* Q.1 : What is SDLC

The Software Development Life Cycle (SDLC) refers to a methodology with clearly defined processes for creating high-quality software. in detail, the SDLC methodology focuses on the following phases of software development:

* Planning
* Analysis
* Design
* Build
* Testing
* Deployment
* Maintenance

This article will explain how SDLC works, dive deeper in each of the phases, and provide you with examples to get a better understanding of each phase.

* What is the software development life cycle?

SDLC or the Software Development Life Cycle is a process that produces software with the highest quality and lowest cost in the shortest time possible. SDLC provides a well-structured flow of phases that help an organization to quickly produce high-quality software which is well-tested and ready for production use

* What is agile methodology?.

Agile is an iterative approach to project management and software development that helps teams deliver value to their customers faster and with fewer headaches. Instead of betting everything on a "big bang" launch, an agile team delivers work in small, but consumable, increments. Requirements, plans, and results are evaluated continuously so teams have a natural mechanism for responding to change quickly.

* What is SRS

A software requirements specification (SRS) is a document that captures complete description about how the system is expected to perform. It is usually signed off at the end of requirements engineering phase

* What is oop

Object-oriented programming (OOP) is a computer programming model that organizes software design around data, or objects, rather than functions and logic. An object can be defined as a data field that has unique attributes and behavior.

OOP focuses on the objects that developers want to manipulate rather than the logic required manipulating them. This approach to programming is well-suited for programs that are large, complex and actively updated or maintained. This includes programs for manufacturing and design, as well as mobile applications; for example, OOP can be used for manufacturing system simulation software.

* Write Basic Concepts of oops
* 0n [Class in OOP -](https://www.topperskills.com/tutorials/oop/object-oriented-programming-classes-concepts.html)

The class is a model or blueprint or prototype of an object that defines or specifies all the properties of the objects.

* [An Object in OOP -](https://www.topperskills.com/tutorials/oop/object-oriented-programming-objects-concepts.html)

An object is an entity or instance of a class. The objects are mostly the physical entity but it can be a logical entity as well. Each object has state and behaviors.

* [Abstraction in OOP -](https://www.topperskills.com/tutorials/oop/object-oriented-programming-abstraction-concepts.html)

Abstraction means hiding the implementation and showing only functionality to the user

* [Encapsulation in OOP -](https://www.topperskills.com/tutorials/oop/object-oriented-programming-encapsulation-concepts.html)

Encapsulation is the process of binding the data by making it private and functions to process that data into a single unit. Encapsulation provides security to the data. Inheritance is the process of acquiring or sharing the properties and behaviors of the base class into the sub class to achieve reusability

1. [Polymorphism in OOP -](https://www.topperskills.com/tutorials/oop/object-oriented-programming-polymorphism-concepts.html)

Polymorphism mean same name having different functionality.

* Data Hiding in OOP -

Data hiding is the process of making the data members private so those will not be visible to other classes. The data of a object is not accessible to another objects, because each object has independent copy of the instance variables.

* What is object?

[object-oriented programming (OOP)](https://www.techtarget.com/searchapparchitecture/definition/object-oriented-programming-OOP), objects are the things you think about first in designing a program and they are also the units of code that are eventually derived from the process. In between, each object is made into a generic [class](https://www.techtarget.com/whatis/definition/class) of object, and even more generic classes are defined so that objects can share models and reuse the class definitions in their code. Each object is an instance of a particular class or subclass with the class's own methods or procedures and data [variables](https://www.techtarget.com/whatis/definition/variable). An object is what actually runs in the computer Objects can do things and can have things done to them. For example, a function or method object can be programmed to modify the contents of a data structure or variable object

* What is class?

In the real world, you often have many objects of the same kind. For example, your bicycle is just one of many bicycles in the world. Using object-oriented terminology, we say that your bicycle object is an [*instance*](javascript:var%20meth=openWin;%20meth('instance');) of the class of objects known as bicycles. Bicycles have some state (current gear, current cadence, two wheels) and behavior (change gears, brake) in common. However, each bicycle's state is independent of and can be different from that of other bicycles.

In object-oriented software, it's also possible to have many objects of the same kind that share characteristics: rectangles, employee records, video clips, and so on. Like the bicycle manufacturers, you can take advantage of the fact that objects of the same kind are similar and you can create a blueprint for those objects. A software blueprint for objects is called a [*class*](javascript:var%20meth=openWin;%20meth('class');).

* What is encapsulation?

In object-oriented computer programming (OOP) languages, the notion of encapsulation (or OOP Encapsulation) refers to the bundling of data, along with the methods that operate on that data, into a single unit. Many programming languages use **encapsulation**frequently in the form object that has both variables (data) and behaviors (functions or methods). A class is an example

encapsulation in computer science in that it consists of data and methods that have been bundled into a single unit.

* What is inheritance?

In [object-oriented programming](https://www.computerhope.com/jargon/o/oop.htm), inheritance refers to the ability of an [object](https://www.computerhope.com/jargon/o/object.htm) to take on one or more characteristics from other [classes](https://www.computerhope.com/jargon/c/class.htm) of objects. The characteristics inherited are usually instance [variables](https://www.computerhope.com/jargon/v/variable.htm) or member [functions](https://www.computerhope.com/jargon/f/function.htm). An object that inherits these characteristics is known as a subclass. The object it inherits them from is known as a [super class](https://www.computerhope.com/jargon/s/superclass.htm). The details of how inheritance is implemented vary between [languages](https://www.computerhope.com/jargon/p/programming-language.htm), but the first language to implement it was Similar in [1967](https://www.computerhope.com/history/1967.htm).

* What is polymorphism?

Polymorphism is an object-oriented programming concept that refers to the ability of a variable, function, or object to take on multiple forms. In a programming language exhibiting polymorphism, class objects belonging to the same hierarchical tree (inherited from a common parent class) may have functions with the same name, but with different behaviors.

* What is RDBMS?

RDBMS stands for Relational Database Management Systems. It is basically a program that allows us to create, delete, and update a relational database. Relational Database is a database system that stores and retrieves data in a tabular format organized in the form of rows and columns. It is a smaller subset of DBMS which was designed by E.F Cod in the 1970s. The major DBMS like SQL, My-SQL, ORACLE are all based on the principles of relational DBMS.

* What is SQL?

QL pronounced SQL stands for Structured Query Language. SQL is used to communicate with a database. According to ANSI (American National Standards Institute), it is the standard language for relational database management systems. SQL statements are used to perform tasks such as update data on a database, or retrieve data from a database. Some common relational database management systems that use SQL are: Oracle, Sybase, Microsoft SQL Server, Access, Ingres, etc.

* Write SQL Commands

1)Data Manipulation Language

DML commands are used to modify the database. It is responsible for all form of changes in the database.

2) Data Definition Language (DDL)

DDL changes the structure of the table like creating a table, deleting a table, altering a table, etc.

3) Data Control Language

DCL commands are used to grant and take back authority from any database user.

4) Transaction Control Language

TCL commands can only use with DML commands like INSERT, DELETE and UPDATE only.

5) Data Query Language

DQL is used to fetch the data f

* Write SDLC phases with basic introduction

Software Development Life Cycle (SDLC) is a framework that defines the steps involved in the development of software at each phase. It covers the detailed plan for building, deploying and maintaining the software.

SDLC defines the complete cycle of development i.e. all the tasks involved in planning, creating, testing, and deploying a Software Product.

SDLC is a process that defines the various stages involved in the development of software for delivering a high-quality product. SDLC stages cover the complete life cycle of a software i.e. from inception to retirement of the product.

* Explain Phases of the waterfall model

The waterfall model is a linear process model that divides development processes into successive project phases. In contrast to iterative models, each phase is run through only once. The results of each preceding phase are used as assumptions in the subsequent phase. The waterfall model is used especially in software development.

1. 1st analysis: planning, requirements analysis, and specification
2. Design: system design and specification
3. Implementation: programming and module tests
4. System integration: system and integration tests
5. Operation: delivery, maintenance, improvement

* Write phases of spiral model

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 analyzed and any potential risk is identified, analyzed, 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.

* Write agile manifesto principles

1)Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

2)Welcome changing requirements, even late in development. Agile processes harness change for the customer’s competitive advantage.

3)Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

4)Business people and developers must work together daily throughout the project.

5)Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

6) The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Working software is the primary measure of progress.

7)Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

8)Continuous attention to technical excellence and good design enhances agility.

Simplicity–the art of maximizing the amount of work not done–is essential.

9)The best architectures, requirements, and designs emerge from self-organizing teams.

10)At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior according

11)The best architectures’, requirements, and designs emerge from self-organizing teams.

12)At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

* What is join?

A join is an SQL operation performed to establish a connection between two or more database tables based on matching columns, thereby creating a relationship between the tables. Most complex queries in an SQL database management system involve join commands.  
There are different types of joins. The type of join a programmer uses determines which records the query selects. Three algorithms work behind join operations: hash join, sort-merge join and nested loop join.

* Write type of joins.

SQL Server supports many kinds of different joins including INNER JOIN, SELF JOIN, CROSS JOIN, and OUTER JOIN. In fact, each join type defines the way two tables are related in a query. OUTER JOINS can further be divided into LEFT OUTER JOINS, RIGHT OUTER JOINS, and FULL OUTER JOINS.

1)SQL INNER JOIN creates a result table by combining rows that have matching values in two or more tables.

2)SQL LEFT OUTER JOIN includes in a result table unmatched rows from the table that is specified before the LEFT OUTER JOIN clause.

3)SQL RIGHT OUTER JOIN creates a result table and includes into it all the records from the right table and only matching rows from the left table.

4)SQL SELF JOIN joins the table to itself and allows comparing rows within the same table.

5)SQL CROSS JOIN creates a result table containing paired combination of each row of the first table with each row of the second table.

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

Agile methodology is a project management strategy that divides the project into multiple phases, encouraging continuous improvement for each phase. In the beginning of the project, the team cycles through planning, evaluation and execution stages to collaborate toward multiple project goals. As a methodology, the agile project strategy contains four fundamental values, including:

* Cons of agile methodology

There are some disadvantages from using the agile methodology style of project management, including:

1. Transfer difficulties

When switching from one management style to another for an entire department, it may take time to fully acclimate to the new responsibilities and style of how projects progress within the system. Leaders may have some initial difficulties adjusting to the management style, although consistent practice and training can help all employees adjust to new tactics. While agile management encourages employees to act independently toward goals, consider dividing departments into groups for the first few objectives of the project. This is so that each member can consult another concerning questions or transfer difficulties.

2. Variable goals

Because the agile methodology focuses on multiple goals at a time, certain goals may not receive as much focus throughout the entire project. Variable goals within a department can lead to a lack of specific goals for a team, which may lead to unknown deadlines and less accurate costs. One way to prevent variable goals from allowing unwarranted costs or missed deadlines can be to schedule regular meetings to discuss goals. Another way to unify goal costs is to set a policy regarding spending in the department so that cash flow follows a policy to reduce unforeseen spending.

3. Lack of documentation

In the agile methodology system, documentation is less important than reactionary planning and progress. This can make documentation responsibilities such as record keeping, billing statements and plan-keeping complete more slowly in comparison to other duties. To help improve documentation while using this methodology, consider taking record of documented statements at least once a month. If documentation has fallen behind, emphasizing it on the schedule can help bring it back to proper importance again.

4. Less documented improvement

Because agile methodology strategies focus on reactionary improvement rather than documented improvement phases, documented success and calculated methodology strategies may be ineffective throughout the project. While reactionary tactics can be effective for improving a process or product, consider improving your team's documented improvement efforts by scheduling document checks regularly. Keeping a gradual record of success and progress can help the team gain hindsight concerning objectives and next steps.

5. Goal focus shifting

Since agile methodology involves focus shifting based on which part of a project requires the most attention, it may be difficult to lead all team members toward a singular goal. In later stages of the project, it may become preferable to have the majority of department members focusing on a singular goal. To help achieve goal unity, consider allocating goal-related tasks to multiple members of the department during the final phases of the project. Unity through allocation can help a department work as a unified force, even if only temporarily.

6. Less predictability

Because agile production relies on continuous improvement and customer feedback, the department may not be able to predict profits before production begins. Agile production focuses on sending acceptable products to customers as soon as possible, which can make potential issues harder to predict over time. However, using feedback effectively and quickly during the process may be able to increase the product's fault predictability marginally over time, especially if the department documents issues during the process.

* Pros of agile methodology

There are many benefits to using an agile methodology strategy in your department, including:

1. Timely delivery

Because it focuses on product placement, the agile strategy allows a department to deliver products to clients as quickly as possible. While products and development processes may not be as optimal as possible due to lessened planning stages, the agile method allows improvement to progress through reactionary feedback. For departments that can benefit from producing a stable product first before optimizing the process, agile methodology is an acceptable approach.

2. Adaptability

Since the improvement increments between product delivery dates are small, projects can easily change and adapt when underneath an agile strategy system. This allows production to continue at an acceptable rate while improving the process continuously. Departments that work with an agile production system can adapt quickly to change requests, even between clients that have slightly different preferences.

3. Ease of collaboration

Because agile working requires much feedback between the client and employees, agile methodology introduces a great system for collaborating between both customers and other departments. In an agile system, management encourages employees to think creatively to solve problems and create solutions for project challenges as they arise. Using an agile system in your department can help your employees learn to collaborate with others to produce a better product with each piece of feedback they receive.

4. Increased performance improvement

Because agile department test product increments as they produce them, department members can quickly react to problems as they may arise. The fragmented nature of production and correction allows agile departments to better understand and correct problems quickly. By using feedback from clients and other members, departments can fix issues quickly before the next production cycle.

5. Transparency

With the agile approach to departmental work, both potential issues and process improvements become evident with each production cycle. This type of management allows employees to both correct mistakes and improve production details quickly. Because management encourages employees to act independently in this format, the time between understanding an improvement and implementing it may be shorter.

6. Continuous improvement

Because the agile department strategy relies on improving a process as it produces products, improvement can directly affect the next product. Additionally, because agile strategies don't stop production to implement fixes, management encourages employees to act on feedback as soon as possible. A continuously improving department can become even more efficient as they sell products.

7. Higher profits

The agile department focuses on producing a continuously improving product rather than a perfect one. This enables agile departments to gain profits as quickly as possible, since each product profit brings feedback back to the team. Because an agile department focuses on production, an agile strategy may help a team produce products without stopping production for major fixes or process changes.

8. Less preparatory work

Because agile development focuses on the product more than process improvement, an agile department can produce a product more quickly than other management methods. By accepting feedback and making changes over time rather than stopping production to make larger changes, products can improve while not fully compromising improvement efforts. Additionally, when starting a production cycle, the agile departmental values have less preparatory work required before they can begin selling products, saving time between the planning and profit stages

* Draw use case on Online shopping product using COD.



* Draw use case on Online shopping product using payment gateway.

