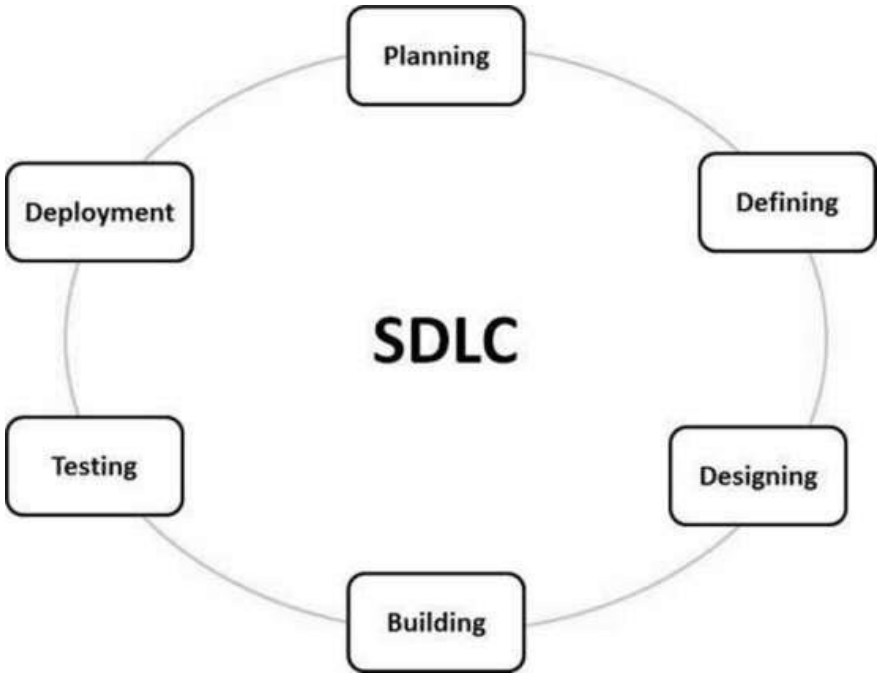
1. what is SDLC

- **Software Development Life Cycle** (SDLC) is a process used by the software industry to design, develop and test high quality softwares.
- The SDLC aims to produce a high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.
- SDLC is a process followed for a software project, within a software organization.
- It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software.
- The life cycle defines a methodology for improving the quality of software and the overall development process.
- The following figure is a graphical representation of the various stages of a typical SDLC.



2. What is software testing?

- Software testing can be stated as the process of verifying and validating whether a software or application is bug-free, meets the technical requirements as guided by its design and development, and meets the user requirements effectively and efficiently by handling all the exceptional and boundary cases.
- The process of software testing aims not only at finding faults in the existing software but also at finding measures to improve the software in terms of efficiency, accuracy, and usability.
- It mainly aims at measuring the specification, functionality, and performance of a software program or application.

➢ **Software testing can be divided into two steps:**

- **1. Verification:**
  it refers to the set of tasks that ensure that the software correctly implements a specific function.

- **2.Validation:**
  it refers to a different set of tasks that ensure that the software that has been built is traceable to customer requirements.

3. What is agile methodology ?

- Agile Methodology meaning a practice that promotes continuous iteration of development and testing throughout the software development lifecycle of the project.
- In the Agile model in software testing, both development and testing activities are concurrent, unlike the Waterfall model.

➢ **Agile Testing**

- Type of software testing that follows the principles of agile software development to test the software application.
- All members of the project team along with the special experts and testers are involved in agile testing.
- Agile testing is not a separate phase and it carried out with all the development phases i.e. requirements, design and coding and test case generation.
- Agile testing takes place simultaneously through the Development Life Cycle.
- Agile testers participate in the entire development life cycle along with development team members and the testers help in building the software according to the customer requirements and with better design and thus code becomes possible.
- Agile testing team works as a single team towards the single objective of achieving quality.
- Agile Testing has shorter time frames called iterations or loops.
- This methodology is also called delivery driven approach because it provides a better prediction on the workable products in less duration of time.

➢ Agile Testing Principles:

- **Shortening feedback iteration:**
  In Agile Testing, testing team get to know the product development and its quality for each and every iteration. Thus continuous feedback minimizes the feedback response time and the fixing cost is also reduced.

- **Testing is performed alongside:**
  Agile testing is not a different phase. It is performed alongside the

development phase. It ensures that the features implemented during that iteration are actually done. Testing is not kept in pending for a later phase.

- **Involvement of all members:**
  Agile testing involves the each and every member of the development team and the testing team. It include various developers and experts.

- **Documentation is weightless:**
  In place of global test documentation, agile testers use reusable checklists to suggest tests and focus on the essence of the test rather than the incidental details. Lightweight documentation tools are used.

- **Clean code:**
  The defects that are detected are fixed within the same iteration. This ensures clean code at any stage of the development.

➢ **Advantage :**

1. Cross training teamwork
2. Use for big project
3. Suitable for big project
4. Little or no planning requirement
5. It saves time and money
6. Agile testing reduces documentation
7. It is flexible and highly adaptable to changes
8. It provides a way for receiving regular feedback from the end user
9. Better determination of issues through daily meetings

➢ **Disadvantage:**

1. In Agile methodology the documentation is less.
2. Sometimes in Agile methodology the requirement is not very clear hence it's difficult to predict the expected result.
3. In few of the projects at the starting of the software development life cycle it's difficult to estimate the actual effort required.
4. Because of the ever-evolving features, there is always a risk of the ever-lasting project.

4. What is SRS?

- A **software requirement specification (**SRS**)** is a comprehensive information/description of a product/system to be developed with its functional and non-functional requirements.
- The **software requirement specification (**SRS**)** is developed based on the agreement between customer and supplier.
- It may include the use cases of how a user is going to interact with the product or software system.
- SRS helps to reduce the time and effort to develop software.
- To develop the software system we should clearly understand the Software requirements.
- To achieve this we need to continuously communicate with clients to gather all related information and requirements.

- A typical SRS document describes all the software requirements and sometimes even contains a collection of use cases that describe the user interactions needed by the software.
- It defines the purpose of a software project, provides the overall definition and specifications of its features.

➢ **In general, SRS documents contain three kinds of program requirements:**

- **Functional specifications** that include measures to be performed by the system
- **Non-functional requirements** determining the software system's performance attributes
- **Domain requirements** that are device limits on the service domain

5. What is Oops?

- **Object-Oriented Programming System (OOPs)** is a programming concept that works on the principles of abstraction, encapsulation, inheritance, and polymorphism. It allows users to create objects they want and create methods to handle those objects.
- The basic concept of OOPs is to create objects, re-use them throughout the program, and manipulate these objects to get results.

    - Data security
    - Memory management
    - Code debility

6. Write basic concepts of oops

1. **Class**

A class can be stated as the blueprint of the object.

This states what an object can do.

It contains all the features, attributes, and behavior of what the model can do.

We can say that class contains the definition of the object.

2. **Object**

Object is the entity that makes the classes to be implemented into the program.

It makes the features, properties, and behaviors of the class to be implemented.

3. **Inheritance**

Inheritance can be defined as the process where one (parent/super) class acquires the properties (methods and fields) of another (child/sub).

With the use of inheritance, the information is made manageable in a hierarchical order.

4. **Polymorphism**

Polymorphism is the ability of an object to perform different actions (or, exhibit different behaviors) based on the context.

5. **Abstraction**

Abstraction is a process of hiding the implementation details from the user, only the functionality will be provided to the user.

In other words, the user will have the information on what the object does instead of how it does it.

6. **Encapsulation**

Encapsulation in Java is a mechanism for wrapping the data (variables) and code acting on the data (methods) together as a single unit.

In encapsulation, the variables of a class will be hidden from other classes and can be accessed only through the methods of their current class.

Therefore, it is also known as data hiding.

7. What is object?

- Object is the entity that makes the classes to be implemented into the program.
- It makes the features, properties, and behaviors of the class to be implemented.

- **Example:**

- A car is an object that has property color, model, brand name, fuel type, etc, and behaviors like, it runs.
- So these properties and behavior make an object (CAR).

8. What is class

- A class can be stated as the blueprint of the object.
- This states what an object can do. It contains all the features, attributes, and behavior of what the model can do.
- We can say that class contains the definition of the object.

- **Example :**
- Car color, engine type, etc. And with the definition, we can create any number of objects.

9. What is encapsulation

- Encapsulation can be defined as the binding of data and attributes or methods and data members in a single unit.

- In classes, we have Data and attributes that perform operations on that data.

- So according to the OOPs principle of Encapsulation, that data can be merged into a single unit.

- Encapsulation enhances more security of the data as everything related to a single task must be grouped and access to the data is given as per need.

- And this can be achieved using the concept of **Data Hiding**.

- **Encapsulation = Data Hiding + Abstraction.**

- **Data Hiding –** It means hiding the data of the class and restricting access to the outside world.

- **Example :**

- Using the access specifier keywords like **private** that restricts the data to only accessible and modifiable in the same class. Outside users can not access the data.

10. What is inheritance

- Inheritance is the method of acquiring features of the existing class into the new class.

- Suppose there is a class, considered as the parent class that has some methods associated with it.

- And we declare a new class, considered as the child class that has its own methods.

- So, when a child class is inherited from the parent class then it will have all the methods associated with parent class are available in the child class along with the child class own methods also. So that is Inheritance.

- **Example:**

- We all have seen the updates which receive on gadgets both in terms of hardware and software.

- Like firstly mobile phones are their only to talk and then used for media access, Internet, Camera, etc.

- So these are evolution that arrives by adding some extra feature and making a new product without rebuilding the complete functionality so it is an example of inheritance.

- Similarly, in software, regular updates are with some new features are also the inheritance.

11. What is polymorphism

- Polymorphism is the most essential concept of the Object-Oriented Programming principle.

- It has meaning '**poly**' – many, '**morph**' – forms. So polymorphism means many forms.

- In Object-Oriented Programming, any object or method has more than one name associated with it. That is nothing but polymorphism.

  For Objects When any object can hold the reference of its parent object then it is a polymorphism.

- To understand this concept more clear, let's consider an example

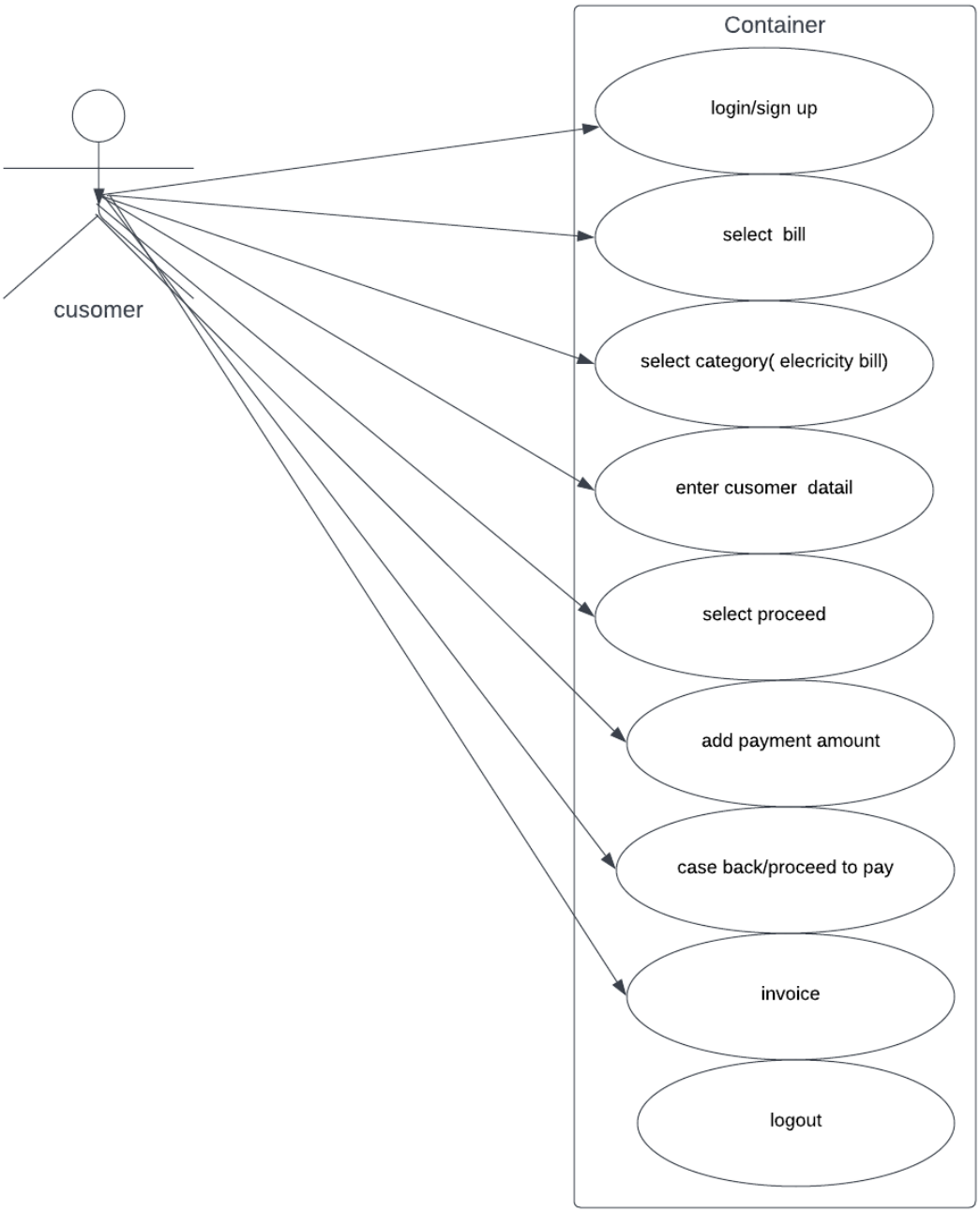
- **Example:**

- We have Television from different companies, Let's say Samsung TV, LG TV, Xiaomi TV, etc.

- Now if we have to call it then mostly we don't call it with its brand name. We simply call it a Television, and it can be of a different brand. But in general, we call it television. So this can be said to be polymorphism.

12. Draw user case on online book shopping

13. Draw user on online bill payment system (paytm)



14. Write SDLC phase with basic introduction

Phase 1

1. **Requirement:**

- In this phase, all the requirements are collected from the customer/client. They are provided in a document called Businessmen requirement specification (BRS) and System requirement specification (SRS). All the details are discussed with the customer/client in detail.

Phase 2

2. **Analysis:**

- As a Product Manager, this is probably the most important phase for you.
- This phase defines the importance of system development life cycle. This is also called the SDLC planning phase.
- This is where you do your market research, conduct customer interviews, research your competition and conduct surveys.
- The feedback that you gather helps you analyse your product market fit.
- This eventually leads us to the next stage, which is the feature definition stage.

Phase 3

3. **Design:**

- In this phase system design specification is prepared from the requirement document once the project is feasible, this design specification give input for the next phase of the model.

- Design is a blue print of the application and it helps in specifying hardware and requirements of the system and helps in defining architecture of the system.

Phase 4

4. Implementation:

- This is the longest phase.
- This phase consists of Front end + Middleware + Back-end.
- **In front-end:** Development of coding is done even SEO settings are done.
- **In Middleware:** They connect both the front end and back end.
- **In the back-end:** A database is created.

Phase 5

5. Testing:

- Testing is carried out to verify the entire system.

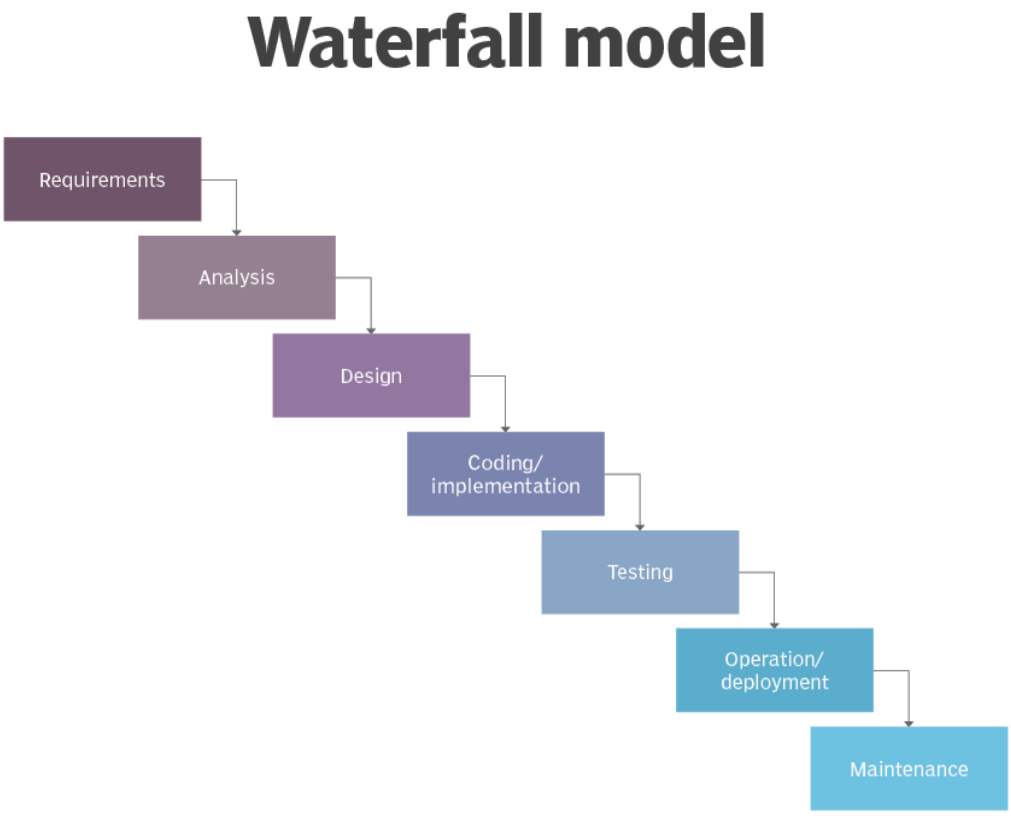
- The aim of the tester is to find out the gaps and defects within the system and also to check whether the system is running according to the requirement of the customer/client.

Phase 6

6. Maintenance:

- Once the product has been delivered to the client a task of maintenance starts as when the client will come up with an error the issue should be fixed from time to time.

15. Explain phase of the waterfall model



1. Requirements

- Specifically, we need to know and understand what we have to design, what we have to develop, its processes, what will be its functionality, etc. It provides input material to the product being made; thus, the upcoming product is studied, finalized, and marked. It also gives us the extension to decide the hardware or software requirements of the product which will be designed, developed, and captured in all phases.

2. Analysis

Not only is this requirement divided into two parts:

- **Requirement gathering and analysis:** First, all the information and requirement for the product development is gathered from the customer and processed for analysis. The primary role of this part is to eradicate incompleteness and inconsistencies related to software product development.

- **Requirement specification:** Then, the above-analysed requirements are documented in an SRS (software requirement specification) document. It serves as a path between the customer and the SRS development team. Any disputes in the future are managed and settled through this SRS documentation only.

3. Design

- After the first phase is completed and verified, it is the next most important phase to be studied as it is used for system design. It helps in specifying software and hardware requirements for the product design. It also helps in the overall architecture of the system design. So the requirement specification is mainly studied and verified in this phase. It is also helpful in transforming the SRS document into functional design and development of the software product. So we can say that in the designing phase, one makes the overall architecture for the software development project.

4. **Implementation**

- With system design fully verified, the implementation phase comes in a row. In this phase, the inputs from system design are taken, and it is first developed in small programs known as units, which are tested and implemented in the upcoming phase. Each unit in the implementation phase undergoes development, and its full functionality is tested, also known as unit testing. So in this phase, the system design is converted into source code with fully functional program modules. It includes the development, proving, and integration of the software.

5. **Integration and Testing**

- Each unit design and development in the earlier phases are incorporated from the implementation phase, which is integrated into a module or system for various tests like load test, load test, etc., after testing each unit. The testing environment undergoes a constant software check to determine if there is any flow or error in the design or code. Testing is done to maintain the stability and feasibility of the software so that the client does not face any disturbances or bugs during its production. So in this phase, the whole system is tested thoroughly for any faults and failures after implementation.

6. **Deployment of the System / Operations**

- Once the non-functional, functional, alpha, and beta testing are done, the software product is deployed to the user or customer system or released to the market. The

deployment phase includes installation, migration, and support of the complete system to the user or customer environment.

7. Maintenance

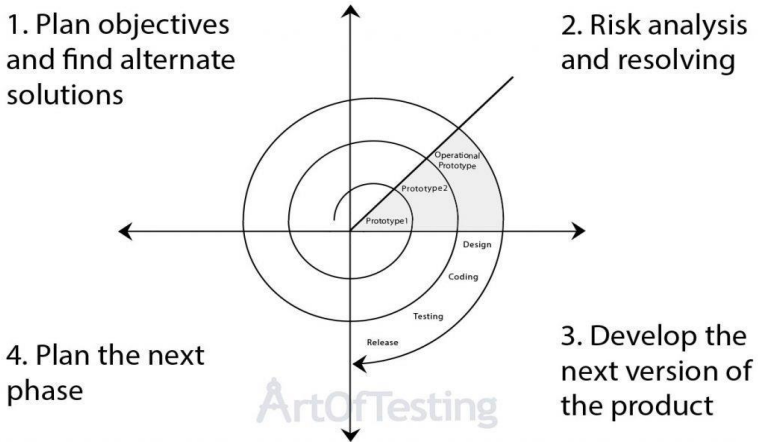
- It is the last but the most important phase in the waterfall workflow model. This step comes just after installation, and it includes making the appropriate modification to the product or system or enhancing, changing, or modifying attributes related to performance issues related to the system. Its main role is to improve the performance of the system with the maximum accuracy result of the software output.

- These changes raised during the maintenance phase are majorly related to initiated modifications to be done by the customer or users after the installation and testing phase, which includes bugs like defects uncovered during live uses of the system or requests raised by the customers. So the client is provided with timely and scheduled maintenance and support for the developed product. You will be amazed to know that the effort made in the design and development phase of the software product is only 60% compared to the efforts made in the maintenance phase.

16. Write phases of the spiral model

- The spiral model is an SDLC model that combines elements of an iterative software development model with a waterfall model. It is advisable to use this model for expensive, large and complex projects.

- In its diagrammatic representation, we have a coil having many cycles or loops. The number of cycles varies for each project and is usually specified by the project manager. Each spiral cycle is a stage in the software development process.



- The Spiral Model allows the product to be rolled out and refined in each phase of the spiral, with the ability to build prototypes in each stage. A prototype is created at the beginning of each phase as a risk management technique.

- The most important feature of the model is that once the project starts, it has the ability to manage unknown risks. Let's go through the different phases of the Spiral model first and after that, we would be able to see how risk is handled in this model.

➢ **Spiral Model Phases**

- It has four stages or phases: The planning of objectives, risk analysis, engineering or development, and finally review. A project passes through all these stages repeatedly and the phases are known as a Spiral in the model.

1. **Determine objectives and find alternate solutions: This** phase includes requirement gathering and analysis. Based on the requirements, objectives are defined and different alternate solutions are proposed.

2. **Risk Analysis and resolving:** In this quadrant, all the proposed solutions are analysed and any potential risk is identified, analysed, and resolved.

3. **Develop and test:** This phase includes the actual implementation of the different features. All the implemented features are then verified with thorough testing.

4. **Review and planning of the next phase:** in this phase, the software is evaluated by the customer. It also includes risk identification and monitoring like cost overrun or schedule slippage and after that planning of the next phase is started.

- **Spiral Model Advantages**

1. The spiral model is perfect for projects that are **large and complex** in nature as continuous prototyping and evaluation help in mitigating any risk.

2. Because of its **risk handling ability**, the model is best suited for projects which are very critical like software related to the health domain, space exploration, etc.

3. This model supports the client feedback and **implementation of change requests** (CRs) which is not possible in conventional models like a waterfall.

4. Since customer gets to see a prototype in each phase, so there are higher chances of customer satisfaction.

- **Spiral Model Disadvantages**

1. Because of the prototype development and risk analysis in each phase, it is very **expensive and time taking**.

2. It is **not suitable for a simpler and smaller** project because of multiple phases.

3. It requires **more documentation** as compared to other models.

4. Project **deadlines can be missed** since the number of phases is unknown in the beginning and frequent prototyping and risk analysis can make things worse.

17 Write agile manifesto principles

- **There are seven principles in software testing:**

1. **Testing shows the presence of defects**
2. **Exhaustive testing is not possible**
3. **Early testing**
4. **Defect clustering**
5. **Pesticide paradox**
6. **Testing is context-dependent**
7. **Absence of errors fallacy**

- **Testing shows the presence of defects:** The goal of software testing is to make the software fail. Software testing reduces the presence of defects. Software testing talks about the presence of defects and doesn't talk about the absence of defects. Software testing can ensure that defects are present but it can not prove that software is defect-free. Even multiple testing can never ensure that software is 100% bug-free. Testing can reduce the number of defects but not remove all defects.

- **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. Exhaustive testing is impossible means the software can never test at every test case. It can test only some test cases and assume that the software is correct and it will produce the correct output in every test case. If the software will test every test case then it will take more cost, effort, etc., which is impractical.

- **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. For better performance of software, software testing will start at the initial phase i.e. testing will perform at the requirement analysis phase.

- **Defect clustering:** In a project, a small number of modules can contain most of the defects. Pareto Principle to software testing state that 80% of software defect comes from 20% of modules.

- **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.

- **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.

- **Absence of errors fallacy:** If a built software is 99% bug-free but it 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 fulfill all the customer requirements.

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

- The meaning of Agile is swift or versatile."**Agile process model**" refers to a software development approach based on iterative development.
- Agile methods break tasks into smaller iterations, or parts do not directly involve long term planning.
- The project scope and requirements are laid down at the beginning of the development process.
- Plans regarding the number of iterations, the duration and the scope of each iteration are clearly defined in advance.
- Each iteration is considered as a short time "frame" in the agile process model, which typically lasts from one to four weeks.
- The division of the entire project into smaller parts helps to minimize the project risk and to reduce the overall project delivery time requirements.
- Each iteration involves a team working through a full software development life cycle including planning, requirements analysis, design, coding, and testing before a working product is demonstrated to the client.

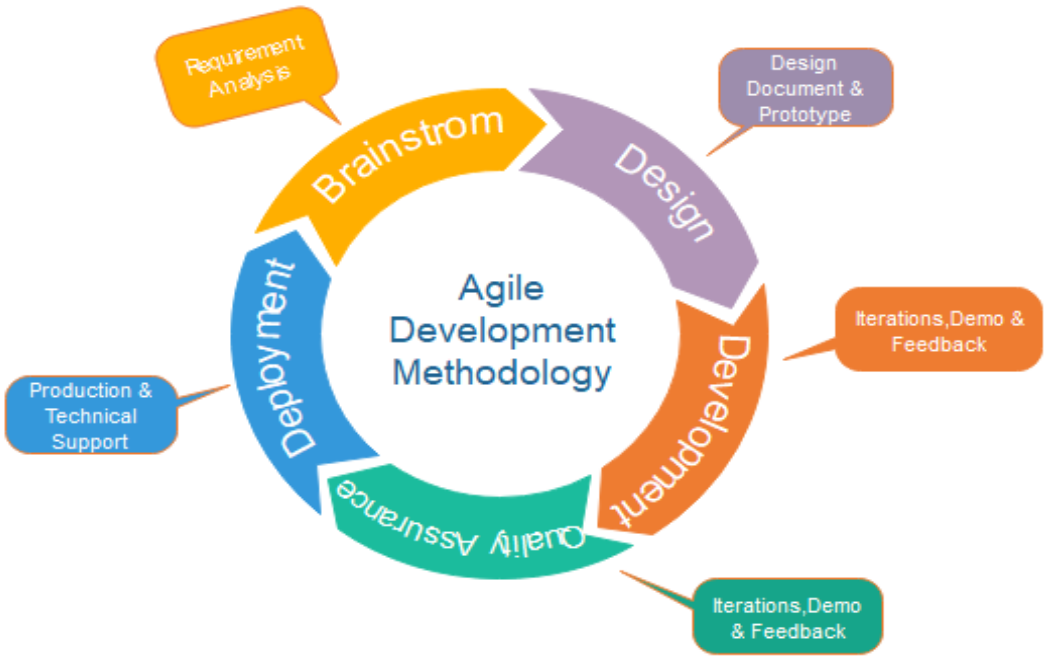


Fig. Agile Model

Phases of Agile Model:

- Following are the phases in the agile model are as follows:

1. Requirements gathering
2. Design the requirements
3. Construction/ iteration
4. Testing/ Quality assurance
5. Deployment
6. Feedback

1. **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.
2. **Design the requirements:**
   - When you have identified the project, work with stakeholders to define requirements. You can use the user flow diagram or the high-level UML diagram to show the work of new features and show how it will apply to your existing system.
3. **Construction/ iteration:**
   - When the team defines the requirements, the work begins. Designers and developers start working on their project, which aims to deploy a working product.
   - The product will undergo various stages of improvement, so it includes simple, minimal functionality.
4. **Testing:**
   - In this phase, the Quality Assurance team examines the product's performance and looks for the bug.
5. **Deployment:**
   - In this phase, the team issues a product for the user's work environment.

6. Feedback:

- After releasing the product, the last step is feedback. In this, the team receives feedback about the product and works through the feedback.

When to use the Agile Model?

- When frequent changes are required.
- When a highly qualified and experienced team is available.
- When a customer is ready to have a meeting with a software team all the time.
- When project size is small.

Advantage (Pros) of Agile Method:

1. Frequent Delivery
2. Face-to-Face Communication with clients.
3. Efficient design and fulfils the business requirement.
4. Anytime changes are acceptable.
5. It reduces total development time.

Disadvantages (Cons) of Agile Model:

1. Due to the shortage of formal documents, it creates confusion and crucial decisions taken throughout various phases can be misinterpreted at any time by different team members.
2. Due to the lack of proper documentation, once the project completes and the developers allotted to another project, maintenance of the finished project can become a difficulty.

19. Draw user case on online shopping product using payment gateway.

20. Draw user case on online shopping product using COD.