Software Testing Assignment Module 1 (Fundamental) Basic

B1. What is Software Engineering?

Ans. Software engineering is a method of software development that is based on a systematic engineering approach. A software engineer designs, develops, maintains, tests, and evaluates computer software using software engineering concepts.

B2. What is SDLC?

Ans. SDLC is a process for planning, implementation, testing, documentation, deployment, and ongoing maintenance and support that is used in the development of a software product. A variety of development models are available.

A Software Development Life Cycle (SDLC) is a set of processes, or phases, that act as a model for the development and management of software applications.

The method used in the SDLC process varies by industry and organization, but standards like ISO/IEC 12207 provide processes for developing a software lifecycle and providing a mechanism for developing, acquiring, and customizing software systems.

B3. What is Software Development Methodology?

Ans. A procedure or set of processes used in software development is known as a software development methodology. Again, it's fairly broad, but it includes things like design and development phases. There are different ways of thinking of things like waterfalls as non-iterative processes. It usually takes the form of clearly defined phases. Its purpose is to define the how of a piece of software's life cycle.

It is also a form of communication that has been codified. So you're basically developing a set of standards among a group of people that say, “This is how we're going to operate, and this is how we're going to communicate information between each of us in particular ways," whether that means documentation, discussion, or drawings on paper.

B4. What is agile methodology?

Ans.

- The Agile SDLC model combines iterative and incremental process models with a focus on process adaptation and customer satisfaction through the delivery of a working software product quickly.
- Agile development methods divide the product into small, incremental builds.
- These builds are delivered in iterations, with each iteration lasting one to three weeks.
- Cross-functional teams work on planning, requirements analysis, design, coding, unit testing, and acceptance testing during each iteration.
- A workable product is presented to the client and key stakeholders at the end of each iteration.

B5. What is the use-case?

Ans.

- A use-case is a definition of a set of activities that a system (or other entity) can do while engaging with the system's actors, including variants. For example, Purchase a DVD on the internet,
- A scenario is a specific sequence of action occurrences that begins with a known initial state. Ex: connect to my DVD.com Ex: go to the “search” page.
- Use cases are very easy-to-understand techniques for describing software or system behaviour.
- A use case is a textual description of all the different ways that the software or system's targeted users could interact with it.
- A use case is a method to collect the potential requirements of a new system or software development.
- Each use case contains one or more scenarios that describe how the system should interact with the end user or another system in order to fulfil a given business goal.
- In most usage scenarios, technical terms are avoided in favour of the end user's or domain expert's language. Requirements engineers and stakeholders frequently collaborate on use cases.
- Use cases do not define the system's core workings or how it will be implemented.
- They simply display the steps a user takes to complete a task.

- This is how all the ways that people interact with a system may be explained.

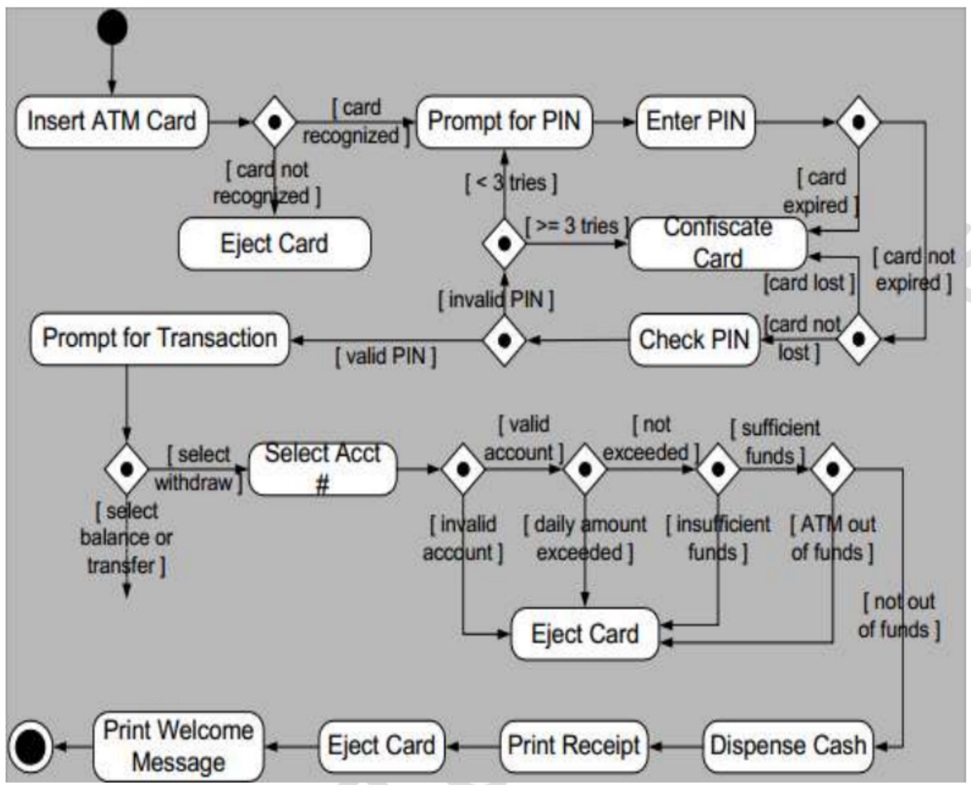
B6. What are "Activity Diagrams"?

Ans.

- The flow of activities is shown in activity diagrams.
- User-level steps should be modelled in activities.
- There are two types of nodes: There are two types of nodes:
    - Action states
    - Sequential branches

- In the activity diagram, use case descriptions are converted into action state nodes.
- Other options are sequential branch nodes (sbn)
- Edges are flow between stages.

ATM Withdraw Activity Graph.



B7. What is SRS?

Ans. A software requirements specification (SRS) is a detailed description of how the system will behave after it is built. It contains a set of use cases that describe all the software's interactions with users.

Functional requirements are another name for use cases. The SRS also includes nonfunctional (or supplemental) requirements in addition to use cases.

IEEE 830-1998 specifies the procedures for defining software requirements that should be used. A software requirements specification can have a variety of structures, content, and attributes, according to this standard.

B8. What is programming?

Ans. Programming is the process of writing a set of instructions that tell a computer how to do something. Programming can be done in a variety of languages, including JavaScript, Python, and C++.

However, programming is not as simple as it appears, because good software takes time to develop. It requires a great deal of expertise, as well as a great deal of insight from the programmers.

B9. What is an oops?

Ans. Object-oriented programming (OOP) is a programming model characterized by the identification of object classes and their related methods (functions). It also covers concepts like attribute and method inheritance. Object-oriented through of a web of interconnected objects, each of which maintains its own state.

B10. Write Basic Concepts of oops?

Ans. Encapsulation, abstraction, inheritance, and polymorphism are four key principles in object-oriented programming. While these principles may appear difficult, understanding how they work in general can help you understand the basics of an OOP computer program.

B11. What is the object?

Ans. An object is a unique instance of a class that exists in the real world, and a class can have multiple objects (or instances). An object is a unique, recognized item, unit, or thing. Or a real or abstract entity having a well-defined function in the domain of the problem.

An "object" is anything to which a concept applies. The core unit of object-oriented programming (OOP) is the object, which includes both data and functions that operate on data.

B12. What is a class ?

Ans. A class is a blueprint for creating objects (a specific data structure), providing starting values for state (member variables or attributes), and performing behaviour in object-oriented programming (member functions or methods). The class keyword is used to build user-defined objects.

A class can be considered of as a blueprint, definition, or template for an object that describes its features and behaviour but does not exist. An object is a unique instance of a class that exists in the real world, and a class can have multiple objects (or instances).

B13. What is RDBMS ?

Ans. RDBMS (Relational Database Management System) is an abbreviation for Relational Database Management System. All modern database systems, such as MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access, are built on top of RDBMS.

The relational database management system (RDBMS) is a database management system (DBMS) based on E. F. Codd's relational model.

Most of today's databases are relational:
database contains 1 or more tables.
table contains 1 or more records.
record contains 1 or more fields.
fields contain the data

So why is it called "relational"? tables are related (joined) based on common fields

B14. What is SQL?

Ans. SQL stands for Structured Query Language, and it is a computer language used to store, manipulate, and retrieve data from relational databases. The Relation Database System standard language is SQL. SQL is the standard database language used by all relational database management systems, including MySQL, MS Access, Oracle, Sybase, Informix, Postgres, and SQL Server.

They also use several languages, such as

MS SQL Server using T-SQL, ANSI SQL
Oracle using PL/SQL,
MS Access version of SQL is called JET SQL (native format) etc.

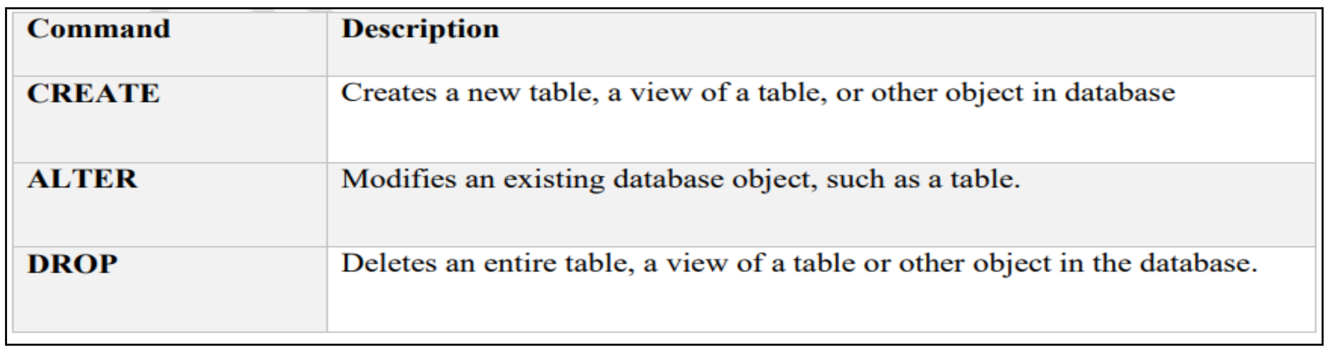
B15. Write SQL Commands

Ans. There are four types of SQL commands, such as below.

- DDL (Data Definition Language)
- DML (Data Manipulation Language)

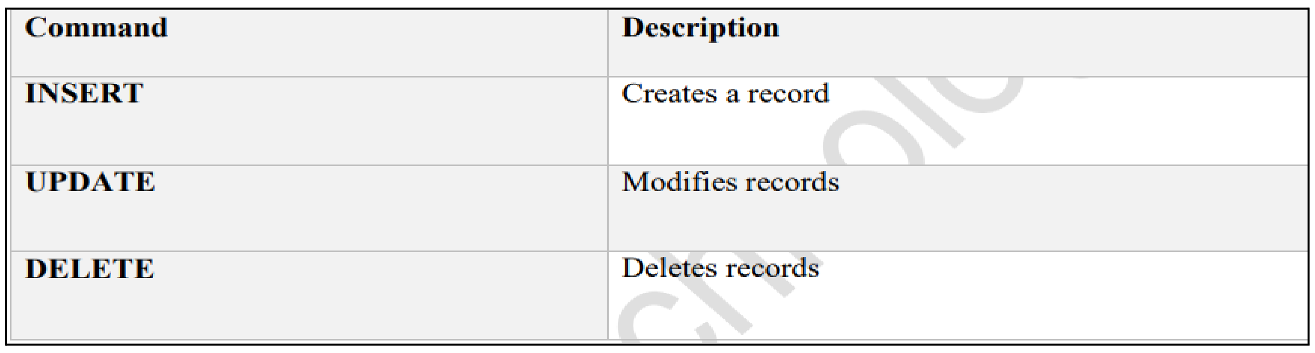
- DCL (Data Control Language).
- DQL (Data Query Language)

DDL (Data Definition Language)



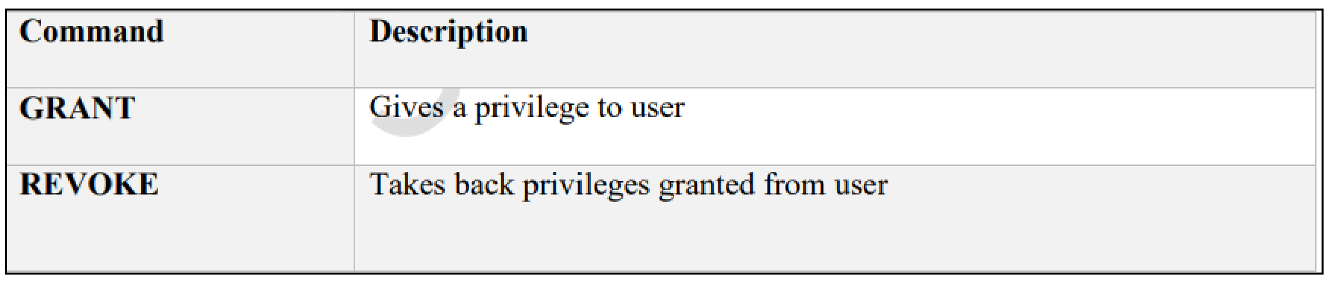
Command	Description
CREATE	Creates a new table, a view of a table, or other object in database
ALTER	Modifies an existing database object, such as a table.
DROP	Deletes an entire table, a view of a table or other object in the database.

DML - Data Manipulation Language



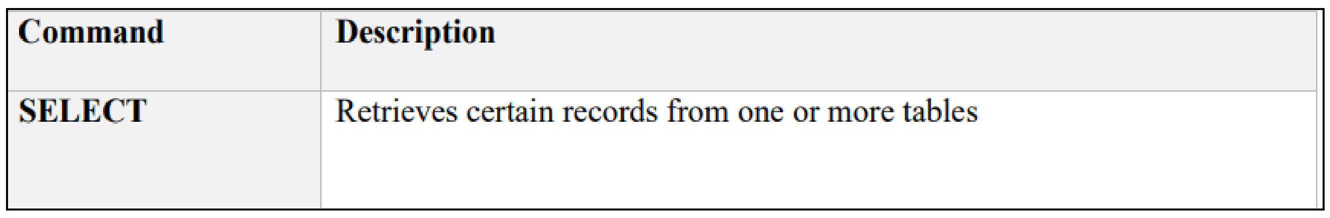
Command	Description
INSERT	Creates a record
UPDATE	Modifies records
DELETE	Deletes records

DCL - Data Control Language



Command	Description
GRANT	Gives a privilege to user
REVOKE	Takes back privileges granted from user

DQL (Data Query Language)



Command	Description
SELECT	Retrieves certain records from one or more tables

B16. Draw Use case on Online book shopping

Ans.

B17. Draw Use case on online bill payment system (paytm)

Ans.

Software Testing Assignment Module 1 (Fundamental) Intermediate

I1. Write SDLC phases with basic introduction.

Ans. Introduction: An SDLC is a process that describes the planning, implementation, testing, documentation, deployment, and ongoing maintenance and support of a software product. There are a variety of development models from which to choose.

A Software Development Life Cycle (SDLC) is a set of processes, or phases, that act as a framework for the development and management of software applications.

Requirement Gathering: The requirement gathering and analysis phase is the most important part of the SDLC because it is here that the project team understands what the clients expect from the project. The project team meets with the client to define each requirement in detail during the requirements gathering sessions.

Analysis Phase: The analysis phase defines the requirements of the system, independent of how these requirements will be accomplished. The problem that the client is trying to solve is defined in this step. A requirement document is the delivered outcome at the conclusion of this phase.

Design Phase : The Software Development Life Cycle's Design Process is important. Design decisions are based on the list of requirements you created during the defining process. One or more designs are generated throughout the design process to achieve the project's outcome.

Implementation Phase : The team either builds the components from scratch or via composition during the implementation phase. The team should build exactly what was asked given the architecture document from the design phase and the requirement document from the analysis phase. However, there is still scope for creativity and flexibility.

Testing Phase : In the software development lifecycle (SDLC), the testing phase is when you focus on investigation and discovery. During the testing phase, developers check to see if their code and programming meet the requirements of the client. A customer who's really happy with a product's quality will continue with it and wait for new features in the next iteration. A system's quality is a defining feature that shows its level of perfection.

- Regression Testing
- Internal Testing
- Unit Testing
- Application Testing
- Stress Testing

Maintenance Phase : Software maintenance is the process of improving and optimizing deployed software (software release), as well as fixing defects, and is one of the activities in software development. In the field of software development, software maintenance is also a part of the System Development Life Cycle (SDLC). After the deployment of software into the field, the maintenance phase begins. When it comes to software development, one of the phases in the System Development Life Cycle (SDLC) is software maintenance. The maintenance phase begins once the software has been deployed into the field.

I2. Explain the types of requirements.

Ans. There are several ways to classify requirements. The following are some examples of technical management requirement classifications:

- Customer Requirements
- Functional Requirements
- Non-Functional Requirements

Customer Requirements : Customers are people who perform the eight primary roles of system design, with the operator functioning as the most important client. Operational requirements will define the basic needs and, at the very least, respond to the issues posed in the list below.

- Where will the system be used? Operational distribution or deployment:
- How will the system achieve its mission objective, according to the mission profile or scenario?
- What are the important system parameters for completing the challenge in terms of performance and related factors?
- How will the various system components be used in different environments?

Functional Requirements: In the system development process, functional requirements are important. These specifications include the suggested system's technical specifications, system design parameters and standards, data manipulation, data processing, and calculation modules, and so on.

For example, the following are the requirements of Google Email Service.

- The system must be capable of receiving emails.
- The system should be capable of sending emails.
- The system should allow users to create new folders.
- The system must allow you to filter emails in different folders.
- The system that supports the ability to attach various types of attachments.
- The system shall support the ability to create and maintain address book
- The system must allow the creation of an unlimited number of user accounts, each with a unique email address.

Non-Functional Requirements : Non-functional requirements, on the other hand, are factors that can be used to judge a system's functioning rather than specific actions. Non-functional requirements are qualities or standards that the developing system must have or meet, but which are not tasks that the system will perform.

Examples of non-functional requirements for a system include:

- The system must be built for a total cost of $1,050,000.00 to be deployed.
- The system must run on Windows Server 2003.
- The system must be secured against Trojan attacks.

A software development technique helps in the identification, documentation, and fulfilment of requirements. The following categories can be used to categorize non-functional requirements:

- Usability
- Reliability
- Performance
- Security

13. States the importance of the design phase?

Ans. The Design Phase's aim is to convert the requirements into thorough and complete system design specifications. The development team starts the development phase once the design has been accepted. In SDLC, the design phase is a stage where software developers define the technical details of the product. Depending on the project, these details can include screen designs, databases, sketches, system interfaces, and prototypes.

14. What are the tasks performed in the coding phase?

Ans. Developers study the feasibility of each coding language during the coding phase and start programming according to coding specifications. The product will not perform according to the customer's specifications without proper coding, and new codes may be needed.

During this phase, developers start writing code in the programming language of their choice, such as C, C++, Java, PHP, and others, in order to develop the full system. Tasks are divided into parts or modules and given to different developers during the coding phase. It is the most time-consuming part of the software development life cycle.

In this phase, the decompiler needs to follow certain predefined coding guidelines. They also need to use program compilers like interpreters and debuggers to generate and implement the code.

15. Briefly explain Testing Phase

Ans. In the software development lifecycle (SDLC), the testing phase, that's where you focus on research and discovery. During the testing phase, developers check to see if their code and programming meet the needs of the customer. While it is impossible to fix all the faults found during the testing phase, the results of this phase can be used to reduce the number of errors in the software program.

The project team must create a test plan before testing can start. The test strategy outlines the types of testing you'll do, the resources you'll need, how the software will be tested, who should be the testers at each stage, and test scripts, which are the instructions each tester follows when testing the product. The use of test scripts ensures consistency in the testing process.

During the testing phase, there are numerous forms of testing, including quality assurance testing (QA), system integration testing (SIT), and user acceptability testing (UAT). Testing is a separate phase that comes after the implementation is complete and is carried out by a different team.

This strategy has value; it is difficult to detect one's own errors, and a fresh eye can detect obvious errors much more quickly than someone who has read and re-read the text many times.

Unfortunately, assigning (alternative) testing to another team's performance in the deployment team's lack of (dull) attitude to quality is problematic. If the teams are to be known as craftsmen, they must be in charge of maintaining high quality throughout all phases. To ensure quality, a change in attitude is required. Testing is frequently based on a regression approach divided into many key focuses, including internal, unit, application, and stress, whether done after the fact or continuously.

16. Explain Phases of the waterfall mode

Ans. Waterfall Model, the Software Development Life Cycle: The first process model introduced was the waterfall model. A linear-sequence life cycle model is another name for it. It's easy to understand and use. In a waterfall model, each phase must be finished before the next one can start in a waterfall model, and the phases must not overlap. The waterfall model was the first SDLC method used in software development.

The waterfall model represents the software development process in a sequential and linear manner. This indicates that any step of the development process may start only when the previous one is finished. The phases in this waterfall model do not cross.

The Waterfall Model was the first SDLC model to be widely used in software development to assure project success. The entire software development process is separated into several phases in “The Waterfall” technique. Typically, the result of one phase serves as the input for the next step in this waterfall model.

Every piece of software developed is unique and requires a unique SDLC strategy, depending on internal and external factors. The use of the waterfall model is best suited in the following situations:

- Requirements are very well documented, clear, and fixed.
- The product definition is stable.
- Technology is understood and is not dynamic.
- There are no requirements that are unclear.
- To support the product, there are plenty of resources with the necessary experience.

17. Write phases of spiral model

Ans. The spiral SDLC model includes a combination of a continuous software development methodology with a waterfall model. This model is best suited for large, complicated, and expensive projects. In its graphical representation form, we have a coil with multiple cycles or loops. The number of cycles varies with each project and is generally decided by the project manager. Each spiral cycle represents a milestone in the software development process.

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.

- **Determine objectives and find alternate solutions** : During this phase, you'll gather and analyse requirements. Objectives are specified, and many alternative solutions are provided based on the requirements.
- **Risk Analysis and Resolving:** All proposed solutions are analysed in this sector, and any potential danger is detected, analysed, and resolved.
- **Developing and testing:** The practical application of the various functionalities is included in this phase. After that, all the implemented features are thoroughly tested.
- **Review and planning of the next phase**: The customer evaluates the program at this phase. It also includes risk identification and monitoring, such as huge costs or timetable delays, before moving on to phase planning.

18. Write agile manifesto principles

Ans.

- Individuals and interactions: Self-organization and motivation, as well as interactions like co-location and pair programming, are critical in agile development.
- Working software: Instead of relying just on documentation, the easiest way to communicate with a customer is to demonstrate functional software.

- Customer collaboration: Because requirements cannot be fully gathered at the start of a project due to a range of variables, continuous customer involvement is critical to obtaining accurate product specifications.
- Responding to change: Agile development promotes rapidly changing responses and continuous improvement.

I9. What is an Actor in a Use Case?

Ans. An actor is a type of agent that interacts with the product.

I10. How many kinds of nodes are in activity diagrams? Which?

Ans. There are two kinds of nodes:

- Action states
- Sequential branches

I11. What is Encapsulation?

Ans. Encapsulation is the process of storing everything an object needs in a way that keeps it hidden from other things. Other objects are normally unable to access the internal state.

Encapsulation is the process of putting data and the functions that operate on it in the same location. It's not always clear which functions work on which variables when dealing with procedural languages, but object-oriented programming gives you a framework for putting data and relevant functions together in the same object.

I12. What is inheritance?

Ans. The word "inheritance" refers to when one class acquires the features of another. This is referred to as a "is a" connection. Code reusability is one of the most beneficial features of object-oriented programming. As the title suggests, inheritance is the process of creating a new class from an existing base class.

I13. What is polymorphism?

Ans. Polymorphism means "having many forms." The word polymorphism means having many forms. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form.

Polymorphism allows us to carry out a single action in multiple ways. Polymorphism, in other words, allows you to specify one interface and have several implementations. The term "polymorphs" means "many forms," so it means "many forms."

I14. What is abstraction?

Ans. One of the fundamental ideas of object-oriented programming (OOP) languages is abstraction. Its primary purpose is to deal with complexity by hiding unnecessary details from the user. This allows the user to build more complicated logic on top of the provided abstraction without having to understand or even consider all the hidden complexity.

I15. Why SQL?

Ans. SQL is used to create, define, and implement databases, as well as to execute different activities on them. SQL is also used to access, maintain, and manipulate databases that have previously been built. SQL is a well-designed database language for entering, modifying, and extracting data.

I16. Write SQL commands in detail

Ans. There are four types of SQL commands, which are described as below

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

DDL - Data Definition Language

Commands	Descriptions
CREATE	Creates a new table, a view of a table, or other object in database
ALTER	Modifies an existing database object, such as a table.
DROP	Deletes an entire table, a view of a table or other object in the database.

DML - Data Manipulation Language

Command	Description
INSERT	Creates a record
UPDATE	Modifies records
DELETE	Deletes records

DCL - Data Control Language

Command	
GRANT	Gives a privilege to user
REVOKE	Takes back privileges granted from user

DQL - Data Query Language

Command	Description
SELECT	Retrieves certain records from one or more tables

117. What is join?

Ans. A join is a SQL action that creates a relationship between two or more database tables by establishing the connection between them based on matching columns. Join commands are used in the majority of complicated SQL database management queries. Joins come in a variety of styles.

118. Write type of joins.

Ans. Here are the different types of JOINS in SQL:

- **(INNER) JOIN:** Returns records that have matching values in both tables.
- **LEFT (OUTER) JOIN**: Returns all records from the left table, and the matched records from the right table.
- **RIGHT (OUTER) JOIN**: Returns all records from the right table, and the matched records from the left table.
- **Full (OUTER) JOIN:** Returns all records when there is a match in either the left or right table.

119. Draw Use case on Online payment transfer via bank to bank

Ans.

120. Draw use case on Student Registration Process on University

Ans.

Software Testing Assignment Module 1 (Fundamental)Advanced

A1. Explain phases of SDLC in detail.

Ans. **SDLC Phases**

SDLC Phases

Requirements Collection/Gathering	Evaluate the needs of the customer.
Analysis	Specify and model the requirements- "What"
Design	Specify and model a solution -"Why"
Implementation	Build a Software Solution
Testing	Check the solution for compliance with the specifications.
Maintenance	Repair all defects and adapt the solution to meet the new requirements.

Requirement Gathering: The requirement gathering and analysis phase is the most important part of the SDLC because it is here that the project team understands what the clients expect from the project. The project team meets with the client to define each requirement in detail during the requirements gathering sessions.

There are the following steps for the requirement gathering process:

- Features
- Usage scenarios
- Even if requirements are documented in writing, they may be incomplete, unclear, or incorrect.
- Requirements will Change!
- During the project, user and business requirements vary.
- Natural language is used to define requirements, which is supported with diagrams and tables.

Types of Requirements:

- Functional Requirements: It can include to describe the services or functions of the system, calculate the amount of sales and, on the server, update the database. tax due on a purchase.

- Non-Functional Requirements : Non-Functional Requirements: These are limitations placed on the system or development process. Non-functional requirements may be more important than functional requirements in some cases. The system is useless if these are not met.

Analysis Phase: The analysis phase defines the requirements of the system, independent of how these requirements will be accomplished. The problem that the client is trying to solve is defined in this step. A requirement document is the delivered outcome at the conclusion of this phase.

This document will ideally define what is to be built in a clear and exact manner. This analysis represents the "what" phase. By defining goals, the requirement specifications capture the requirements from the customer's perspective. This step takes the requirements documents from the previous phase and transfers them into the architecture. Components, interfaces, and behaviours are defined by the architecture.

Design Phase : The Software Development Life Cycle's Design Process is important. Design decisions are based on the list of requirements you created during the defining process. One or more designs are generated throughout the design process to achieve the project's outcome.

During the design phase, one or more designs are created that appear to be able to achieve the project's goal. Dioramas, drawings, flow charts, site trees, HTML screen designs, prototypes, picture impressions, and UML schemas are some products of the design phase, depending on the project's theme.

Implementation Phase : The team either builds the components from scratch or via composition during the implementation phase. The team should build exactly what was asked given the architecture document from the design phase and the requirement document from the analysis phase. However, there is still scope for creativity and flexibility.

Testing Phase : In the software development lifecycle (SDLC), the testing phase is when you focus on investigation and discovery. During the testing phase, developers check to see if their code and programming meet the requirements of the client. A customer who's really happy with a product's quality will continue with it and wait for new features in the next iteration. A system's quality is a defining feature that shows its level of perfection.

- Regression Testing
- Internal Testing
- Unit Testing
- Application Testing
- Stress Testing

Maintenance Phase : Software maintenance is the process of improving and optimizing deployed software (software release), as well as fixing defects, and is one of the activities in software development. In the field of software development, software maintenance is also a part of the System Development Life Cycle (SDLC). After the deployment of software into the field, the maintenance phase begins. When it comes to software development, one of the phases in the System Development Life Cycle (SDLC) is software maintenance. The maintenance phase begins once the software has been deployed into the field.

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

Ans. Project planning and requirements are the most fundamental phases of the SDLC. Without understanding the initial requirements, no software team can develop a solution that gives value to clients. Usually, the senior members in a project team are responsible for carrying out requirement analysis.

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

Ans. All stakeholders are involved in the phase of the waterfall model

A4. Explain the team Requirement Gathering concerning SDLC

Ans. The most important phase of the 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.

For project managers and internal stakeholders, gathering project requirements is the most critical step in the SDLC. During this phase, the client describes the project's expectations, including who will use the product, how the customer will use the product, and any special customer requirements related to the software. To give the requirements, the client meets with the business managers and analysts. It's necessary for the project team to understand the customer's requirements because this information is crucial for developing the product that the customer has requested.

Three types of problems can arise:

- Lack of clarity: It is hard to write documents that are both precise and easy-to-read.
- Requirements confusion: functional and non-functional requirements tend to be intertwined.
- Requirements Amalgamation: Several requirements may be expressed together.

After the customer provides requirements for the product requirements, the project manager and members of the project team begin to analyze the requirements. The business managers analyse each requirement to ensure it can be included in the software without causing breaks or problems with system functionality.

A5. What issues are faced in the waterfall model?

Ans. Waterfall development has the disadvantage of reducing reflection and modification. It's quite tough to update something that wasn't well-documented or considered in the design stage once an application has reached the testing stage.

The major disadvantages of the waterfall model are as follows:

- At the end of the life cycle, no working software is developed.
- There is a great deal of risk and uncertainty.
- This is not a good model to use.
- Poor model for long-term projects.
- It is not recommended for projects with a moderate to high risk of change in requirements.
- As a result, this process model has a high level of risk and uncertainty.
- Within stages, it is difficult to evaluate development.
- Can not adapt to changing needs.

A6. Write applications of iterative and incremental model?

Ans. In this model, you can start with some software specifications and develop the first version of the software. If changes to the software are required after the first version, a new version with a new iteration is developed. The iterative model's releases are completed in iterations, which are exact and fixed periods of time.

The Iterative Model allows you to go back in time and examine earlier phases where the changes were made. When the Software Development Life Cycle (SDLC) process is completed, the project's final output is renewed.

The iterative approach starts with a basic implementation of a portion of the software requirements and continues to iteratively improve the evolving versions until the entire system is built. Design changes and additional functional features are introduced with each iteration. The primary concept behind this strategy is to build a system in little pieces over time (iterative) (incremental).

A7. Write the pros and cons of iterative and incremental models.

Ans. **Advantages**

- Early diagnosis and correction of potential flaws
- Early in the project lifecycle, functional prototypes are developed.
- There is less time spent documenting, and more time spent designing.
- It's simple to track progress.
- Changes to the scope of a project are less expensive and easier to implement.
- Because the modules are small, testing is made easier.
- The majority of risks may be identified throughout the iteration, and greater risks can be handled as soon as possible.
- It is simple to handle successive iterations as milestones.
- Operating time is reduced.

Disadvantages

- More resources may be required.
- It's possible that you'll need more active project management.
- Due to the lack of a complete requirements' specification for the entire system, system architecture issues may be proven to be a main difficulty.
- It may be difficult to set a deadline for the project's completion.
- For risk analysis, highly skilled people are required.

A8. Write applications of spiral model

Ans. The spiral model is widely used in the software business because it is in line with any product's natural growth process, i.e., learning as it develops, and it also involves minimal risk for both the client and the development firm. The spiral model is commonly used for the following purposes:

When there is a financial restriction, risk assessment is critical.

- For medium-to high-risk projects.
- Long-term project commitment, given the possibility of changing economic priority as requirements change over time
- The customer is not sure of their requirements, which is usually the case.
- Requirements are complex and require evaluation to get clarity.
- New product line that should be introduced in phases to provide enough client input.
- During the development period, there will be significant changes to the product.

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

Ans. The Agile methodology divides a project into phases to make it easier to manage. It requires ongoing collaboration with stakeholders as well as continuous improvement at all levels. Teams go through a planning, execution, and evaluation phase once the work starts.

The Agile SDLC model combines iterative and incremental process models with a focus on process flexibility and customer satisfaction through the delivery of a functioning software product quickly.

Every iteration involves cross-functional teams working on planning, requirements analysis, design, coding, unit testing, and acceptance testing at the same time. Every project should be treated differently, according to the Agile model, and current methods should be customized to the project's demands. To deliver particular features for a release, tasks in agile are broken into time boxes (short time spans).

Pros

- is a very realistic approach to software development.
- It promotes teamwork and cross-training.
- Functionality can be developed rapidly and demonstrated.
- Resource requirements are minimal.
- Suitable for fixed or changing requirements,
- delivers early partial working solutions.
- Minimal rules, documentation, are easily employed.
- Little or no planning is required.
- easy to manage.
- It gives flexibility to developers.

Cons

- Not suitable for handling complex dependencies.
- more at risk of sustainability, maintainability, and extensibility.
- It will not work unless you have a detailed plan, an agile leader, and an agile PM practice.
- The scope, functionality to be delivered, and changes to fulfil deadlines are all controlled by strict delivery management.
- It depends heavily on customer interaction, so if the customer is not clear, the 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 a lack of

A10. Create a use-case for the ATM machine

Ans.

A11. Create Use-case for E-shopping User module.

Ans.

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

Ans.

Sr. No.	Project	Products
1	It includes all of the processes involved in developing software before it is released.	It's the project's development for users.
2	A project's main goal is to develop a new product that hasn't been created before.	The product's main goal is to effectively perform the task
3	A new software is being developed as part of a project.	The project's final outcome is known as the product.
4	It focuses on improving the performance of the software being developed.	A product is Acrobat Reader focused with the end result and the speed with which it can address the problem.
5	It is handled by the project managers.	It is handled by the product managers.
6	A software application that is developed by a company with its own budget.	A software application that is developed by a company with the budget from a customer
7	Examples of software products: Tally (by TCS), Acrobat reader/writer (Adobe), Internet Explorer (MS), Finale (Infosys), Windows (MS), QTP (HP) etc.	Example for software projects: A separate application ordered by a manufacturing company to maintain its office inventory, etc.

A13. Explain polymorphism with an example.

Ans. Polymorphism, in simple terms, is the capacity of a message to be presented in several forms. A real-life example of polymorphism is when a person has many characteristics at the same time. At the same time, he is a parent, a spouse, and an employee.

Polymorphism means "having many forms". It allows different objects to respond to the same message in different ways, the response being specific to the type of object.

The most important aspect of an object is its behavior (the things it can do). A behaviour is initiated by sending a message to the object (usually by calling a method).

Polymorphism is the capacity to use an operator or function in multiple ways, or to provide different meanings or functions to the operators or functions.

E.g. the message displayDetails() of the Person class should give different results when send to a Student object (e.g. the enrolment number).

A14. Explain Abstraction with Example

Ans. In simple terms, abstraction "displays" only the relevant attributes of objects and "hides" the unnecessary details. Abstraction is one of the key concepts of object-oriented programming (OOP) languages. Its main goal is to handle complexity by hiding unnecessary details from the user. That enables the user to implement more complex logic on top of the provided abstraction without understanding or even thinking about all the hidden complexity.

Data abstraction refers to, providing only essential information to the outside word and hiding their background details, i.e., to represent the needed information in program without presenting the details. Data abstraction is described as giving just important information to the outer world while hiding background details, i.e., representing the required information in a program without showing the specific details.

For example, a database system hides certain details of how data is stored, created, and maintained. In a similar way, C++ classes provide different methods to the outside world without giving internal detail about those methods and data.

In plain English, abstract means a concept or idea not associated with any specific instance and does not have a concrete existence.

Abstraction Example

```
package com.abstractdemo;
public class MatrixImpl extends Matrix
{
	@Override
	public int[][] add(int[][] a, int[][] b)
	{
		// TODO Auto-generated method stub
		int[][] c = new int[a.length][b.length];
		for (int i = 0; i < a.length; i++)
		{
			for (int j = 0; j < b.length; j++)
			{
				c[i][j] = a[i][j] + b[i][j];
			}
		}
		return c;
}
```

Demo.java

```
System.out.println("Enter a values : ");
for (int i = 0; i < n; i++)
{
	for (int j = 0; j < n; j++)
	{
		a[i][j] = scanner.nextInt();
	}
}
System.out.println("Enter b values : ");
for (int i = 0; i < n; i++)
{
	for (int j = 0; j < n; j++)
	{
		b[i][j] = scanner.nextInt();
	}
}

MatrixImpl impl = new MatrixImpl();
c = impl.add(a, b);
System.out.println("addition:::: ");
for (int i = 0; i < c.length; i++)
{
	for (int j = 0; j < c.length; j++)
		{
		System.out.print(" " + c[i][j]);
		}
		System.out.println("\n")
```

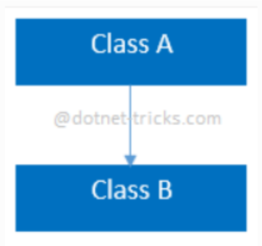
A15. Explain types of inheritance with an example.

Ans. Different Types of Inheritance

OOPs, support the six different types of inheritance as given below :

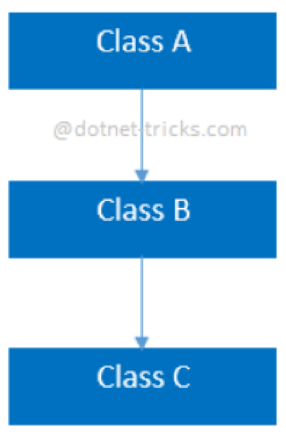
- Single inheritance
- Multi-level inheritance
- Multiple inheritance
- Multipath inheritance
- Hierarchical Inheritance
- Hybrid Inheritance

Single inheritance : In this inheritance, a derived class is created from a single base class. In the given example, Class A is the parent class, and Class B is the child class,
Since Class B inherits the features and behaviour of its parent class A.



Single Inheritance

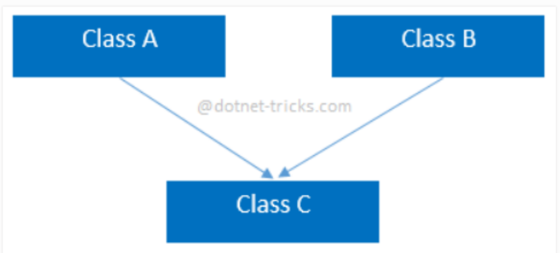
Multi-level inheritance : In this inheritance, a derived class is created from another derived class. In the given example, class c inherits the properties and behavior of class B, and class B inherits the properties and behaviour of class B. So, here, A is the parent class of B and class B is the parent class of C. So, here, class C implicitly inherits the properties and behaviour of class A along with class B, i.e., there is a multilevel of inheritance.



Multi-Level Inheritance

Multiple inheritance : In this inheritance, a derived class is created from more than one base class. This inheritance is not supported by .NET Languages like C#, F#, etc., and Java Language.

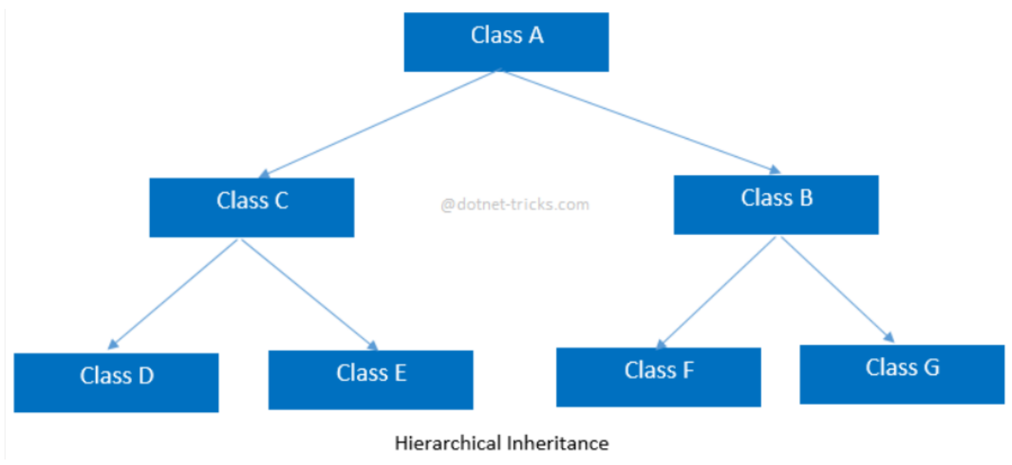
In the given example, class c inherits the properties and behaviour of class B and class A at the same level. So, here, A and Class B both are the parent classes for Class C



Multiple Inheritance

Hierarchical Inheritance : In this inheritance, more than one derived class is created from a single base class and further child classes act as parent classes for more than one child class.

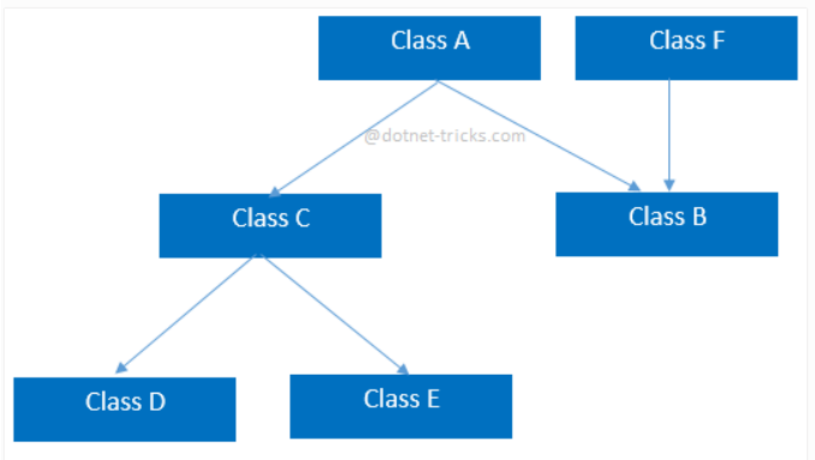
In the given example, class A has two children, class B and class D. Further, class B and class C both are having two children - class D and E; class F and G respectively.



Hierarchical Inheritance

Hybrid inheritance : This is a combination of more than one inheritance. Hence, it may be a combination of multilevel and multiple inheritance or hierarchical and multilevel inheritance. Hierarchical and Multipath inheritance, or Hierarchical, Multilevel, and Multiple inheritances.

Since.NET languages like C#, F#, etc. do not support multiple and multipath inheritance. Hence, hybrid inheritance with a combination of multiple or multipath inheritances is not supported by .NET Languages



Hybrid Inheritance – (a combination of Hierarchical and multiple)

A16. What is Inner Join?

Ans. Inner joins use a comparison operator to match rows from two tables based on the values of common columns from each table. For example, getting all rows from the courses and students' tables when the student identification number is the same

The basic syntax of INNER JOIN is as follows:

```
SELECT table1.column1, table2.column2...FROM table1INNER JOIN table2ON table1.common_filed = table2.common_field;
```

A17. Explain left and right join with example

Ans.

LEFT JOIN: The LEFT JOIN, also known as the LEFT OUTER JOIN, provides all entries from the left table as well as those records from the right table that fulfill a condition. Also, the output or result set will contain NULL values for entries that have no matching values in the correct table.

Example

```
SELECT Employee.EmpFname, Employee.EmpLname, Projects.ProjectID,
Pr:ojects.ProjectName

FROM Employee

LEFT JOIN

ON Employee.EmpID = Projects.EmpID ;
```

Employee Table:

EmpID	EmpFname	EmpLname	Age	EmailID	PhoneNo	Address
1	Vardhan	Kumar	22	vardy@abc.com	9876543210	Delhi
2	Himani	Sharma	32	himani@abc.com	9977554422	Mumbai
3	Aayushi	Shreshth	24	aayushi@abc.com	9977555121	Kolkata
4	Hemanth	Sharma	25	hemanth@abc.com	9876545666	Bengaluru
5	Swatee	Kapoor	26	swatee@abc.com	9544567777	Hyderabad

Project Table:

ProjectID	EmpID	ClientID	ProjectName	ProjectStartDate
111	1	3	Project1	2019-04-21
222	2	1	Project2	2019-02-12
333	3	5	Project3	2019-01-10
444	3	2	Project4	2019-04-16
555	5	4	Project5	2019-05-23
666	9	1	Project6	2019-01-12
777	7	2	Project7	2019-07-25
888	8	3	Project8	2019-08-20

Output:

EmpFname	EmpLname	ProjectID	ProjectName
Vardhan	Kumar	111	Project1
Himani	Sharma	222	Project2
Aayushi	Shreshth	333	Project3
Aayushi	Shreshth	444	Project4
Swatee	Kapoor	555	Project5
Hemanth	Sharma	NULL	NULL

Right Join : The RIGHT JOIN, also known as the RIGHT OUTER JOIN, provides all of the records from the right table as well as any records from the left table that meet the condition. Additionally, the output or result-set will include NULL values for any records for which there are no matching values in the left table.
Example:

```
SELECT Employee.EmpFname, Employee.EmpLname, Projects.ProjectID, Projects.ProjectName
FROM Employee
RIGHT JOIN
ON Employee.EmpID = Projects.EmpID;
```

Output:

EmpFname	EmpLname	ProjectID	ProjectName
Vardhan	Kumar	111	Project1
Himani	Sharma	222	Project2
Aayushi	Shreshth	333	Project3
Aayushi	Shreshth	444	Project4
Swatee	Kapoor	555	Project5
NULL	NULL	666	Project6
NULL	NULL	777	Project7
NULL	NULL	888	Project8

A18. Draw use case on Online shopping product using COD

Ans.

A19. Draw use case on Online shopping product using payment gateway

Ans.

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

Ans.