ASSIGNMENT

MODULE 1) SOFTWARE TESTING-INTRODUCTION AND FUNDAMENTALS

Submitted by:

ANUMOL.G

Module 1 (Fundamental)

Basic

B1. What is software testing?

Software testing is a process of verifying and validating whether a software application or product meets the business and technical requirements that guide as design and development.

B2. What is SDLC?

Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality Software.

B3. What is Software Development Methodology?

The software development methodology is a framework that is used to structure, plan and control the process development of an information system.

B4. What is agile methodology?

Agile methodology is a type of project management process, mainly used for software development, where demands and solutions evolve through the collaborative effort of self-organizing and cross-functional teams and their customers.

B5. What is use-case?

Use case is the tool to represent software systems or application system interact

with its environment. It shows the sequence of actions systems performs that adds value to the actor.

B6. What is Activity Diagrams?

The activity diagram is a flow chart to represent the flow of control among the activities in a system.

B7. What is SRS?

A document use to describe the behavior of the software system, Functional, Non - Functional requirements of the software system.

B8. What is Programming?

A program has a set of instructions written in correct order to get the desired result. The method of writing the instructions to solve the given problem is called programming.

B9. What is OOP?

OOP stands for Object Oriented Programming language, the main purpose of OOP is to deal with real world entity using programming language.

B10. Write Basic Concepts of oops?

The basic concept of oops

- Object
- Class
- Encapsulation
- Inheritance
- Polymorphism
    - Overriding
    - Overloading
- Abstraction

B11. What is object?

Object is an instance of class is created, it takes up space like other variable in memory.

B12. What is class?

Class is a collection of objects and it doesn't take any space on memory, class is also called as blueprint/logical entity.

B13. What is RDBMS?

Relational Database Management system (RDBMS)is a software system which is used to store only data which need to be stored in the form of tables. In this kind of system data is managed and stored in rows and columns which is known as tuples and attributes, RDMS is a powerful data management system & is widely used across the world.

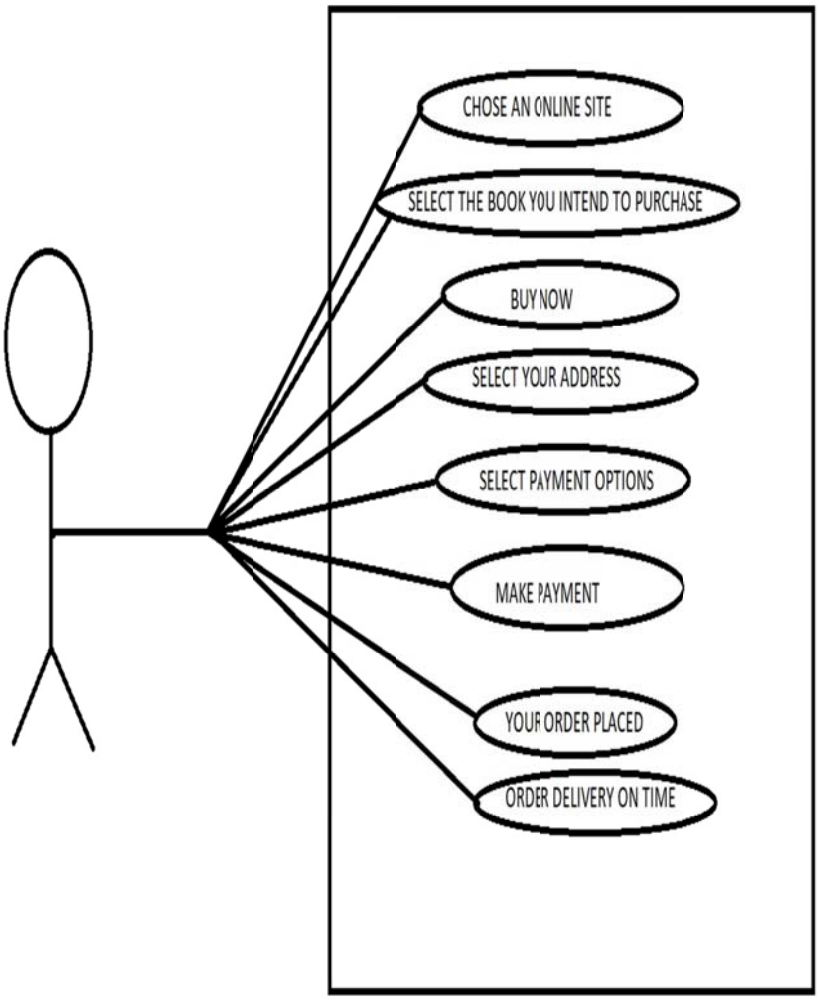
B14. What is SQL?

SQL stands for "structured Query Language. Every relational database software interact with a language known as SQL. because it is a simple English like language which guidelines are provided by a standard organization 'ANSI' adopted by all database from venders like Oracle, MySQL, Microsoft etc.

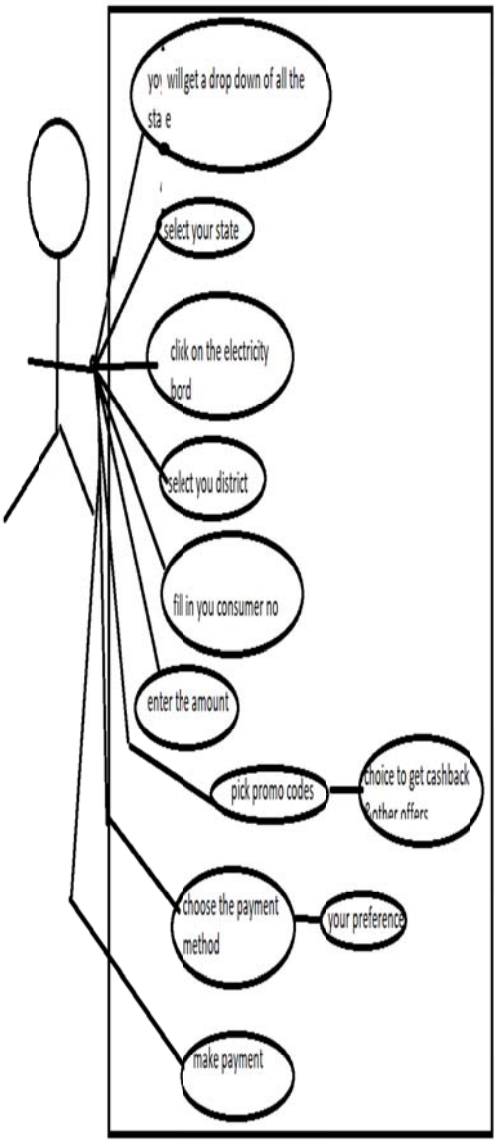
B15. Write SQL commands?

- DDL - Data Definition Language
- DML - Data Manipulation Language
- DCL - Data Control Language
- DQL - Data Query Language

B16. Draw Use case on online book shopping?



B17. Draw Use case on online bill payment system (pay tm) ?



Intermediate

I1. Write SDLC phases with basic introduction?

The following are the various steps of the software life cycle

- Planning and Requirement analysis
- Defining
- Designing the software
- Coding or Developing the product
- Testing the product
- Deployment in the market
- Maintenance

❖ Planning and Requirement analysis

The requirement is the first stage in the SDLC process.

It is conducted by the senior team members by collecting the inputs from the stakeholders or customer and domain experts in the industry.

Planning for the quality assurance requirements and recognization of the risks involved is also done at this stage.

❖ Defining (Feasibility study)

Once the requirement analysis phase is completed the next step is to define and document software needs.

This process is conducted with the help of Requirement Specification document also known as 'SRS' document.

It includes everything which should be designed and developed during the project life cycle.

❖ Designing

In this third phase, the system and software design documents are prepared as per the requirement specification document .

This helps to define the overall system architecture.

❖ Coding and Developing the product

In this phase of SDLC,the actual development begins,and the product built.

The implementation of design begins with writing code.

❖ Testing the product

After the code is generated, it is tested against the requirements to make sure that the products are solving the needs addressed and gathered during the requirements stage.

During this stage, unit testing, integration testing, system testing, acceptance testing are done.

- Deployment

Once the software is certified, and no bugs or errors are stated, then it is deployed.

After the software is deployed then its maintenance begins.

- Maintenance

Once when the client starts using the developed systems, then the real issues come up and requirement s to be solved from time to time .

This procedure where the care is taken for the developed product is known as maintenance.

I2. Explain the types of requirements?

- Functional Requirements: describe system services or functions.
    - Compute sales tax on a purchase
    - Update the database on the server
- Non-Functional Requirements: are constraints on the system or the development process.
- Non-Functional Requirements may be more critical than functional requirements.
- If these are not met, the system is useless.

I3. State the importance of Design phase?

In this phase, the requirement gathered in the SRS Document is used as an input and software architecture that is used for implementing system development is derived. Analyzing the trade-offs of necessary complexity allows for many things to remain simple which, in turn, will eventually lead to a higher quality product. The architecture team also converts the typical scenarios into a test plan.

I4. What are the tasks performed in coding phase?

In this phase, developers start build the entire system by writing code using the chosen programming language. In the coding phase, tasks are divided into units or

modules and assigned to the various developers. It is the longest phase of the Software Development Life Cycle process.

I5. Briefly explain Testing phase?

Testing is one of the most critical processes of the Software Development Life cycle (SDLC). The testing phases of the software development lifecycle help companies to identify all the bugs and errors in the software before the implementation phase begins.

- I6. Explain phases of the Waterfall model?

  - Requirement Gathering and analysis
  - System Design
  - Implementation
  - Integration and Testing
  - Deployment of system
  - Maintenance

  - Requirement Gathering and analysis

    All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.

  - System Design

    The requirement specifications from first phase are studied in this phase and the system is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.

  - Implementation

    With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.

  - Integration and Testing

    All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire test is tested for any faults and failures.

  - Deployment of system

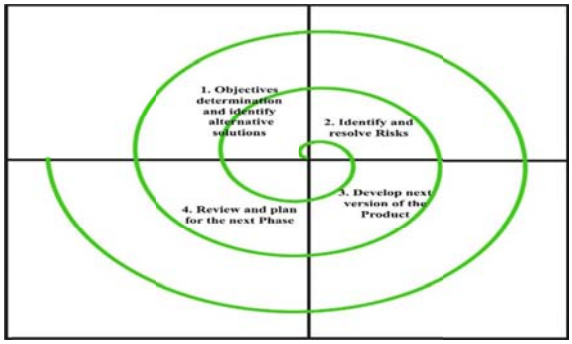
    Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.

- Maintenance

  There are some issues which comes up in the client environment. To fix those issues patches are released. Also to enhance the product some better versions are release. Maintenance is done to deliver these changes in the customer environment.

I7. Write phases of spiral model?

The spiral model has four phases. A software project repeatedly passes through these phases in iterations called Spirals.



- Identification

  This phase starts with gathering the business requirements in the baseline spiral. In the subsequent spirals as the product matures, identification of system requirements, subsystem requirements and unit requirements are all done in this phase.

  This phase also includes understanding the system requirements by continuous communication between the customer and the system analyst. At the end of the spiral, the product is deployed in the identified market.

- Design

  The Design phase starts with the conceptual design in the baseline spiral and involves architectural design, logical design of modules, physical product design and the final design in the subsequent spirals.

- Construct or Build

  The Construct phase refers to production of the actual software product at every spiral. In the baseline spiral, when the product is just thought of and the design is

being developed a POC (Proof of Concept) is developed in this phase to get customer feedback.

Then in the subsequent spirals with higher clarity on requirements and design details a working model of the software called build is produced with a version number. These builds are sent to the customer for feedback.

- Evaluation and Risk Analysis

  Risk Analysis includes identifying, estimating and monitoring the technical feasibility and management risks, such as schedule slippage and cost overrun. After testing the build, at the end of first iteration, the customer evaluates the software and provides feedback.

I8. Write Agile manifesto principles?

- Individuals and interactions

  In agile development, self-organization and motivation are important, as are interactions like co-location and pair programming.

- Working software

  Demo working software is considered the best means of communication with the customer to understand their requirement, instead of just depending on documentation.

- Customer collaboration

  As the requirements cannot be gathered completely in the beginning ,of the project due to various factors, continuous customer interaction is very important to get proper product requirements.

- Responding to change

  Agile development is focused on quick responses to change and continuous development.

I9. What is Actor in Use-case?

An Actor in use-case modeling specifies a role played by a user or any other system that interacts with the subject. An A Actor models a type of role played by an entity that interacts with the subject (e.g., by exchanging signals and data), but which is external to the subject.

Actors may represent roles played by human users, external hardware or other subjects. Actors do not necessarily represent specific physical entities but merely particular roles of some entities that are relevant to the specification of its associated use cases.

Types of actors includes:

- Users

- Database systems
- Clients and servers
- Cloud platforms devices

I10. How many kinds of nodes in Activity Diagrams? which?

There are **two types** of final node: activity and flow final nodes. The activity final node is depicted as a circle with a dot inside.In activity diagrams, a control node is **an abstract activity node that coordinates the flow of control in an activity**. The following table describes the types of control nodes that you can use in activity diagrams. This node represents a point where all flows in an activity stop.

Activity diagram indicates flow among activities.it should model user level steps.
Two kinds of nodes:

- Action state
- Sequential state

I11. What is Encapsulation?

In object -oriented computer programming languages, the notion of encapsulation refers to building of data, along with the methods that operate on that data, into a single unit. Many programming languages use encapsulation frequently in the form of classes. A class is a program-code-templates that allows developers to create an object that has both variables and behaviors. A class is an example of encapsulation in computer science in that it consists of data and methods that have been bundled into a single unit.

Encapsulation may also refer to a mechanism of restricting the direct access to some components of an object ,such that users cannot access state values for all of the variables of a particular object.

Encapsulation can be used to hide both data members and data functions or methods associated with an instantiated class or object.

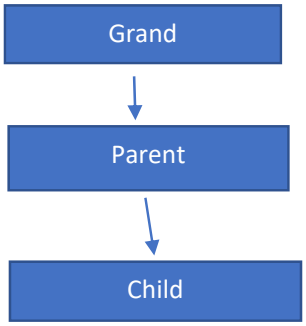
I12. What is inheritance?

When we construct a new class from existing class in such a way that the new class access all the features and properties of existing class called inheritance

It provides code reusability. We cannot access private members of class through inheritance. A subclass contains all the features of super class so we should create the object of sub class. Method overriding only possible through inheritance.

In general, Java supports single-parent, Multiple-children inheritance and multilevel inheritance. Java supports multiple inheritances only through interfaces.

In a class context, inheritance is referred to as implementation inheritance, and in an interface context, it is also referred to as interface inheritance.



I13. What is polymorphism?

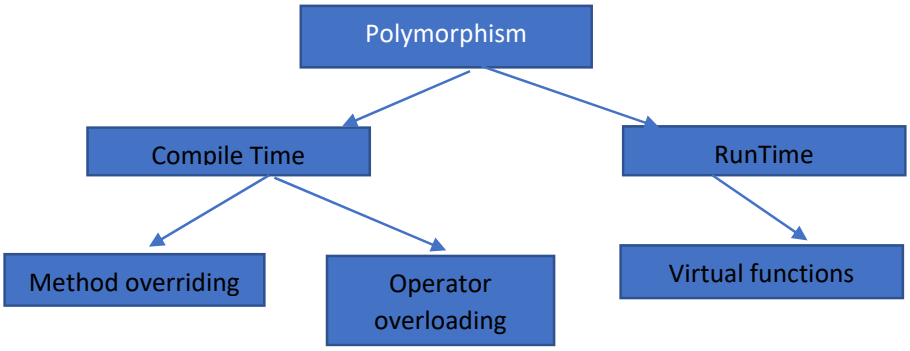
Polymorphism is the ability of any data to be processed in more than one form. The word itself indicates the meaning as poly means many and morphism means types.it is one of the most important concept of object oriented programming language. The most common use of polymorphism in object-oriented programming occurs when a parent class reference is used to refer to a child class object.

Real life example of polymorphism, a person at the same time can have different roles to play in life. Like a women at the same time is a mother, a wife, an employee and a daughter.so the same person has to have many features but has to implement each as per the situation and the condition.

Polymorphism is considered as one of the important features of Object Oriented Programming. It implements the concept of function overloading, function overriding and virtual functions.

There are two types of polymorphism in Java

- Compile time polymorphism (Overloading)
- Runtime polymorphism (Overriding)



I14. What is Abstraction?

Abstraction is the representation of the essential features of an object. These are 'encapsulated' into an abstract data type. Data abstraction refers to, providing only essential information to the outside word and hiding their background details. That is to represent the needed information in program without presenting the details.

Abstraction in Object Oriented programming refers to the ability to make a class abstract. Abstraction tries to reduce and factor out details so that the programmer can focus on a few concepts at a time. Java provides interfaces and abstract classes for describing abstract types

An interface is a contract or specification without any implementation. An interface cannot have behavior or state.

An abstract class that cannot be instantiated. All other functionality of the class implementation

A detailed comparison of interfaces and abstract classes can be found at interface vs abstract-class.

I15.Why SQL?

SQL is widely popular because it offers the following advantages:

- Allows users to access data in the relational database management systems.
- Allows users to describe the data.
- Allows users to define the data in database and manipulate that data
- Allows to embed within other languages using SQL modules, libraries and pre-compilers.
- Allows users to create and drop databases and tables
- Allows users to create view, stored procedure, functions in a database.
- Allows users to set permissions on tables, procedures and views.

I16. Write SQL commands in detail?

The standard SQL commands to interact with relational databases are CREATE, SELECT, INSERT, UPDATE, DELETE AND DROP. These commands can be classified into the following groups based on their nature.

- DDL-Data Definition Language

Sr. No	Commands and Description
1	CREATE Creates a new table, a view of a table, or other object in the database
2	ALTER Modifies an existing database object, such as table.

3	DROP Deletes an entire table, a view of a table or other objects in the database.

- DML-Data Query Language

Sr. No	Commands and description
1	SELECT Retrieves certain records from one or more tables.

- DML-Data Manipulation language

Sr. No	Commands and Description
1	INSERT Insert records
2	UPDATE Update records
3	DELETE Delete records

- DCL-Data Control Language

Sr. No	Commands and Description
1	GRANT Gives a privilege to user.
2	REVOKE Takes back privileges granted from user.

I17. What is join?

SQL join statements allow us to access information from two or more tables at once they also keep our database normalized. Normalization allows to keep data redundancy low so that we can decrease the amount of data anomalies in our application when we delete or update a record.

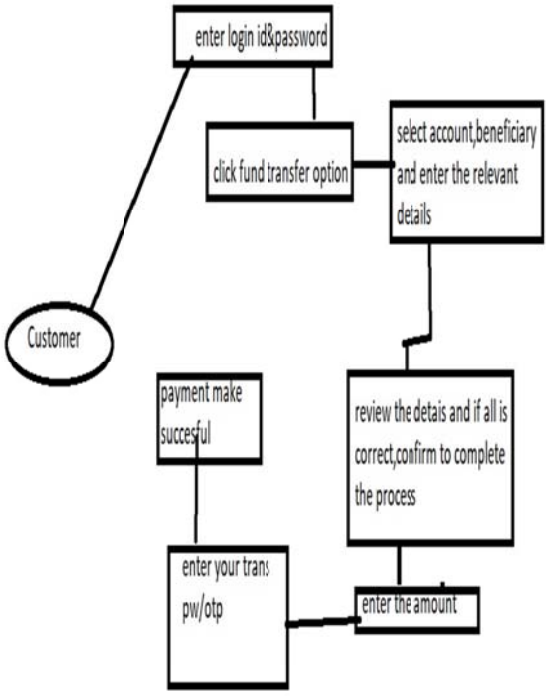
- A Join clause allows us to combine rows from two or more tables based on a related column.

I18. Write type of Joins?

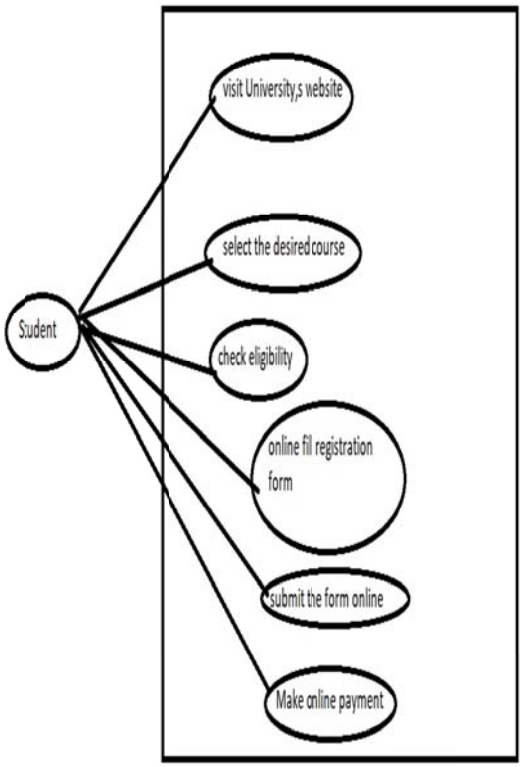
The type of join statement we can use depends on our use case. There are four different types of join operations:

- INNER JOIN: Returns dataset that have matching values in both tables.
- LEFT JOIN: Returns all records from the left table and matched records from the right table.
- RIGHT JOIN: Returns all records from the right table and matched records from the left table
- FULL JOIN: Returns all records when there is a match in either the left and right table.

I19. Draw Use case on Online payment transfer via bank to bank?



I20.Draw usecase on Student Registration Process on University?



Advanced

A1.Explain phases of SDLC in details?

The software life cycle is a process that consists of a series of planned activities to develop or after the Software Products.

- The SDLC aims to produce high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates
- Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality Software.

The following are the various steps of the software life cycle

- Planning and Requirement analysis
- Defining
- Designing the software
- Coding or Developing the product
- Testing the product
- Deployment in the market
- Maintenance

❖ Planning and Requirement analysis

The requirement is the first stage in the SDLC process.

It is conducted by the senior team members by collecting the inputs from the stakeholders or customer and domain experts in the industry.

Planning for the quality assurance requirements and recognization of the risks involved is also done at this stage.

❖ Defining (Feasibility study)

Once the requirement analysis phase is completed the next step is to define and document software needs.

This process is conducted with the help of Requirement Specification document also known as 'SRS' document.

It includes everything which should be designed and developed during the project life cycle.

❖ Designing

In this third phase, the system and software design documents are prepared as per the requirement specification document .

This helps to define the overall system architecture.

❖ Coding and Developing the product

In this phase of SDLC, the actual development begins, and the product built.

The implementation of design begins with writing code.

❖ Testing the product

After the code is generated, it is tested against the requirements to make sure that the products are solving the needs addressed and gathered during the requirements stage.

During this stage, unit testing, integration testing, system testing, acceptance testing are done.

- Deployment

Once the software is certified, and no bugs or errors are stated, then it is deployed.

After the software is deployed then its maintenance begins.

- Maintenance

Once when the client starts using the developed systems, then the real issues come up and requirement s to be solved from time to time.

This procedure where the care is taken for the developed product is known as maintenance.

A2. According to you which is most creative and challenging phase of system life cycle?

The most Creative and challenging phase of the system development life cycle is system design. The term design describes a final system and the process by which it is developed.

It refers to the technical specifications (analogous to engineer's blueprints) that will be applied in implementing the candidate system. It also includes the construction of programs and program testing.

- The aim of the design phase is to determine how the product will be made.
- Architectural Design
    - Decomposes the product into modules.
    - Describes the functionality of each module.
    - Describes the interface of each module
        - Methods names

- Inputs
- Outputs

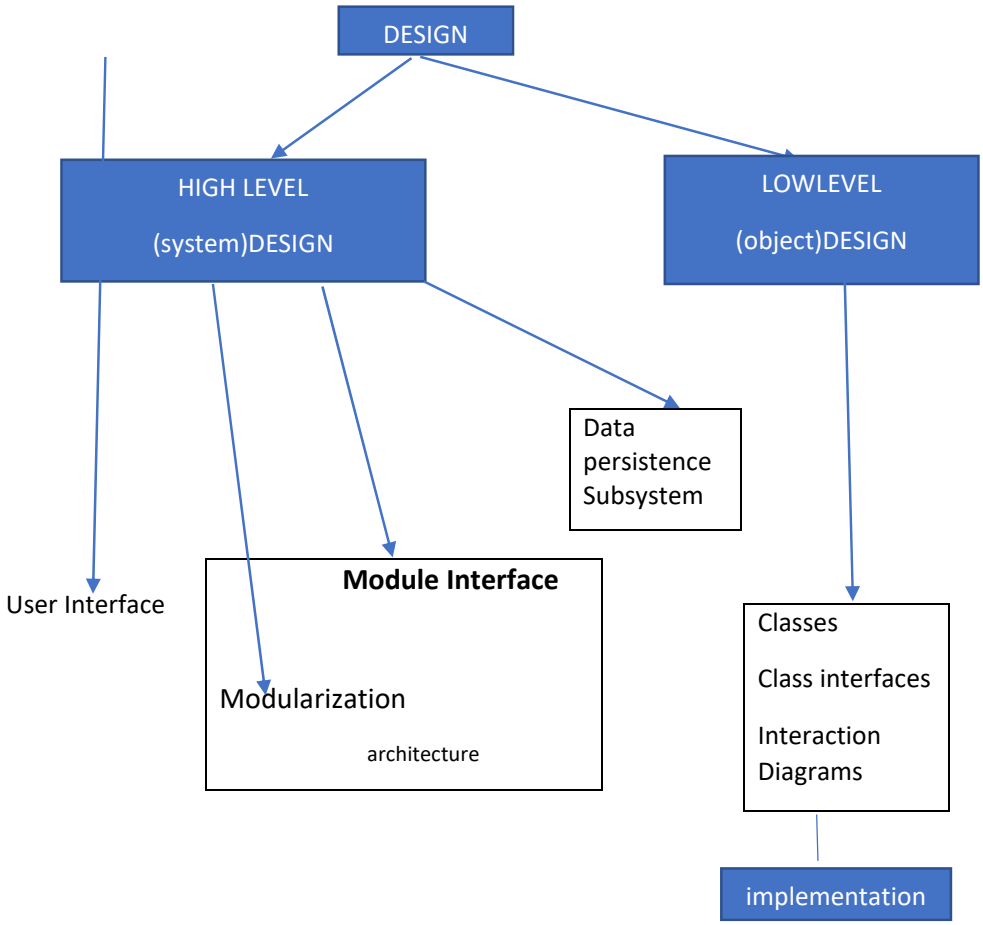
- Detailed Design

Determines the internal structure of each module.

Algorithms are selected and data structure are chosen

The internal data flows are determined.

Map of design phase



- **Activities of Design phase of SDLC**
  - Design an integrate the network
  - Design the application architecture
  - Design the user interfaces
  - Design the system interfaces
  - Design and integrate the database
  - Prototypes for design details
  - Design and integrate a system controls

The design phase is to transform the requirement into complete and detailed system design specifications

A3. Who are the people involved in the phases of Waterfall model?

Each software development has a particular team structure that should be implemented to hold its principles. Waterfall is no exception. This popular software development methodology has lots of specific aspects in its team building.in this article, we will tell you about the roles in waterfall methodology. But before doing that we should give the general definition of Waterfall and its team.

Waterfall is considered the traditional model of software development. That is because it was originated in the 1950s.At that time software development was a completely new invention, so the first programmers just took a typical hardware development methodology and applied it to the newborn industry. The methodology is based on three simple principles: strong documentation, low degree of customer involvement, and sequential structure of project realization.

Waterfall teams are usually quite large. They include more than 15 people. Partially that is because of the strict hierarchical structure of such teams. Their members are not interchangeable, unlike the members of Agile teams. Each of them is responsible for a certain piece of work. Every Waterfall developer must report on his activities to the project manager. This person is the formal leader of each Waterfall team.

Waterfall teams have four typical roles.

A developer is a person who creates the code. This is one of the most important role in Waterfall teams. Waterfall programmers must avoid bugs during their work because one single defect may be a reason to run the entire project from the very beginning.

The role of a tester is also extremely important. In Waterfall projects, tests are usually conducted at the final stages of their realization. That is why testes have to find all bugs in final products and return the software to the developers so that they can fix all defects.

A Business analyst is a person responsible for making the software product popular in the digital market. His main task is to write business strategies.

A project manager is the main person in every Waterfall team. He is responsible for the quality of final software. His main task is to manage the projects and to subdivide tasks among other team members.

In software development, it tends to be among the less iterative and flexible approaches, as progress flows in largely one direction downwards, like a waterfall, through the phases of conception, initiation, analysis, design.

1.Requirements

If you are into software development or any type of project creation team, you would want to know the business context of what you are trying to create — you want to define what kind of problems you are trying to resolve and how people would react to your finished product. After you define all these “requirements”, you have the input that you need to move on to the next step.

2. Designing

This step is made up of all the steps that you need to satisfy all the requirements that you have determined earlier. In software development, this is the part where you define all the software and hardware architecture, programming language, data storage, etc. This is also the part wherein you determine how the project would be useful to its end user.

3. Implementation

In this step, you begin to construct what you have designed in your plan. This part of the Waterfall method is dedicated to meeting the standards that you have made in the previous steps. This is the part where people from the development team come in and make all the things discussed in the previous steps happen.

4. Verification

This is the part of the method where quality assurance people enter to ensure that the development team did not make any mistakes. This is also most likely the part where people realize what is working or not working in their plan.

5.Maintenance

It is the task performed by every user once the software ha been delivered to the customer, installed and operational. There are some issues which come up in the client environment. To fix these issues patches are released. Also enhance the product some better versions are released. Maintenance is done to deliver there changes in the customer environment.

A4. Explain the team Requirement Gathering concerning SDLC?

The software life cycle is a process that consists of a series of planned activities to develop or after the Software Products.

- The SDLC aims to produce high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates
- Software Development Life Cycle (SDLC) is a process used by the software.

The most important phase of SDLC is the requirement gathering and analysis phase because this is when the project team begins to understand what the customer wants from the project. During the requirements gathering sessions, the project team meets with the customer to outline each requirement in detail.

Step Requirements Gathering Process

1. Identify the relevant stakeholders.
2. Establish project goals and objectives.
3. Elicit requirements from stakeholders.
4. Document the requirements.
5. Confirm the requirements.
6. Prioritize the requirements

Three types of problems can arise:

- Lack of clarity: It is hard to write documents that are both precise and easy to read
- Requirement confusion: Functional and Non-Functional requirements tend to be interviewed.
- Requirements Amalgamation: Several different requirements may be expressed together.

Types of Requirements:

- Functional Requirements: describe system services or functions.
  - Compute sales tax on a purchase
  - Update the database on the server
- Non-Functional Requirements: are constraints on the system or the development process.

- Non-Functional Requirements may be more critical than functional requirements.
- If these are not met, the system is useless.

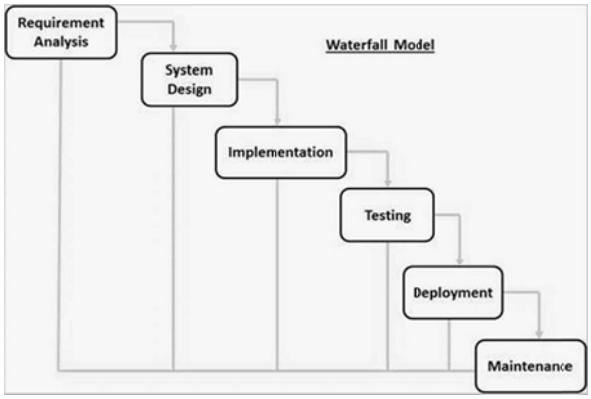
A5. What are problems faced in the waterfall model?

The Waterfall Model was the first Process Model to be introduced. It is also referred to as a **linear-sequential life cycle model**. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.

The Waterfall model is the earliest SDLC approach that was used for software development.

The waterfall Model illustrates the software development process in a linear sequential flow. This means that any phase in the development process begins only if the previous phase is complete. In this waterfall model, the phases do not overlap.

The following illustration is a representation of the different phases of the Waterfall Model.



Waterfall Model - Application

Every software developed is different and requires a suitable SDLC approach to be followed based on the internal and external factors. Some situations where the use of Waterfall model is most appropriate are

- Requirements are very well documented, clear and fixed.
- Product definition is stable.
- Technology is understood and is not dynamic.
- There are no ambiguous requirements.
- Ample resources with required expertise are available to support the product.
- The project is short.

Clients may not know exactly what their requirements are before they see working software and so change their requirements, leading to redesign, redevelopment, and retesting, and increased costs.

Designers may not be aware of future difficulties when designing a new software product or feature, in which case it is better to revise the design than persist in a

design that does not account for any newly discovered constraints, requirements, or problems.

Thus, there is no guarantee that the requirements that the organization has thought of would actually work. From here, you would realize that the Waterfall model has the following problems:

1.**People blindly follow plans.**

In the traditional method, people pay more attention to how things will happen during the right moment without being mindful if things are really falling into place. While planning is important, it is also important that developers and quality checkers understand how things should happen, especially with the client or the end-user. It is also important that all people involved in the project can immediately say how a particular step in project fulfillment can fall apart without having to wait for the testing stage.

2.**Sequential process and changes become costly**

- This approach does not make allowance for the change of defined requirements as the project progresses. Thus there is a big potential that the software will not fully meet the requirement of the user, it will be inefficient and have poor functionality.
- It is inadequate as developers cannot just go back and change something in a previous phase as the consumers requirement change but the developer has to go back to where the requirement needs to change and start that phase all over. Not until that phase is complete can he move on to the next phase.

3. **End-users do not know what they want.**

Most times a end users mind is constantly changing and most persons have a vague idea of their software requirements and its as the software develops that they specify their requirements.

When it is time to hand over the finished product to a client, it is likely that they will not like how it turned out, despite deliberately saying otherwise during the initial stages. It is easy for clients and end-users to change what they want over time. The Waterfall system does not have a way to resolve that yet, without having to revise plans and redoing the entire project altogether.

4. **Testing for quality may suffer.**

It is impossible to accurately predict the outcomes of a project, and when the entire team is pressed for time, it is possible to cut the testing stage short in order to meet the deadline.

5. **You'll never know what stage you really are on.**

Since the product that you are trying to create will not be produced until the very end, you are not really sure if you are still on planning or you are already on developing stage. That means that it is also likely for you to spend more time on a stage than what you have expected because of this poor visibility.

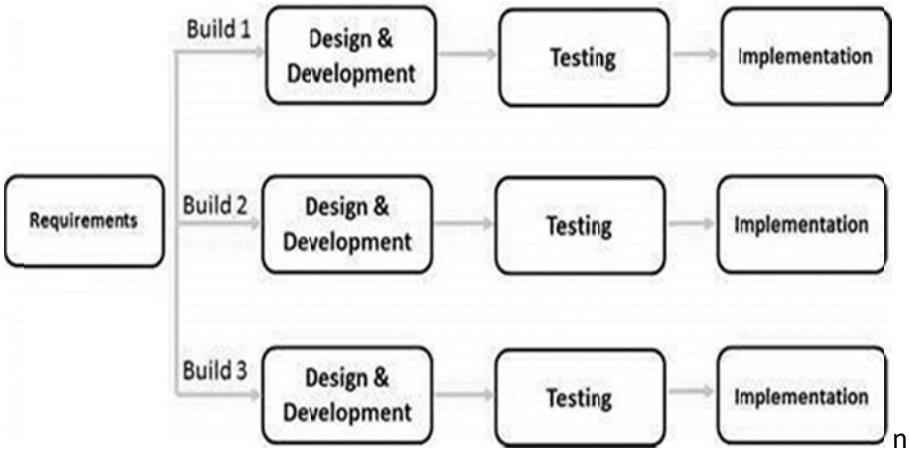
In the end, the Waterfall method can be too risky since it is too rigid. In order for you to make you produce a product that works and would be flexible enough to help you figure out what is working or not

A6. Write applications of iterative and incremental model?

In the Iterative model, iterative process starts with a simple implementation of a small set of the software requirements and iteratively enhances the evolving versions until the complete system is implemented and ready to be deployed.

An iterative life cycle model does not attempt to start with a full specification of requirements. Instead, development begins by specifying and implementing just part of the software, which is then reviewed to identify further requirements. This process is then repeated, producing a new version of the software at the end of each iteration of the model. Iterative process starts with a simple implementation of a subset of the software requirements and iteratively enhances the evolving versions until the full system is implemented. At each iteration, design modifications are made and new functional capabilities are added. The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental).

The following illustration is a representation of the Iterative and Incremental model :



nctional

Iterative and Incremental development is a combination of both iterative design or iterative method and incremental build model for development. "During software development, more than one iteration of the software development cycle may be in progress at the same time." This process may be described as an "evolutionary acquisition" or "incremental build" approach."

Iterative model-Application

- Like other SDLC models, Iterative and incremental development has some specific applications in the software industry. This model is most often used in the following scenarios – Requirements of the complete system are clearly defined and understood.
- Major requirements must be defined; however, some functionalities or requested enhancements may evolve with time.
- There is a time to the market constraint.
- A new technology is being used and is being learnt by the development team while working on the project.
- Resources with needed skill sets are not available and are planned to be used on contract basis for specific iterations.
- There are some high-risk features and goals which may change in the future.

Iterative Model - Pros and Cons

The advantage of this model is that there is a working model of the system at a very early stage of development, which makes it easier to find functional or design flaws. Finding issues at an early stage of development enables to take corrective measures in a limited budget.

The disadvantage with this SDLC model is that it is applicable only to large and bulky software development projects. This is because it is hard to break a small software system into further small serviceable increments/modules.

The advantages of the Iterative and Incremental SDLC Model are as follows −

- Some working functionality can be developed quickly and early in the life cycle.
- Results are obtained early and periodically.
- Parallel development can be planned.
- Progress can be measured.
- Less costly to change the scope/requirements.
- Testing and debugging during smaller iteration is easy.
- Risks are identified and resolved during iteration; and each iteration is an easily managed milestone.
- Easier to manage risk - High risk part is done first.
- With every increment, operational product is delivered.
- Issues, challenges and risks identified from each increment can be utilized/applied to the next increment.
- Risk analysis is better.
- It supports changing requirements.
- Initial Operating time is less.
- Better suited for large and mission-critical projects.
- During the life cycle, software is produced early which facilitates customer evaluation and feedback.

The disadvantages of the Iterative and Incremental SDLC Model are as follows −

- More resources may be required.
- Although cost of change is lesser, but it is not very suitable for changing requirements.
- More management attention is required.
- System architecture or design issues may arise because not all requirements are gathered in the beginning of the entire life cycle.
- Defining increments may require definition of the complete system.
- Not suitable for smaller projects.
- Management complexity is more.
- End of project may not be known which is a risk.
- Highly skilled resources are required for risk analysis.
- Projects progress is highly dependent upon the risk analysis phase.

A7. Write pros and cons of iterative and incremental model?

Iterative Model - Pros and Cons

The advantage of this model is that there is a working model of the system at a very early stage of development, which makes it easier to find functional or design flaws. Finding issues at an early stage of development enables to take corrective measures in a limited budget.

The disadvantage with this SDLC model is that it is applicable only to large and bulky software development projects. This is because it is hard to break a small software system into further small serviceable increments/modules.

The advantages of the Iterative and Incremental SDLC Model are as follows −

- Some working functionality can be developed quickly and early in the life cycle.
- Results are obtained early and periodically.
- Parallel development can be planned.
- Progress can be measured.
- Less costly to change the scope/requirements.
- Testing and debugging during smaller iteration is easy.
- Risks are identified and resolved during iteration; and each iteration is an easily managed milestone.
- Easier to manage risk - High risk part is done first.
- With every increment, operational product is delivered.
- Issues, challenges and risks identified from each increment can be utilized/applied to the next increment.
- Risk analysis is better.
- It supports changing requirements.
- Initial Operating time is less.
- Better suited for large and mission-critical projects.

- During the life cycle, software is produced early which facilitates customer evaluation and feedback.

The disadvantages of the Iterative and Incremental SDLC Model are as follows −

- More resources may be required.
- Although cost of change is lesser, but it is not very suitable for changing requirements.
- More management attention is required.
- System architecture or design issues may arise because not all requirements are gathered in the beginning of the entire life cycle.
- Defining increments may require definition of the complete system.
- Not suitable for smaller projects.
- Management complexity is more.
- End of project may not be known which is a risk.
- Highly skilled resources are required for risk analysis.
- Projects progress is highly dependent upon the risk analysis phase.

A8. Write application of spiral model?

Spiral Model - Design

- The spiral model has four phases. A software project repeatedly passes through these phases in iterations called Spiral model

Identification

- This phase starts with gathering the business requirements in the baseline spiral. In the subsequent spirals as the product matures, identification of system requirements, subsystem requirements and unit requirements are all done in this phase.
- This phase also includes understanding the system requirements by continuous communication between the customer and the system analyst. At the end of the spiral, the product is deployed in the identified market.

Design

- The Design phase starts with the conceptual design in the baseline spiral and involves architectural design, logical design of modules, physical product design and the final design in the subsequent spirals.

Construct or build

- The Construct phase refers to production of the actual software product at every spiral. In the baseline spiral, when the product is just thought of and the design is being developed a POC (Proof of Concept) is developed in this phase to get customer feedback.

- Then in the subsequent spirals with higher clarity on requirements and design details a working model of the software called build is produced with a version number. These builds are sent to the customer for feedback.

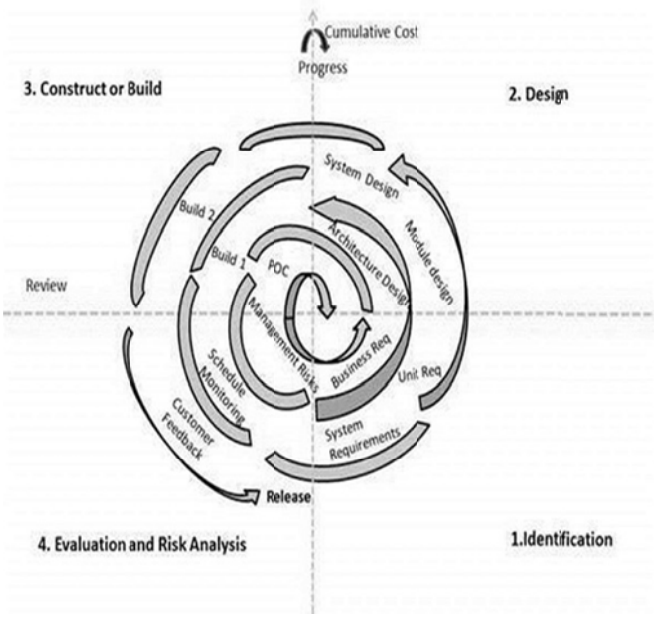
Evaluation and Risk analysis

- Risk Analysis includes identifying, estimating and monitoring the technical feasibility and management risks, such as schedule slippage and cost overrun. After testing the build, at the end of first iteration, the customer evaluates the software and provides feedback.

The following illustration is a representation of the Spiral Model, listing the activities in each phase.

Risk Analysis includes identifying, estimating and monitoring the technical feasibility and management risks, such as schedule slippage and cost overrun. After testing the build, at the end of first iteration, the customer evaluates the software and provides feedback.

The following illustration is a representation of the Spiral Model, listing the activities in each phase.



Spiral model -Application

The Spiral Model is widely used in the software industry as it is in sync with the natural development process of any product, i.e. learning with maturity which involves minimum risk for the customer as well as the development firms.

The following pointers explain the typical uses of a Spiral Model −

- When there is a budget constraint and risk evaluation is important.

- For medium to high-risk projects.
- Long-term project commitment because of potential changes to economic priorities as the requirements change with time.
- Customer is not sure of their requirements which is usually the case.
- Requirements are complex and need evaluation to get clarity.
- New product line which should be released in phases to get enough customer feedback.
- Significant changes are expected in the product during the development cycle.

A9. Explain working methodology of agile model and also write pros and cons?

Agile SDLC model is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product. Agile Methods break the product into small incremental builds. These builds are provided in iterations. Each iteration typically lasts from about one to three weeks. Every iteration involves cross functional teams working simultaneously on various areas like −

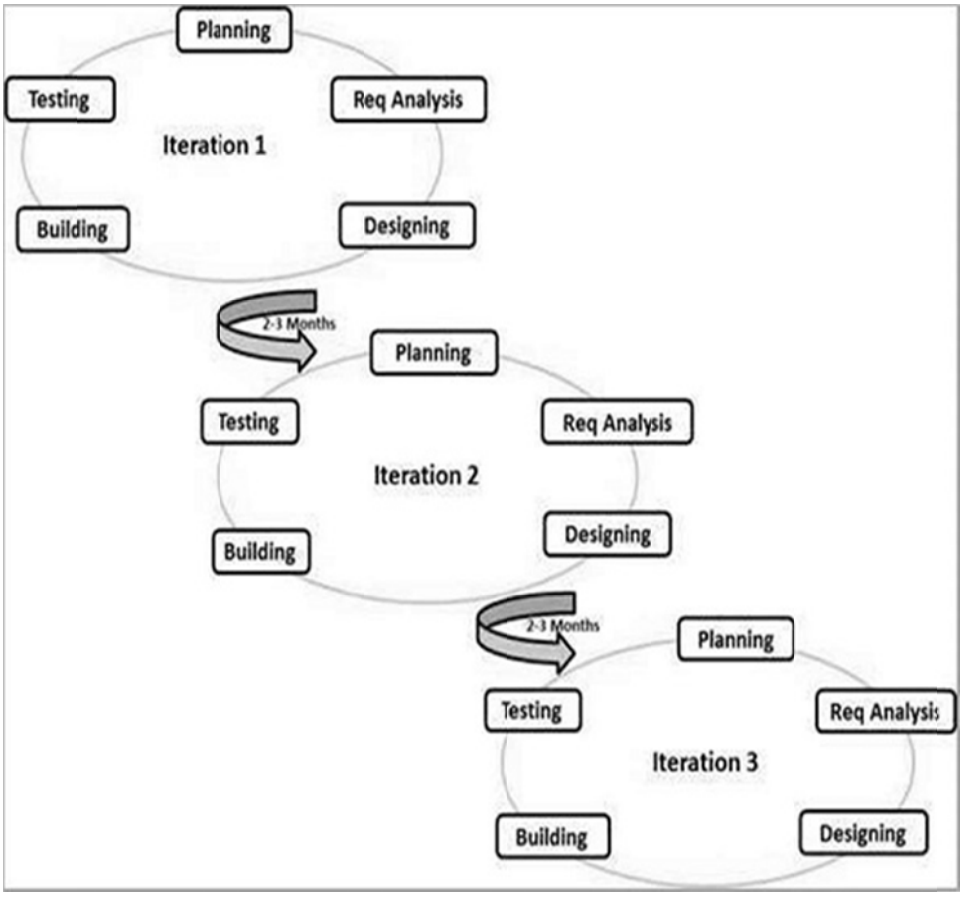
- Planning
- Requirements Analysis
- Design
- Coding
- Unit Testing and
- Acceptance Testing.

At the end of the iteration, a working product is displayed to the customer and important stakeholders.

Agile model believes that every project needs to be handled differently and the existing methods need to be tailored to best suit the project requirements. In Agile, the tasks are divided to time boxes (small time frames) to deliver specific features for a release.

Iterative approach is taken and working software build is delivered after each iteration. Each build is incremental in terms of features; the final build holds all the features required by the customer.

Here is a graphical illustration of the Agile Model −



Agile Manifesto principles –

- **Individuals and interactions** − In Agile development, self-organization and motivation are important, as are interactions like co-location and pair programming.
- **Working software** − Demo working software is considered the best means of communication with the customers to understand their requirements, instead of just depending on documentation.
- **Customer collaboration** − As the requirements cannot be gathered completely in the beginning of the project due to various factors, continuous customer interaction is very important to get proper product requirements.
- **Responding to change** − Agile Development is focused on quick responses to change and continuous development.

Agile model pros and cons

Agile methods are being widely accepted in the software world recently. However, this method may not always be suitable for all products. Here are some pros and cons of the Agile model.

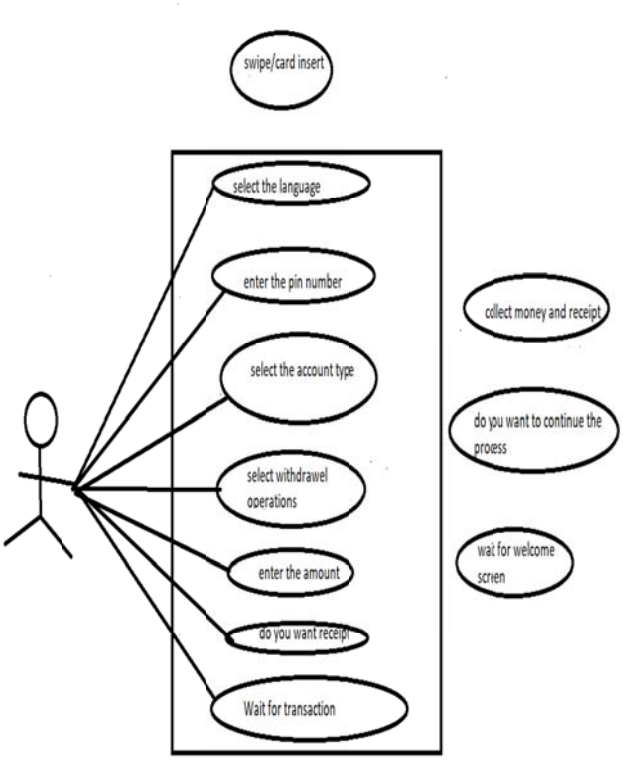
The advantages of the Agile Model are as follows −

- Is a very realistic approach to software development.
- 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.
- Enables concurrent development and delivery within an overall planned context.
- Little or no planning required.
- Easy to manage.
- Gives flexibility to developers.

The disadvantages of the Agile Model are as follows −

- Not suitable for handling complex dependencies.
- More risk of sustainability, maintainability and extensibility.
- 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.

A10. Create use case for ATM machine?



A12.What is the difference between software product and software project?

PROJECT

Also known as a software project comprises the steps involved in making a product before it is actually available to the market. The project can be handled by people which are as less as one person to the involvement of a lot of people (over 100). These are usually assigned by an enterprise and are undertaken to form a new product that has not already been made.

PRODUCT

The study of products is a part of Software engineering. The software is built by developers on requests from the customer. After the customer is satisfied with the development process, he launches the software by manufacturing it. This can be a problem-solving software or computer based system. This is the result of a project. The software project, when completed, is called a product after it is available to the market for usage.

PROJECT	**PRODUCT**
It comprises the steps involved in making a software before it is actually available to the market.	It is the manufacture of the project for users.
The main goal of a project is to form a new product that has not already been made.	The main goal of the product is to complete the work successfully (solve a specific problem).
Project is undertaken to form a new software.	Product is the final production of the project
It focuses on increasing the performance of the software that is being built.	A product focuses on the final result and the efficiency with which it can solve the given problem.
A project is done only once to get a new software.	A product can be made again and again for the purpose of distribution among users.
It is handled by the project managers.	It is handled by the product managers.

A13. Explain Polymorphism with example?

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.

TYPES OF POLYMORPHISM

- **Compile time polymorphism**
  - **Example:** *Method overloading*
- **Runtime polymorphism**
  - **Example:** *Method overriding*

Advantages of Polymorphism

- It helps programmers reuse code and classes once written, tested and implemented.
- A single variable name can be used to store variables of multiple data types (`float`, `double`, `long`, `int`, etc).
- It helps compose powerful, complex abstractions from simpler ones.

Example of polymorphism

Class Shape

{

public:

Shape(){}

Virtual void Draw(){count<<"Drawing a Shape"

};

Class Rectangle:public Shape

{

Public:

Rectangle(){}

virtual void Draw(){count<<"Drawing a rectangle"

};

Class Triangle:public Shape

```
{
Public:
Triangle(){}
Virtual void Draw(){count<<"Drawing  a Traingle"
};
Int main()
{
Shape *s;
  Rectange rec;
Triangle tri;
s=&rec;
s->Draw();
s=&tri;
s->Draw();
```

A14. Explain Abstraction with example?

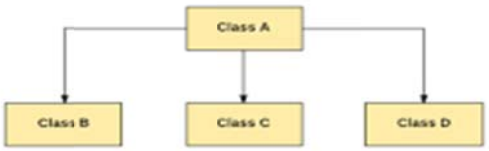
Abstraction is the process of taking away or removing characteristics from something in order to reduce it to a set of essential characteristics. Object oriented programming, abstraction is one of three central principles (along with encapsulation and inheritance). Through the process of abstraction, a programmer hides all but the relevant data about an object in order to reduce complexity and increase efficiency. In the same way that abstraction sometimes works in art, the object that remains is a representation of the original, with unwanted detail omitted. The resulting object itself can be referred to as an abstraction, meaning *a named entity made up of selected attributes and behavior specific to a particular usage of the originating entity*. Abstraction is related to both encapsulation and data hiding.

Example of Abstraction

```
 Package com.abstractdemo;

Public class MatrixImp1 extends Matrix

{

@Override

Public int[][]c = new int [a.length][b.length;j++)

{

      c[i][j] = a[i][j] + b[i][j];}

return c;
```

A15. Explain types of inheritance with example?

Inheritance is a mechanism in which one class acquires the property of another class. For example, a child inherits the traits of his/her parents. With inheritance, we can reuse the fields and methods of the existing class.

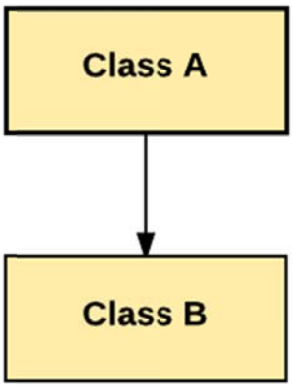


Type of inheritance

Here are the different types of inheritance

- Single inheritance

In Single Inheritance one class extends another class (one class only).

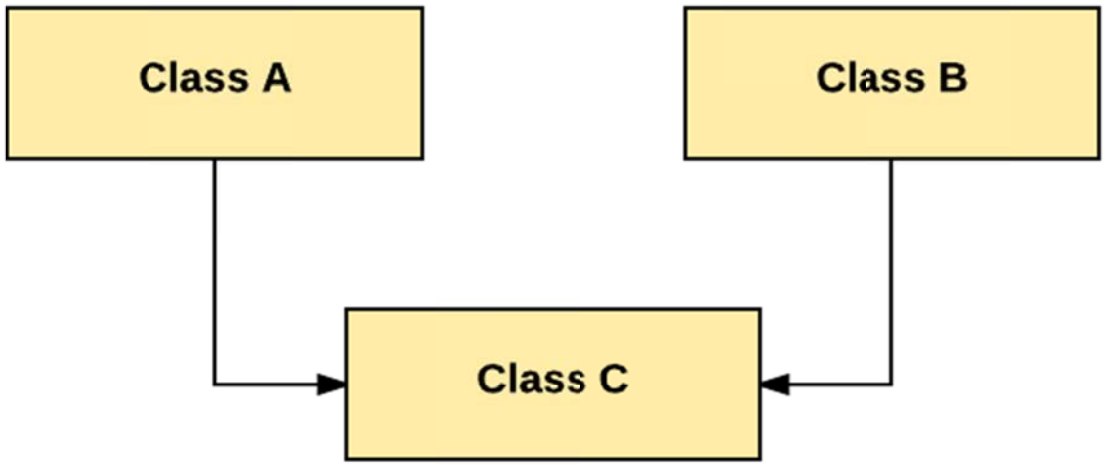


Single Inheritance

In above diagram, Class B extends only Class A. Class A is a super class and Class B is a Sub-class.

- Multiple Inheritance

Multiple Inheritance is one of the inheritance in Java types where one class extending more than one class. Java does not support multiple inheritance.

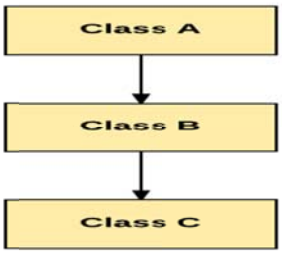


Java Multiple Inheritance

As per above diagram, Class C extends Class A and Class B both.

- Multilevel Inheritance

In Multilevel Inheritance, one class can inherit from a derived class. Hence, the derived class becomes the base class for the new class.

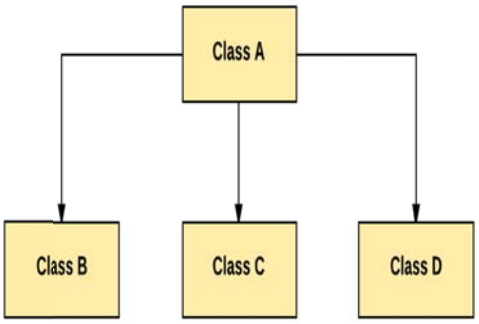


Multilevel Inheritance

As per shown in diagram Class C is subclass of B and B is a of subclass Class A

- Hierarchical Inheritance

In Hierarchical Inheritance, one class is inherited by many sub classes.

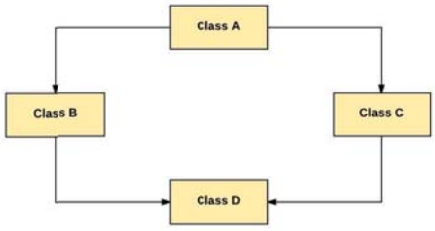


Hierarchical Inheritance

As per above example, Class B, C, and D inherit the same class A.

- Hybrid inheritance

Hybrid inheritance is one of the inheritance types in Java which is a combination of Single and Multiple inheritance.



Hybrid Inheritance in Java

As per above example, all the public and protected members of Class A are inherited into Class D, first via Class B and secondly via Class C.

Example of Inheritance

Shape .Java

```
Public class Shape
{
Public double getArea()
{
Return 0.5*base*height;
}
}
Private String color;
Public Shape (String color){
this.color=color;
}
Public String to String()
{
Return “shape of color=\””+ color +”\”;
}
Public double getArea()
{
System .err.println(“Shape unknown!Cnnot compute area!”);
return 0;
}
Return”shape unknown!Cannot compute area!”);
Return0:
}
```

```
}

Rectangle.java

public class Rectangle extends shape{

int length;

int width;

public Rectangle(String color, int length, int width)

{

Public String to String()

{

Return”Rectangle of length=”+length+”and width=”+ width + “,subclass of” + super.toString();

}

Public double getArea()

{

Return length”width;

}

Triangle .java{

Public class Triangle extends Shape

{

Private int base;

Private int height;

Public Triangle(String color, int base, int height)

{

Super(color);

This.base = base;

This.height = height;
```

```
}

public String toaString()

{

Return"Triangle of base="+ base + "and height=" +height +",subclass of+super.toString();

}
```

A16.What is inner join?

SQL join statements allow us to access information from two or more tables at once they also keep our database normalized. Normalization allows to keep data redundancy low so that we can decrease the amount of data anomalies in our application when we delete or update a record.

- A Join clause allows us to combine rows from two or more tables based on a related column.

Type of Joins

The type of join statement we can use depends on our use case. There are four different types of join operations:

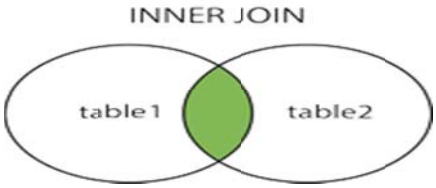
- INNER JOIN: Returns dataset that have matching values in both tables.
- LEFT JOIN: Returns all records from the left table and matched records from the right table.
- RIGHT JOIN: Returns all records from the right table and matched records from the left table
- FULL JOIN: Returns all records when there is a match in either the left and right table.

INNER JOIN

The inner join keyword selects records that have matching values in both tables.

INNER JOIN Syntax

```
SELECT column_name()

FROM table1

INNER JOIN table2

ON table1.column_name = table2.colunm_name;
```



A17. Explain left and Right join with example?

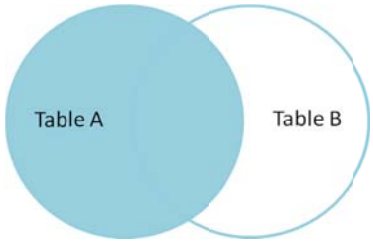
LEFT JOIN

The left join command returns all rows from the left table, and the matching rows from the right table. The result is NULL from the right side, if there is no match.

The following SQL will select all customers, and any orders they might have:

EXAMPLE:

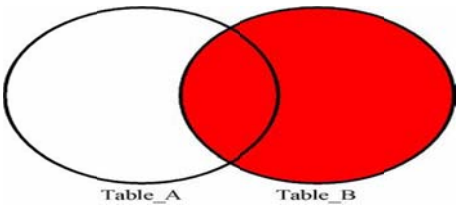
```
SELECT Customers. CustomerName, Orders. OrdersID

FROM Customers

LEFT JOIN Orders ON Customers. CustomerID = Orders. CustomerID

ORDER BY Customers.CustomerName;
```



RIGHT JOIN:

RIGHT JOIN is similar to LEFT JOIN. This join returns all the rows of the table on the right side of the join and matching rows for the table on the left side of join. The rows for which there is no matching row on left side, the result-set will contain *null*. RIGHT JOIN is also known as RIGHT OUTER JOIN.**Syntax**

Note: We can also use RIGHT OUTER JOIN instead of RIGHT JOIN, both are same.



(RIGHT JOIN)

EXAMPLE:

SELECT Orders.OrderID, Employees. LastName,Employees.FirstName

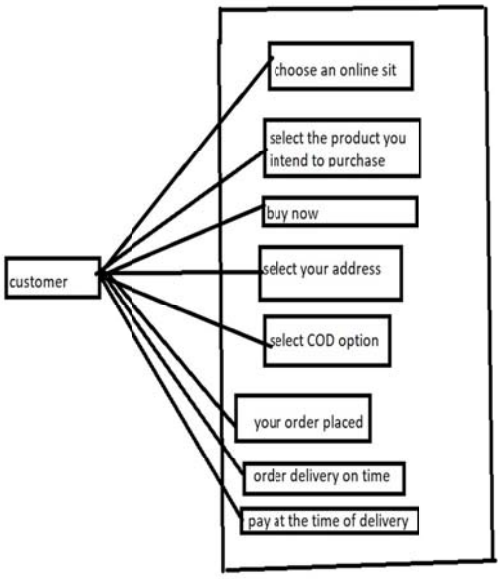
Employees. FirstName

FROM Orders

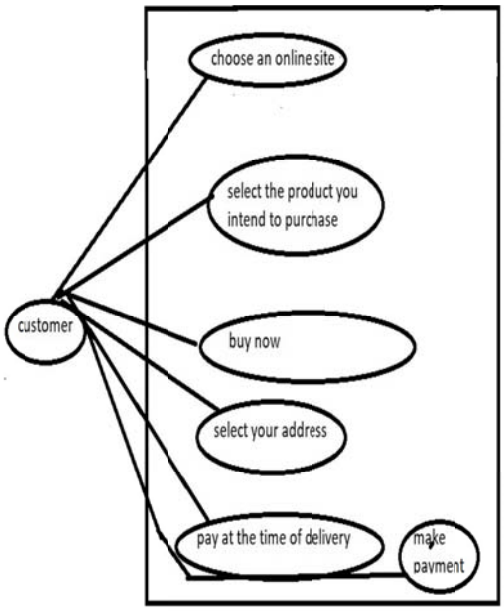
RIGHT JOIN Employees ON Orders.EmplyeeID=Employees.EmployeeID

ORDER BY ORDERS.ORDER ID;

A20. Draw usecase on Online shopping product using COD?



A19. Draw Use case on Online shopping product using payment gateway?



A20.Draw use case on Property Portal web based project?