phase aims to transform the requirements gathered in the SRS into a suitable form which permits further coding in a programming language.
- It defines the overall software architecture together with high level and detailed design.
- All this work is documented as a Software Design Document (SDD).

III. Implementation:
- As the name implies, in this phase the source code is written as per requirements.
- The physical design specifications are turned into a working code.
- The system is developed in small programs called units, after which these units are integrated.

IV. Testing:
- The code is then handed over to the testing team.
- Testers check the program for all possible defects, by running test cases either manually or by automation.
- The client is involved in the testing phase as well, in order to ensure all requirements are met.
- All Flaws and bugs detected during this phase are fixed to ensure Quality Assurance.

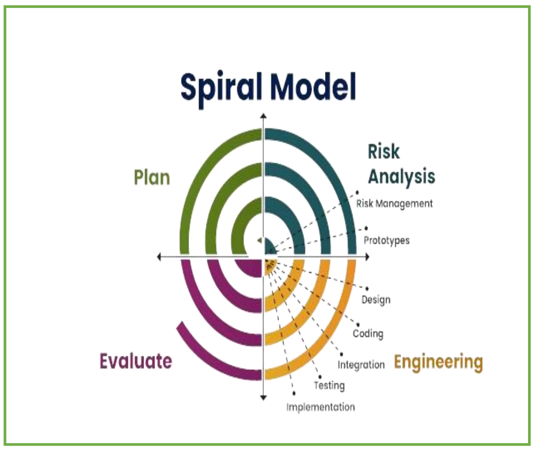
V. Deployment:

In this phase, the software is deployed into a live environment (client's server) in order to test its performance. Once the software is deployed, it becomes available to end-users. Sometimes, this phase also includes training of real-time users to communicate benefits of the system.

VI. Maintenance:
- The final phase of the Waterfall Model is maintenance, which involves fixing any issues that arise after the software has been deployed and ensuring that it continues to meet the requirements over time.

16) Write phases of spiral model?

Ans: Software developers employ the spiral model, which is preferred for complex, large-scale projects. In the spiral model errors or risks are identified and rectified earlier.



I. Plan
- This initial phase involves defining project objectives, risks, and constraints.
- The planning phase aims to establish a clear understanding of what the software should achieve and the potential challenges that may arise.

II. Risk Analysis
- In this phase, the development team assesses and mitigates project risks.
- Risk analysis is helps identify potential problems early in the development process.
- The Spiral model cannot eliminate all risks from a project, but through risk analysis, it can help reduce the risk before it comes.

III. Engineering
- The engineering phase focuses on designing, developing, and testing the software.
- Customer feedback is an important part of the engineering phase of the Spiral software development process model.
- Whether it's architectural design, physical product design, system requirements, or other aspects of the project, customer feedback is valuable to the ultimate success of the project.

IV. Evaluate

- The evaluation phase involves reviewing the progress made so far. It includes a formal assessment of the software's current state.
- The defining feature of the Spiral model is its iterative nature. After completing one cycle through these four phases.
- At the end of the first iteration, the client analyses the program and comments after testing it.

17) Write agile manifesto principles?

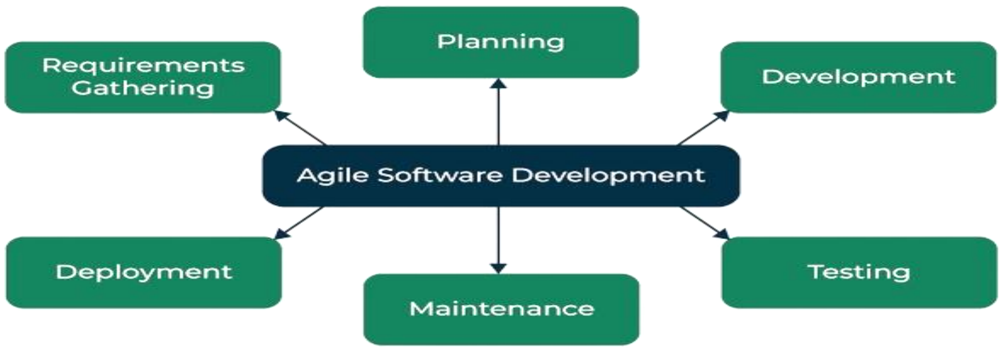
Ans: These are the principal of agile manifesto.

A. Individuals and interactions over processes and tools:
   This value of the Agile manifesto focuses on giving importance to communication with the clients. There are several things a client may want to ask and it is the responsibility of the team members to ensure that all questions and suggestions of the clients are promptly dealt with.
B. Working software over comprehensive documentation:
   In the past, more focus used to be on proper documentation of every aspect of the project. There were several times when this was done at the expense of the final product. The Agile values dictate that the first and foremost duty of the project team is completing the final deliverables as identified by the customers.
C. Customer collaboration over contract negotiation:
   Agile principles require customers to be involved in all phases of the project. The Waterfall approach or Traditional methodologies only allow customers to negotiate before and after the project. This used to result in wastage of both time and resources. If the customers are kept in the loop during the development process, team members can ensure that the final product meets all the requirements of the client.
D. Responding to change over following a plan:
   Contrary to the management methodologies of the past, Agile values are against using elaborate plans before the start of the project and continue sticking to them no matter what. Circumstances change and sometimes customers demand extra features in the final product that may change the project scope. In these cases, project managers and their teams must adapt quickly in order to deliver a quality product and ensure 100% customer satisfaction.

18) Explain working methodology of agile model and also write pros and cons?

Ans: Agile methods are being widely accepted in the software world recently. The Agile thought process had started early in the software development and started becoming popular with time due to its flexibility and adaptability.

Customer Interaction is the backbone of this Agile methodology, and open communication with minimum documentation are the typical features of Agile development environment.



I. Requirements gathering: In this phase, you must define the requirements. You should explain business opportunities and plan the time and effort needed to build the project. Based on this information, you can evaluate technical and economic feasibility.

II. Planning: In this phase defines the problem that the customer is trying to solve. Ideally, this document states in a clear and precise fashion what is to be built. It also Connected with the SRS. It make a fully document of the software. In this document describe "what we are maiking ? " Or " how it is making" in detail.

III. Development: Given the architecture document from the design phase and the requirement document from the analysis phase, the team should build exactly what has been requested, though there is still room for innovation and flexibility. The implementation phase deals with issues of quality, performance, baselines, libraries, and debugging.

IV. Testing: In this phase, the Quality Assurance team examines the product's performance and looks for the bug.

V. Maintenance: Maintenance is the process of changing a system after it has been deployed.

Corrective maintenance: identifying and repairing defects

Adaptive maintenance: adapting the existing solution to the new platforms.

Perfective Maintenance: implementing the new requirements

In a spiral lifecycle, everything after the delivery and deployment of the first prototype can be considered "maintenance"!

VI. Deployment: In this phase, the team issues a product for the user's work environment.

Pros of Agile Model:

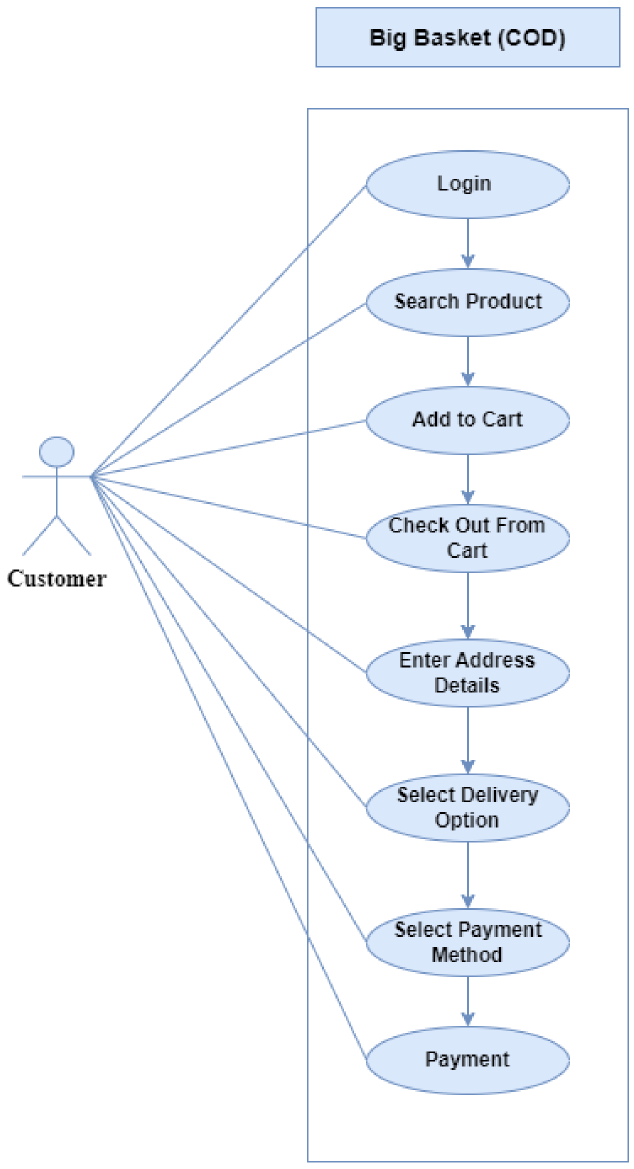
- Promotes teamwork and cross training.
- Functionality can be developed rapidly and demonstrated.
- Resource requirements are minimum.
- Suitable for fixed or changing requirements
- Delivers early partial working solutions.
- Good model for environments that change steadily.
- Minimal rules, documentation easily employed.
- Little or no planning required.
- Easy to manage.
- Gives flexibility to developers

Cons of Agile Model:

- Not suitable for handling complex dependencies.
- More risk of maintainability.
- An overall plan, an agile leader and agile PM practice is a must without which it will not work.
- Strict delivery management dictates the scope, functionality to be delivered, and adjustments to meet the deadlines.
- Depends heavily on customer interaction, so if customer is not clear, team can be driven in the wrong direction.
- There is a very high individual dependency, since there is minimum documentation generated.
- Transfer of technology to new team members may be quite challenging due to lack of documentation.

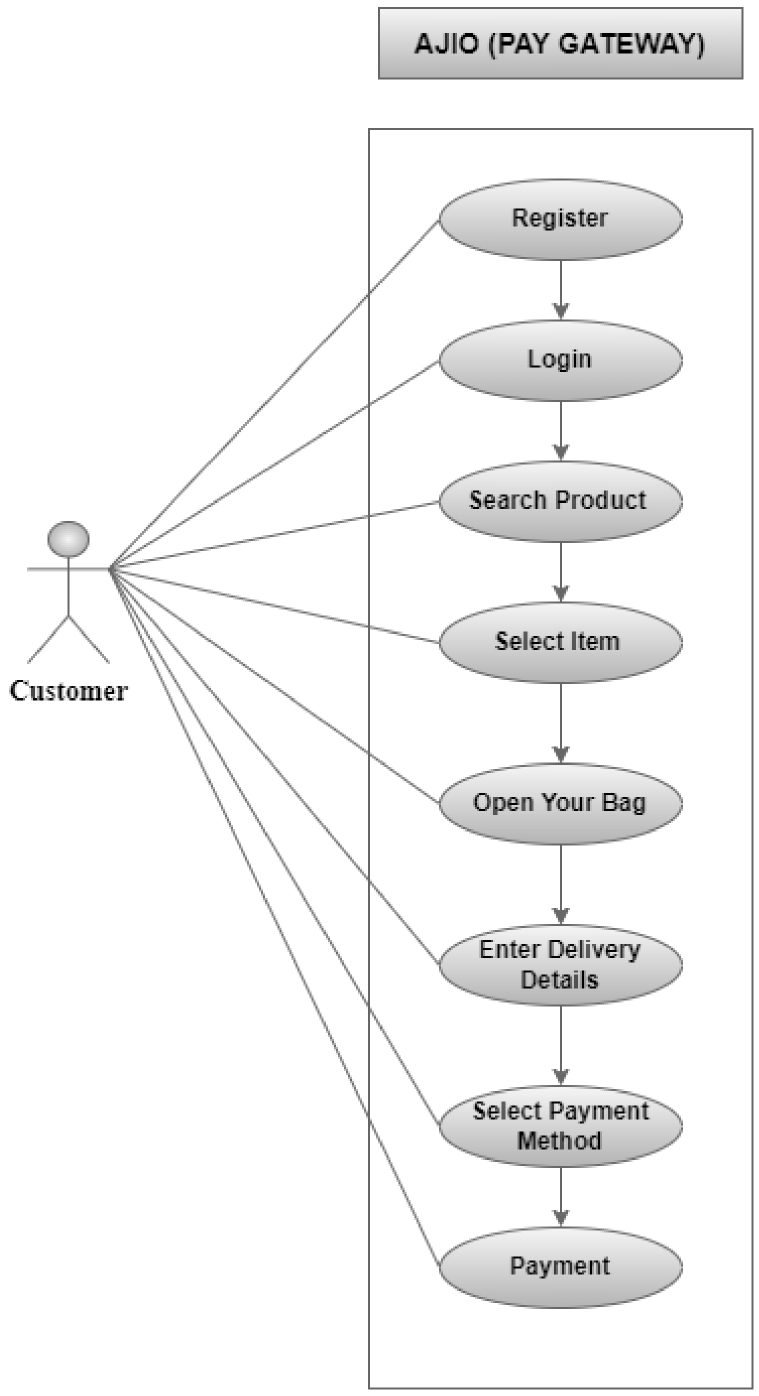
19) Draw Usecase on Online shopping product using COD?

Ans:



20) Draw Usecase on Online shopping product using payment gateway?

Ans:



21) What is 7 key principles? Explain in detail?

Ans: Below-mentioned are the principles of software testing:

I. Testing shows the presence of defects
- The goal of software testing is to make the software fails. Software testing reduces the presence of defects.
- Software testing talks about the presence of defects and doesn't talk about the absence of defects.

II. Exhaustive testing is not possible
- It is the process of testing the functionality of the software in all possible inputs (valid or invalid) and pre-conditions is known as exhaustive testing.
- If the software will test every test case, then it will take more cost, effort, etc., which is impractical.

III. Early testing
- To find the defect in the software, early test activity shall be started.
- The defect detected in the early phases of SDLC will be very less expensive.

IV. Defect clustering
- In a project, a small number of modules can contain most of the defects. The Pareto Principle for software testing states that 80% of software defects come from 20% of modules.

V. Pesticide paradox
- Repeating the same test cases, again and again, will not find new bugs. So, it is necessary to review the test cases and add or update test cases to find new bugs.

VI. Testing is Context-Dependent
- The testing approach depends on the context of the software developed.
- Different types of software need to perform different types of testing.
- For example, the testing of the e-commerce site is different from the testing of the Android application.

VII. Absence of Errors fallacy
- If a built software is 99% bug-free but does not follow the user requirement then it is unusable.
- It is not only necessary that software is 99% bug-free but it is also mandatory to fulfil the customer requirements.

22) Difference between verification and Validation?

Ans:

Verification	**Validation**
Verification is a process which is performed at development level.	Validation is a process which is performed at testing level.
Verification involves examining documents, languages, designs, and code.	Validation involves the testing of the actual product.
Verification does not require code execution.	Validation requires code execution.
Verification is considered static testing.	Validation is considered dynamic testing.
Verification involves checking documents provided by humans.	Validation involves the execution of a program by a computer.
Verification activities are Reviews and Inspections.	Validation activity is Testing.