

SASE – Module 3 Previous Year Questions

Sem 4 BCA MGU



For Notes download BCA Resources App

1. What are use case scenarios? (2019) [2 Marks]

Use case scenarios, also known as use case examples or use case instances, are specific instances or examples that demonstrate how a system or software is used to achieve a particular goal or objective.

Use case scenarios are used in the analysis and design phase of software development to understand and document the interactions between users and the system.



2. What do you mean by scope in SRS? (2019) [2 Marks]

Software Requirements Specification (SRS), scope refers to the boundaries and extent of the software project. It defines what the software will and will not do, including the features, functionalities, and constraints that are included in the project. The scope is a crucial component of the SRS as it sets clear expectations for the stakeholders, including the development team, clients, and end-users.


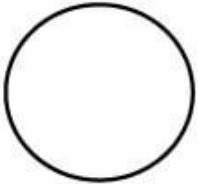
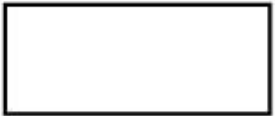
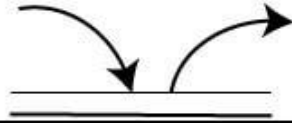
Eg: Functional Requirements, Non - Functional Requirements, Constraints



3. Explain DFD with an example?(2019) [15 Marks]

A Data Flow Diagram (DFD) is a graphical representation of how data flows through a system, showing the processes, data stores, and data flows involved. It is used to understand and model the flow of information within a system or a process.

Eg: Simple online shopping system to explain DFD.

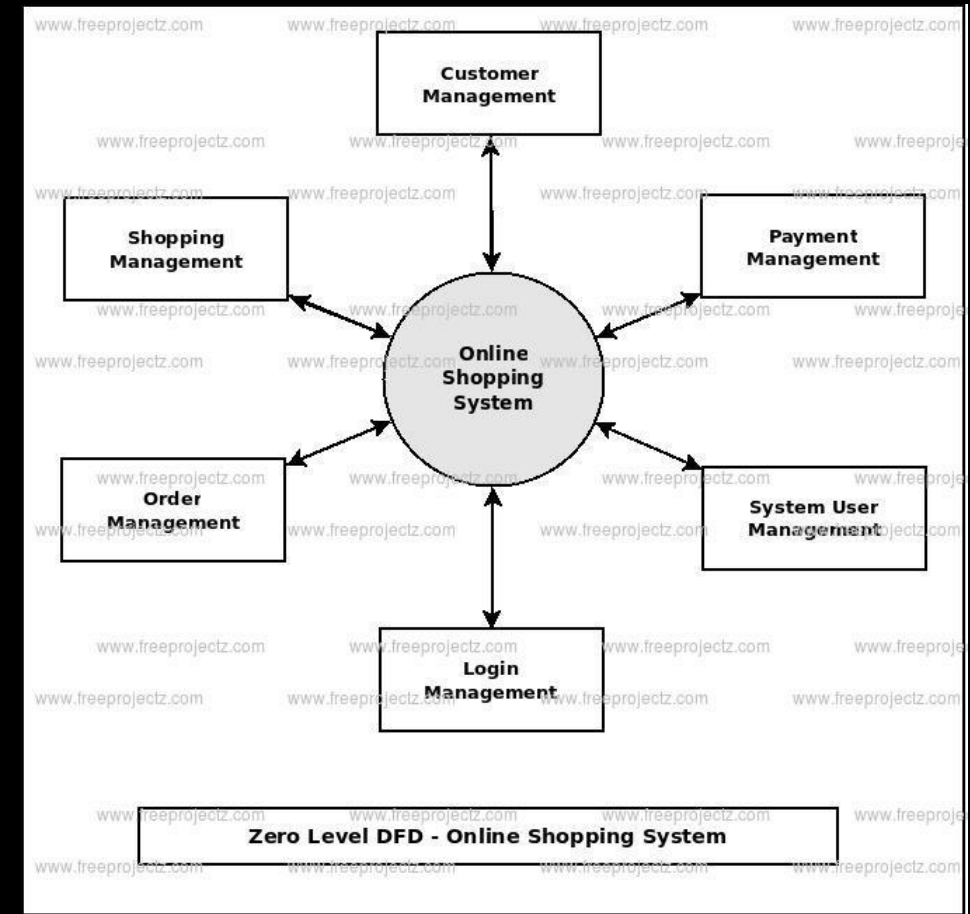
Symbol	Name	Function
	Data flow	Used to Connect Processes to each other, to sources or Sinks; the arrow head indicates direction of data flow.
	Process	Performs Some transformation of Input data to yield output data.
	Source of Sink (External Entity)	A Source of System inputs or Sink of System outputs.
	Data Store	A repository of data; the arrow heads indicate net inputs and net outputs to store.

Symbols for Data Flow Diagrams

3. Explain DFD with an example?(2019) [15 Marks]

Level 0 DFD (Context Diagram):

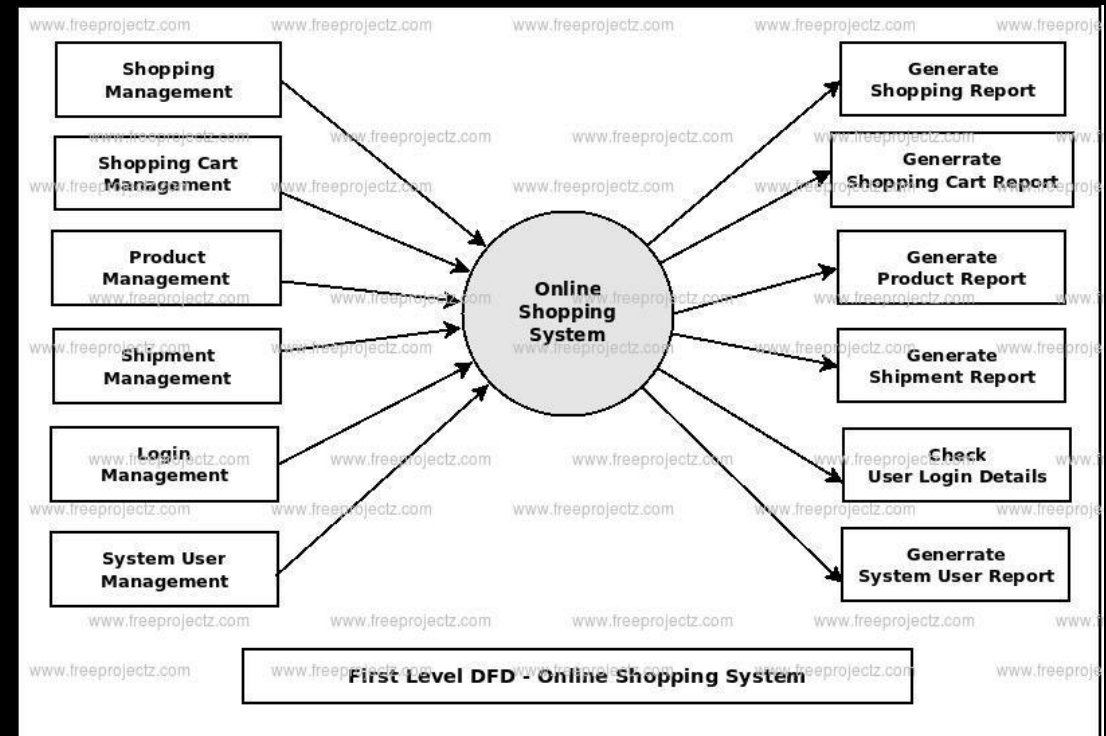
At the highest level, the context diagram represents the entire system as a single process (often referred to as "Level 0").



3. Explain DFD with an example?(2019) [15 Marks]

Level 1 DFD:

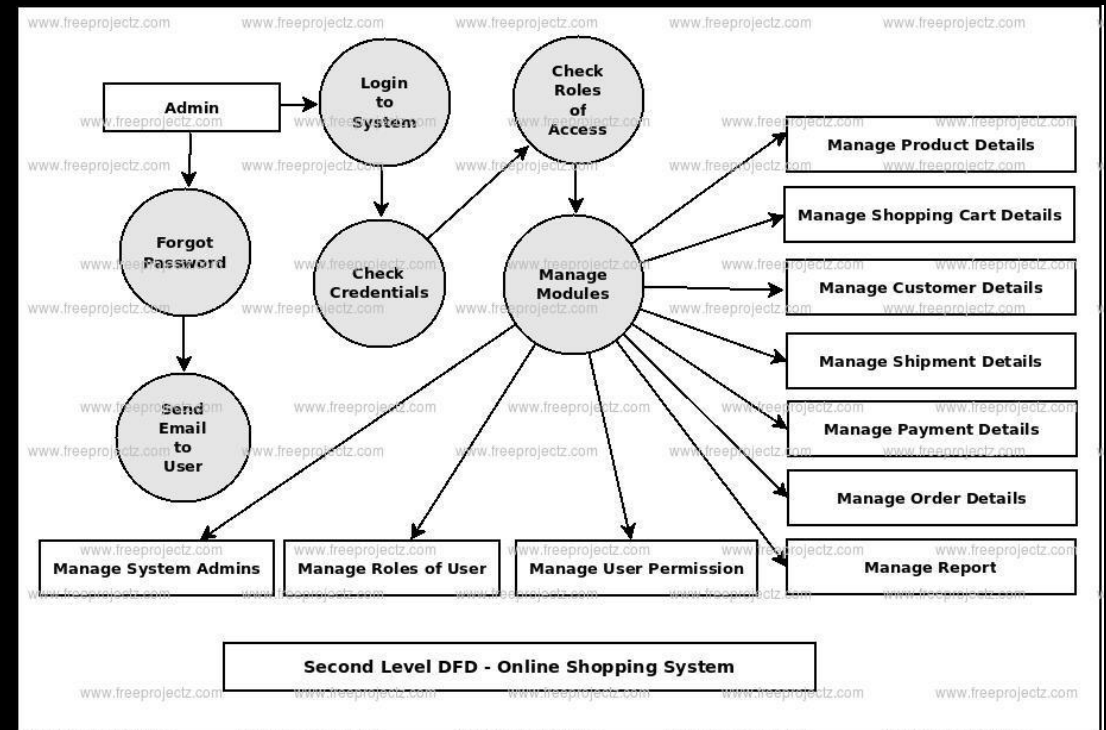
At the next level, we break down the system into sub-processes, which represent specific functions within the system. In this case, Login Management, System User Management etc.



3. Explain DFD with an example?(2019) [15 Marks]

Level 2 DFD (Detailed Level):

At the detailed level, we further break down each sub-process into more detailed sub-processes or individual data flows.



4. List four types of Diagrams?(2019) [2 Marks]

- Data Flow Diagram (DFD): A graphical representation that illustrates how data flows through a system, showing the processes, data stores, and data flows involved.
- Use Case Diagram: A diagram that depicts the interactions between users (actors) and a system, representing the different use cases (functionalities) of the system.
- Entity-Relationship Diagram (ERD): A visual representation that models the logical structure of a database, showing the relationships between different entities (tables) and their attributes.
- Sequence Diagram: A diagram that depicts the interactions between objects or components in a system, showing the sequence of messages exchanged over time. It is commonly used in modeling the dynamic behavior of a system during a specific scenario or process.



5. What is intermediate COCOMO?(2019) [5 Marks]

- Intermediate COCOMO (Constructive Cost Model) is a software cost estimation model, an extension of the basic COCOMO model. It is used to estimate the effort, time, and cost required to develop a software project based on the size of the project and various other factors. The Intermediate COCOMO model considers three different modes of software development: Organic, Semi-detached, and Embedded.

The formula for Intermediate COCOMO is as follows:

- $\text{Effort} = a * (\text{KLOC})^b * \text{EAF}$

where,

- Effort: Effort required to develop the software project in Person-Months (PM).
- KLOC: Estimated size of the software project in thousands of lines of code.
- EAF: Effort Adjustment Factor, a multiplier that considers various project-specific factors such as complexity, development tools, personnel, etc.
- a, b: Constants that depend on the development mode (Organic, Semi-detached, or Embedded).



Intermediate COCOMO categorizes software projects based on their characteristics and requirements into the following development modes:

1. Organic Mode:

- Small to medium-sized projects.
- Well-understood requirements.
- Experienced and skilled development team.
- Low complexity.

2. Semi-detached Mode:

- Projects with some unique characteristics or moderate complexity.
- Some parts of the project are well-understood, but others are not.
- Moderate development team experience.

3. Embedded Mode:

- Large, complex, and critical projects.
- High reliance on hardware, software, and external interfaces.
- Tight project constraints and strict requirements.
- High development team experience and skills.

Each development mode has its own set of constants (a and b) that are used in the estimation formula to calculate the effort required for the software project. The Effort Adjustment Factor (EAF) is used to adjust the effort based on various project-specific factors, such as development tools, schedule, personnel experience, and risk.

Intermediate COCOMO provides a more accurate estimation of effort and cost compared to the basic COCOMO model by considering additional factors and different development modes. However, it still relies on historical data and assumptions, so it should be used with caution and as a part of a comprehensive cost estimation process.

6. Explain Difference between flow chart & structure Chart?(2019) [5 Marks]

	Flow Chart	Structure Chart
Purpose	used to represent the flow of control or sequence of steps in a process or algorithm. It helps to visualize the logical steps in a clear and sequential manner, making it easier to understand the overall flow of the process.	used to represent the hierarchical structure and organization of a software program. It shows the different modules or components of the program and their relationships, depicting how they are interconnected.
Representation	the various steps or actions in a process are represented using different shapes such as rectangles for process steps, diamonds for decision points, and arrows to show the flow of control.	the different modules or components of the software program are represented as boxes, and their relationships are shown using lines connecting the boxes.
Level of Detail	provide a high-level view of the process or algorithm, focusing on the sequence of steps and decision points. They are more suitable for representing the overall logic of a process.	provide a more detailed view of the software program's organization. They show the modular structure of the program and how each module interacts with others.
Application	Flowcharts are commonly used in business process modeling, algorithm design, and representing the logical flow in a program.	Structure charts are primarily used in software engineering for designing and documenting the software architecture, depicting the module structure and their dependencies.
Complexity	Flowcharts are suitable for simple to moderately complex processes or algorithms.	Structure charts are more suitable for complex software programs with multiple modules and interactions between them.



7. Explain various cost estimation techniques?(2020) [15 Marks]

Static, Single Variable Models: When a model makes use of single variables to calculate desired values such as cost, time, efforts, etc. is said to be a single variable model. The most common equation is:

$$C = a L^b$$

Where C = Costs

L= size

a and b are constants

Static, Multivariable Models: These models are based on equation $C = a L^b$, they depend on several variables describing various aspects of the software development environment, for example, methods used, user participation, customer oriented changes, memory constraints, etc.



7. Explain various cost estimation techniques?(2020) [15 Marks]

COCOMO (Constructive Cost Model) is a software cost estimation model developed by Barry W. Boehm in the late 1970s. It is used to estimate the effort, time, and cost required to develop a software project based on the size of the project and various other factors. COCOMO is widely used in the software industry for project planning, budgeting, and resource allocation.

There are three levels of the COCOMO model:

Basic COCOMO:

The basic COCOMO model is the initial version and is suitable for estimating the effort and cost of small to medium-sized software projects. It is based on the following formula:

$$\text{Effort} = a * (\text{KLOC})^b$$

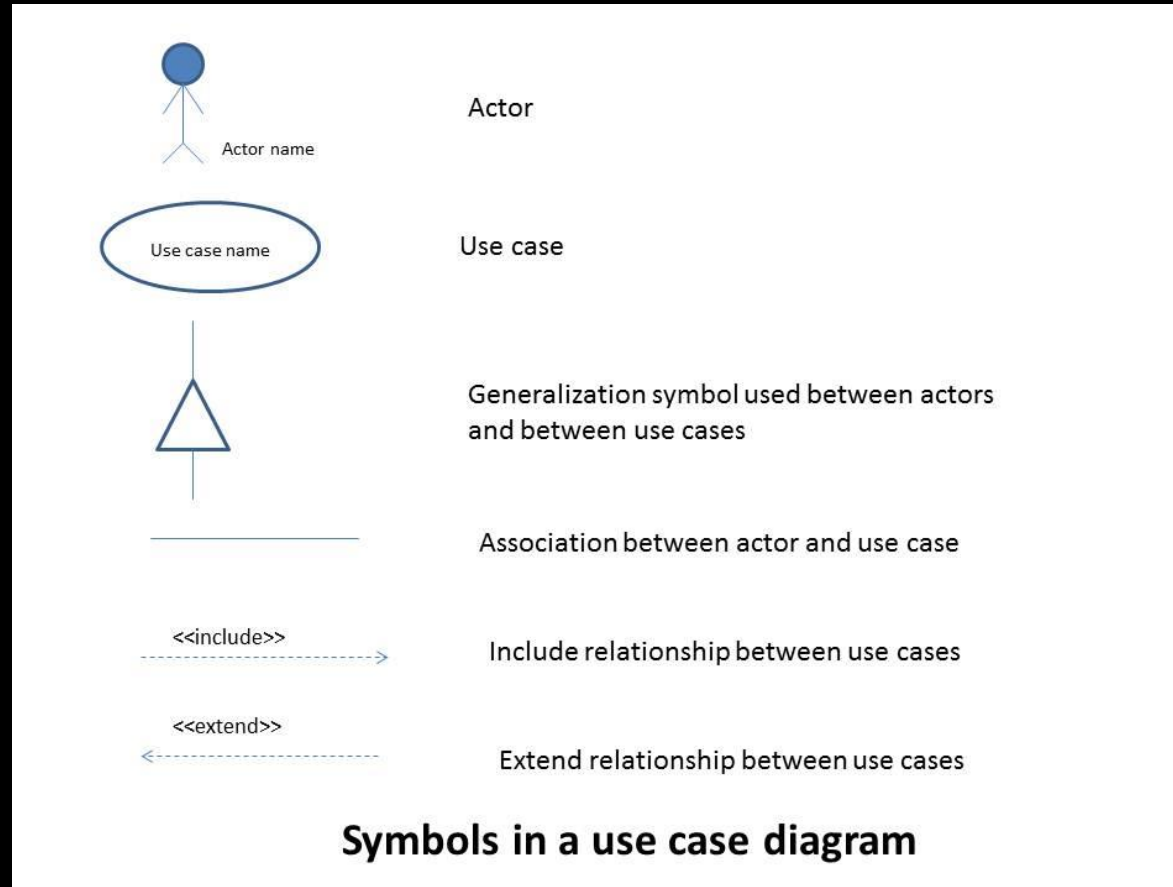
Intermediate COCOMO: Refer Answer 5

Detailed COCOMO:

The detailed COCOMO model is the most comprehensive version and is suitable for estimating the effort, cost, and schedule for complex and critical software projects. It considers a wide range of factors, including personnel capability, risk management, and project complexity.



8. Explain different symbols used in use case diagrams?(2020) [5 Marks]



9. What is technical feasibility?(2020) [2 Marks]

Technical feasibility is an assessment of whether a proposed software project can be developed and implemented using existing or readily available technology, resources, and expertise. It involves evaluating the technical aspects of the project to determine if it is practical and achievable from a technical standpoint. The technical feasibility analysis is an essential part of the project planning and decision-making process and is usually conducted during the early stages of project initiation.



10. Why requirements are difficult to uncover?(2021) [5 Marks]

Uncovering requirements for a software project can be challenging due to several factors. These factors contribute to the complexity of the requirements gathering process and make it difficult to obtain a complete and accurate set of requirements. Some of the main reasons why requirements are difficult to uncover are:

- **Lack of Clarity:** Stakeholders may have a vague understanding of their needs or may struggle to articulate their requirements clearly, leading to ambiguity and misinterpretation.
- **Changing Requirements:** Requirements can change throughout the project lifecycle due to evolving business needs, market changes, or user feedback, making it challenging to capture and manage changing requirements effectively.
- **Implicit and Tacit Knowledge:** Some requirements may be assumed or taken for granted, making them challenging to elicit and document.
- **Hidden Assumptions:** Stakeholders may have assumptions about the system's behavior or functionalities that they do not explicitly communicate, leading to missing or misunderstood requirements.
- **Stakeholder Availability:** In large organizations or projects with multiple stakeholders, it may be challenging to gather all relevant stakeholders' input due to conflicting schedules or other commitments.
- **Incomplete Information:** Stakeholders may not have a comprehensive understanding of their needs or may lack awareness of all the potential functionalities that the system can offer.
- **Communication Issues:** Miscommunication or language barriers between the development team and stakeholders can lead to misunderstood requirements or misinterpretation of expectations.
- **Technical Constraints:** The limitations of existing technology or budget constraints may affect the feasibility of certain requirements, leading to the need for trade-offs and compromises.

11. Explain various steps for requirement analysis?(2021) [15 Marks]

- **Elicitation:** The first step is to gather requirements from stakeholders, including end-users, clients, domain experts, and other relevant parties. Various techniques can be used for elicitation, such as interviews, surveys, workshops, and brainstorming sessions. The goal is to understand the needs, expectations, and objectives of the software project.
- **Requirement Gathering:** During this step, all the gathered information is organized and classified into functional and non-functional requirements. Functional requirements define what the system should do, while non-functional requirements specify how well it should perform in terms of reliability, performance, security, and other quality attributes.
- **Documentation:** The requirements are documented in a clear and concise manner using various artifacts such as use cases, user stories, requirement specifications, and prototypes. Proper documentation ensures that all stakeholders have a common understanding of the project's objectives and functionalities.
- **Analysis and Prioritization:** The gathered requirements are analyzed to identify any inconsistencies, conflicts, or missing information. The development team and stakeholders prioritize the requirements based on their importance, business value, and feasibility. This helps in making informed decisions about the project scope and resource allocation.
- **Validation and Verification:** The requirements are validated to ensure that they meet stakeholders' needs and align with the project's objectives. The development team and stakeholders collaborate to review and verify the requirements for correctness, completeness, and feasibility.
- **Requirement Negotiation:** In cases where there are conflicting or competing requirements, requirement negotiation is conducted to reach a consensus and resolve any conflicts. This may involve trade-offs and compromises to satisfy all stakeholders' needs.
- **Requirements Traceability:** Traceability is established between the requirements and other project artifacts, such as design documents, test cases, and code. This ensures that each requirement is addressed during the development process and helps in managing changes and impact analysis.
- **Change Management:** Requirements are likely to change during the project's life cycle due to evolving business needs or other factors. A proper change management process should be in place to handle requirement changes and ensure they are implemented and communicated effectively.
- **Review and Feedback:** The requirements are continuously reviewed and refined throughout the requirement analysis phase. Feedback from stakeholders is incorporated to improve the quality and accuracy of the requirements.
- **Draw the context Diagram:** The context diagram (level 0 DFD) is a simple model that represent the entire system as a single bubble with input and output data.

11. Explain various steps for requirement analysis?(2021) [15 Marks]

- **Develop prototype (optional):** A prototype helps the client to visualize the proposed system and increases the understanding of requirements. When developers and users are not certain about some of the requirements, a prototype may help both parties to take a final decision. Prototypes are continuously modified based on the feedback from the customer until they are satisfied.
- **Model the requirements:** Different types of models like data flow diagrams, ER diagrams and data dictionaries, state transition diagrams are developed in this step which help for verification.
- **Finalize the Requirements:** The verified requirements are finalized and next step is to document these requirements



12. Purpose of feasibility study?(2022) [5 Marks]

The purpose of a feasibility study is to assess the viability and practicality of a proposed project or initiative before committing significant resources to its development. The study aims to determine whether the project is feasible in terms of technical, economic, legal, operational, and scheduling aspects. It helps stakeholders, including decision-makers, project managers, and investors, to make informed decisions about whether to proceed with the project or not. The key purposes of a feasibility study are as follows:

- **Consider Operational Feasibility:** Operational feasibility assesses whether the project can be smoothly integrated into existing systems and processes without disrupting day-to-day operations.
- **Evaluate Schedule Feasibility:** Schedule feasibility determines whether the project can be completed within the desired time frame, considering various constraints and dependencies.
- **Risk Assessment:** The feasibility study identifies potential risks associated with the project and evaluates their impact on the project's success. It helps in developing risk mitigation strategies.
- **Provide a Basis for Decision-Making:** The findings of the feasibility study serve as the basis for decision-making regarding project initiation, resource allocation, and project planning.
- **Identify Alternatives:** The study explores alternative solutions or approaches to achieve the project's objectives. It helps in comparing different options and selecting the most suitable and feasible one.



13. Requirement elicitation? Explain the types (2022) [15 Marks]

Requirement elicitation is the process of gathering and capturing the needs and expectations of stakeholders for a software project. It involves interacting with stakeholders to understand their requirements, preferences, and constraints, which form the foundation for defining the scope and objectives of the project. Effective requirement elicitation is crucial for the success of a software project, as it lays the groundwork for further analysis, design, and development.

- Interviews
- Brainstorming Sessions
- Facilitated Application Specification Technique (FAST)
- Quality Function Deployment (QFD)
- Use Case Approach



13. Requirement elicitation? Explain the types (2022) [15 Marks]

- Interviews: Interviews involve one-on-one or group discussions with stakeholders, users, or subject matter experts to gather detailed information about their needs, expectations, and concerns regarding the software project. The interviewer asks open-ended questions to understand their requirements better.
- Brainstorming Sessions: Brainstorming sessions involve bringing together stakeholders, users, and development team members in a collaborative environment to generate ideas, requirements, and potential solutions. It encourages creativity and idea sharing to uncover different perspectives and possibilities.
- Facilitated Application Specification Technique (FAST) : FAST is a facilitated workshop approach that involves key stakeholders, end-users, and development team members working together to define and prioritize requirements. It uses structured brainstorming and decision-making techniques to reach a consensus on requirements.
- Quality Function Deployment (QFD):QFD is a technique used to translate customer needs into specific technical requirements. It involves capturing customer requirements and converting them into design and engineering characteristics, ensuring that the final product meets customer expectations.
- Use Case Approach: The Use Case approach involves identifying and documenting specific use cases that represent different interactions between users and the software system. Use cases describe how users interact with the system and the system's responses, helping to define functional requirements.

