

System Analysis & Design

Section – A

► System:

- A system is a group of elements or components which work together to accomplish a common task.
Example: A hospital is a system – doctors, nurses, labs, patients, equipment – all work together to treat patients.
-

► System Analysis & Design?

- In very simple words, system analysis and design is a study in which we learn how to analyze an existing system and create a better one.
-

► Why we need it?

- For system development
 - Creating a new one
 - Updating the existing one
-

System Analysis & Design

1. System Analysis:

- System Analysis is a process of studying and observing a system to know how it works and to identify its goals and purposes.
 - System analysis is conducted for the purpose of studying a system or its parts in order to identify its objectives.
 - It specifies “**what the system should do.**”
-

2. System Design:

- It is a process of planning a new system or replacing an existing system.
 - It is done by defining its components or modules to satisfy the specific requirements.
 - It focuses on “**how to accomplish the objective of the system.**”
-

► Characteristics:

A system has several important characteristics. These help us understand how it functions, why it is organized, and how each part contributes to the goal.

1. Organization:

- Organization means the system is arranged in a structured and orderly way. All components are arranged logically to function efficiently.
- No system can work in a random manner. Each component has a defined position and role in the overall structure.

Example: In a university system, students, faculty, administration, library, and examination sections are all organized into departments for smooth operation.

2. Interaction:

- Interaction refers to the way components of a system communicate and cooperate with each other to perform tasks.
- No component works in isolation. They exchange information, data, or energy to support the functioning of the system.

Example: In a human body, the brain sends signals to the heart and lungs to function. This is interaction between body parts.

3. Interdependence:

- Components of a system are dependent on each other. If one part fails, it may affect the entire system.
- This ensures that each part is important and the failure of any one component can break the system flow.

Example: In a manufacturing plant, if the packaging unit fails, the production has to stop even though the goods are produced.

4. Integration:

- Integration is when all components work together as a unified system to achieve a goal.

- Integration ensures that all parts are coordinated and the system behaves as a single functional unit, not isolated parts.

Example: In a mobile phone, if the screen, processor, battery, and software don't work together, the phone is useless.

5. Central Objective:

- Every system must have a common and defined goal or objective.
- Without a goal, the system has no purpose. All efforts of the components should be directed toward achieving that central aim.

Example: In a transport system, the goal is to move people and goods safely and efficiently from one place to another.

6. Input:

- A system takes input, processes it, and gives output.
- Inputs can be data, energy, materials etc. Outputs are the results of system processing.

Example:

In a school system:

- Input = Students
 - Processing = Teaching, exams
 - Output = Educated students with report cards
-

► Elements of a System:

Each system is made up of core elements that define its working. These elements help us analyze and design system efficiently.

1. Input:

- Input refers to resources, data, energy that are entered into the system for processing.
- A system cannot operate without input. Input starts the functioning of the system.

Example:

In a library management system, inputs are new book entries, student data, and borrowing requests.

2. Process:

- Processing is the transformation of input into output using defined methods.
- This is the core operation where the actual work is done inside the system.

Example:

In a bank, the process involves:

- Checking account balance
 - Deducting or adding money
 - Generating a transaction report
-

3. Output:

- Output is the final result or product produced by the system after processing.
- A system's performance is measured by the quality and accuracy of its output.

Example:

In a payroll system, the output is employees' salary slips and payment transfers.

4. Feedback:

- Feedback is the information about the output that is returned to the system to make corrections if needed.
- Feedback helps in improving performance and ensuring the system is meeting its goal.

Example:

In online learning, feedback from students helps teachers update content or teaching methods.

5. Control:

- Control ensures that the system's processes are within acceptable limits and not deviating from the goal.
- It involves rules, standards, and checks to guide the system.

Example:

In a traffic control system, sensors and signals ensure vehicles follow rules and avoid accidents.

6. Environment:

- The environment consists of all external factors that affect the system but are not part of it.
- A system should be designed keeping in mind its environment because it can affect its working.

Example:

For a restaurant, environment includes customer behavior, weather, competitors etc.

► Types of Systems:

1. Physical Systems:

- These are tangible and real systems that can be seen and touched.

Examples:

- Human body
 - Railway system
 - Machinery
-

2. Abstract Systems:

- These are intangible, based on ideas or logic. You cannot touch them.

Examples:

- Software design
 - Algorithms
 - Mathematical models
-

3. Open Systems:

- Systems that interact with their external environment and adapt accordingly.

Example:

A business organization interacts with customers, suppliers, and the market.

4. Closed Systems:

- Systems that are isolated from their environment and do not interact externally (mostly theoretical).

Example:

A chemical reaction in a sealed container.

5. Deterministic Systems:

- These systems give predictable and fixed output for a given input.

Example:

A calculator – $2 + 2$ always equals 4.

6. Probabilistic Systems:

- These systems produce output with some degree of uncertainty or chance.

Examples:

- Stock market
 - Weather forecasting
 - Exam performance
-

7. Manual Systems:

- All processes are carried out by humans without machines or automation.

Examples:

- Manual attendance register
 - Handwritten bills
-

8. Automated Systems:

- Systems that work using machines, software, or technology, with little or no human effort.

Examples:

- ATM machines
- Online ticket booking

- Robot operated factory
-

► The Role of the System Analyst:

System Analyst:

- A system analyst is a person who uses system analysis and design techniques to solve system/business problems.
 - A system analyst analyzes, designs and implements system to fulfill organizational needs.
 - System analyst conducts system study, identifies requirements & determines the procedures to achieve system objectives.
 - System analyst designs and implements the system to suit organizational requirements for effective results.
 - System analysts carry the responsibilities of researching problems, finding solutions, recommending courses of actions and coordinating with other members in order to meet specified requirements.
 - In the IT industry, system analyst figures out how to solve a problem by linking different computers and by specifying what platforms, protocols, software, hardware and communications medium can be used to solve a problem.
 - System Analyst is responsible for the system from its birth to death.
-

Example:

Suppose a university wants a student result portal. The system analyst:

- Talks to teachers and students.
 - Understands what data is needed (marks, subject, attendance).
 - Designs the flow of result calculation.
 - Guides developers to build it.
 - Tests the portal before launch.
-

► Qualifications of a System Analyst:

To perform such critical responsibilities, a system analyst needs to have multiple qualifications across education, skills, and experience.

1. Education Qualifications:

Qualification Level — Description

- **Bachelor's Degree** – In Computer Science, IT, BCA, BSC (CS), or B.Tech (IT)
- **Master Degree** – MCA, M.Sc (IT), or MBA (for business understanding)

Some system analysts come from a dual background: Technical + Management.

2. Technical Skills Required:

Skill — Use Case

- Programming basic – To understand what developers can or cannot build.
 - Database Management (SQL) – For data design and queries.
 - Software Development Life Cycle – To plan and manage development stages.
 - **Modelling Tools & methods (DFD, ERD, UML)**
→ For system design and documentation
 - **Testing tools & methods**
→ To ensure quality & bug free delivery
 - **Networking fundamentals**
→ For systems involving servers, cloud and devices
-

3. Analytical & Problem Solving Skills:

A system analyst should:

- (i) Break down complex problems.
- (ii) Understand workflows.
- (iii) Suggest optimized, logical solutions.

Example:

Redesigning a railway ticket booking system to reduce server crashes and long queues during festival times.

4. Communication & Soft Skills:

Skill — Importance

- **Clear Communication**
→ To explain system needs to non technical users and vice versa.
- **Active Listening**
→ To understand what users really want.

- **Team coordination**
→ To work with coders, testers, management, and clients.
 - **Report writing**
→ To document all system specs, diagrams, and manuals.
-

► Major Roles and Responsibilities of a System Analyst:

1. Requirement Collector & Problem Investigator:

What they do:

- Meet & talk with users, observe current workflows, interview stakeholders.
- Identify existing system limitations, gaps or inefficiencies.

Why Important:

- If the problem is not well understood, the solution will fail.

Example:

In a hospital, nurses complain that they waste time writing medicine records manually. Analyst observes this and proposes a digital patient charting system.

2. Feasibility Study Expert:

What they do:

- Evaluate whether a proposed system is worth building.
- Perform:
 - (i) Technical feasibility – Do we have the tech?
 - (ii) Economic feasibility – Is it affordable?
 - (iii) Operational feasibility – Will users accept and use it?

Example:

Before building an AI based attendance system, the analyst checks if all classrooms have cameras and whether the institution has the budget for AI software.

3. System Design (Architect):

What they do:

- It converts the user requirements into technical blueprint.
- Prepares:

- (i) Data flow diagrams
- (ii) Entity relationship diagrams
- (iii) Input / Output screen layouts
- (iv) Database schema

Why important:

- Developers follow these diagrams like a construction team follows architectural plans.

Example:

For a library management system, the analyst designs the book borrowing process, overdue fine calculation, and report generation structure.

4. Liaison between users and developers:

- Translates user language (non technical) into developer instructions (technical).
- Ensures that what the users expect is what gets built.

Real example:

In an online learning platform:

- Teachers want a quiz upload with timer.
 - Analyst explains to developers:
“Add multiple choice input form with countdown timer and auto submit feature.”
-

5. Project Planner & Coordinator:

What they do:

- Create detailed plans and timelines for system development.
- Allocate tasks, manage resources, monitor deadlines.

Why important:

- Without planning, the project may go over budget, miss deadlines, or fail.

Example:

While building an e-wallet system, the analyst decides on development phases:

1. Login system
 2. Wallet Top-up
 3. Money transfer
 4. Transaction history
-

6. System Evaluator & Tester:

- Review and test the system before final delivery.
- Test for:
 - (i) Functionality (does it work?)
 - (ii) Performance (is it fast?)
 - (iii) Security (is it safe?)
 - (iv) Usability (is it easy?)

Example:

In an e-commerce app, the analyst tests:

- Product search speed
 - Payment gateway
 - Error handling
 - Mobile friendliness
-

7. Trainer & Documentation Expert:

- Prepare user manuals, guides, help documentation.
- Conduct training sessions for end users.

Why important:

- A system is only useful if people know how to use it correctly.

Example:

After launching new HR payroll software, the analyst creates video tutorials and trains HR staff on how to generate payslips.

8. Maintenance & Support Handler:

- Monitor system performance after deployment.
- Handle user complaints, fix bugs, apply updates.

Example:

In an online ticket booking system, users face login errors during peak hours. The analyst finds the issue in the session handling code and gets it fixed.

Software Development Life Cycle

Section – A

SDLC stands for “Software Development Cycle”.

SDLC is a collection of processes which are followed to develop a software.

SDLC is a methodology that defines some processes which are followed to develop a high quality software.

Flow of SDLC:

Requirement Analysis → Feasibility Study → Design → Coding → Testing → Deployment
→ Maintenance

- It covers a detailed plan for building, deploying and maintaining the software.
 - The main aim of SDLC is to define all the tasks required for developing and maintaining software.
 - It is followed for a software project within a software developing organization.
-

► Phases of SDLC:

Phase 1: Requirement Analysis

- It is the first phase of SDLC in which all the necessary information is collected from the customer to develop the software as per their expectations.

- Some important questions like: what is the need of software, who will be the end-user, what is the future scope of that software etc. are discussed.
 - The main aim of this phase is to collect the details of each requirement of the customer so that the developers will clearly understand what they are developing and how to fulfill the customer's requirements.
 - This phase gives a clear picture of what we are going to build.
-

Phase 2: Feasibility Study

- It is the second phase of software development life cycle in which an organisation discusses about the cost and benefits of the software.
 - It is an important phase because profit from the software plays an important role as if cost is very high then company may face loss.
 - After the feasibility study, the project may be accepted, accepted with modification or rejected.
 - It measures how much beneficial the product is for the organisation.
-

Phase 3: Design

- It is the third phase in which architects starts working on logical designing of the software.
 - In this phase SRS (System Requirement Specification) document is created which contains all logical details like how software will look like, which language will be used, database design, modular designs etc.
 - This phase provides a prototype of the final product.
 - Basically all it includes is design of everything which has to be coded.
-

Phase 4: Coding

- When the designing of the software is completed, then a group of developers starts coding of the design using a programming language.
 - The interface of the software and all its internal working according to design phase is implemented in coding phase.
 - A number of developers code the modules and then all modules are arranged together to work efficiently.
 - It is the longest phase of Software development life cycle.
-

Phase 5: Testing

- Once the software development is completed, then it is sent to the testers. The testing team starts testing the functionality of the entire system.
 - In this phase, the software is checked for bugs or errors.
 - Whenever a bug is found, then the software is resent to the coders to fix it and then overall software is re-tested.
 - This is done to verify that the entire application works accordingly to the customer requirement.
-

Phase 6: Deployment

- After overall testing of the software and after checking that it is bug free, then the software is launched and available for the users to use it.
 - Even after deployment of the software, if any bugs or errors are still found then the software is re evaluated by the maintenance team and then it is redeployed with a new version.
-

Phase 7: Maintenance

- The maintenance team look over the software usage and user's feedbacks.
 - Maintenance is necessary to eliminate errors in the system during its working life and to tune the software.
 - The bug fixing, upgrade and enhancement of the software is looked over by the maintenance team.
-

System Analysis: Initial Investigation, Needs Identification, Determining the user's information requirements, information gathering tools

1. Initial Investigation:

- The first step in the analysis phase of system development.
- A fact-finding mission to identify:
 - (i) The current problem in the system.
 - (ii) What the user needs from a computer based system.

Purpose:

- To build a full picture of the problem.
- To decide whether the project is feasible (technical, operational, economic feasibility).
- To act as a foundation for later stages (analysis & design).

Steps / Areas to Investigate:

1. Background to the problem – identify and describe clearly.
 2. Identify the user(s) who are facing the problem.
 3. Identify what the users need to solve the problem.
 4. Outline limitations of the proposed solutions.
 5. Suggest two or more alternative solutions.
 6. Choose the best solution and justify.
 7. Identify sources & destinations of data used by the system.
 8. Identify all system data requirements and (if needed) draw an ER model.
 9. Provide a clear list of objectives → what the project must achieve to be successful.
-

2. Need Identification:

Meaning:

- First step in the system development life cycle.
- Involves identifying a need to change, improve or enhance an existing system.

Nature of requests:

- Likely to be a stream of user requests.
- Must be handled with standard procedures.
- May be for:
 - (i) A new system
 - (ii) An improvement in the existing system
 - (iii) Or just minor modifications

Purpose:

- To check whether the request is valid and feasible.
- Helps management decide whether to:
 - (i) Do nothing
 - (ii) Improve existing system
 - (iii) Replace existing system

User's request form includes:

1. User-assigned title of work requested.
2. Nature of work requested (new / revision of existing).

3. Date request submitted.
 4. Date job to be completed.
 5. Job objectives (purpose of request).
 6. Expected benefits from proposed changes.
 7. Input / Output description (quantity, frequency, type).
 8. Requester's details (signature, title, dept, phone).
 9. Approver's details (signature, title, dept, phone).
- This form documents the user's needs and helps management decide whether to approve the request.

Importance:

- Success of a system depends on how accurately the problem is defined and investigated.
 - Analyst must focus on real problem.
 - Needs identification ensures that the system is aligned with user wants & business goals.
-

3. Determining the user's information requirements:

Meaning:

- Process of finding out what information users need, in what form and for what purpose.
- Critical because information is the heart of a computer based system.

Purpose:

- To determine:
 - (i) What information is currently available.
 - (ii) What new information is required.
- To reach an agreement between user & analyst about what the new system will provide.

Difficulties in determining user's requirements:

1. Changing requirements – users keep modifying needs.
 2. Hard to identify needs – only experienced users can explain clearly.
 3. Low user involvement – users may not give enough time or input.
 4. Complex user analyst interaction – communication gap may arise.
-

Strategies users adopt (while giving requirements):

1. Kitchen Sink strategy:

- Users throws everything into requirements.
- Overstatements of needs → unnecessary reports, excessive demand.
- Shows lack of experience.

2. Smoking strategy:

- Users demands more features than needed to gain bargaining power.
- Analyst must filter out extra requests.
- Reflects user's experience in negotiation.

3. Same Thing strategy:

- Users asks for the same system, just improved.
 - Example: Give me the same thing in a better format.
 - Shows lack of clarity.
-

► Information Gathering Tools:

What is information gathering?

- Information gathering is an art and science of collecting information regarding the present system.
 - It is done by system analyst so that the existing system can be upgraded easily without any errors and fulfills users requirements.
 - It also helps the system analyst to prepare a precise SRS document.
-

Information Gathering Tools / Techniques:

- It is not simple and easier for system analyst to collect information regarding the system that's why information gathering tools / techniques are used.
- These tools / techniques are:

- (i) Document observation
 - (ii) On-site observation
 - (iii) Interviewing
 - (iv) Questionnaires
-

(i) Document observation:

- This term is also called “Review of Records, Procedures, and Forms”.
 - Document observation refers to seek existing records of the system which helps to understand the system thoroughly, i.e. it will describe the current system functionalities and capabilities.
 - Within a short period of time, the procedure manuals and forms describe the format and functions of present system.
 - It can provide a clear understanding about the operations that are handled in the organization, identifying input for processing and evaluating performance.
-

(ii) On site observation:

- On site observation refers to the visit of system analyst in the organization in order to observe the working of the current system.
 - In this method, information gathering is done by noticing and observing the people, events, and objects in the organization.
 - It is a direct method for gathering information.
 - It is useful in situation when the data collected is in question or prevents clear explanation by end users.
-

(iii) Interviewing:

- Interviewing refers to the face to face communication between systems analyst and users to gather information about the problem.
- The interview can be formal or informal, as the success of an interview depends on the skill of analyst as an interviewer.
- It is the oldest and most often used tool to gather information because it is the best source for gathering qualitative information.

It can handle the complex subjects and bridges the gaps in the areas of misunderstandings and minimizes future problems.

Types of interviews

Unstructured interviews

Structured interviews

Unstructured interviews:

- In this type of interviews, question-answer session are conducted to acquire basic information of the system.

Structured interviews:

- This type of interviews has standard questions which user needs to respond in either close (objective) or open (descriptive) format.

4. Questionnaires:

- Questionnaires refers to the gathering of information from a large audience in the form of survey.
 - This method is used by analyst to gather information about various issues of system from large number of people because interview is not possible when the target audience is large.
 - It is useful in situation to know what proportion of a given group approves or disapproves of a particular feature of the proposed system.
 - It is very effective in surveying interests, attitudes, feelings, and beliefs of users.
-

Types

(i) Open-ended:

- Questionnaires that allows the target audience to voice their feelings and emotions freely are called open-ended questionnaires.
- These questions are not based upon pre-determined responses.

(ii) Closed-ended:

- Questionnaires that have multiple options as answer and allow respondents to select a single option from among them are called closed-ended questionnaires.
-

Structured Analysis Tools

Section – B

- Structured analysis is a technique that uses graphical diagrams to develop and portray system specifications that are easily understood by users.
- Structured analysis is a development method that allows the analyst to understand the system and its activities in a logical way.
- It is a systematic approach which uses graphical tools that analyze and refine the objectives of an existing system and develop a new system specification which can be easily understood by users.
- The basic goal of SA/SD is to improve the quality and reduce the risk of system failure.
- The tools and techniques used during structured analysis phase:
 - (i) DFD (Data flow Diagram)
 - (ii) Data Dictionary
 - (iii) Structured English

- (iv) Decision tree
 - (v) Decision table
 - Structured design is a concept of decomposing problem into several well organized elements of solution.
 - It gives better understanding of how the problem is being solved.
 - Structured design also makes it simpler for designer to concentrate on problem more accurately.
 - Structured design is mostly based on “divide and conquer” strategy, where a problem is broken into smaller problems and each small problem is individually solved (several) until the whole problem is solved.
 - The small pieces of problem are solved by means of solution modules. Structured design emphasis that these modules be well organised in order to achieve precise solution.
-

DFD :-

- DFD stands for Data flow diagram and also known as bubble chart.
 - DFD show the flow of data between various elements of a system in graphical form.
 - It also express the requirement of the system and shows how the current system is implemented.
 - It gives an overview of what data a system processes, what transformations are being performed, what data are stored, what results are produced and where they flow.
 - Its graphical nature makes it a good communication tool b/w users and system designer (System Analyst).
 - DFD is easy to understand and quite effective tool for communication (between user and system analyst) when the required design is not clear.
 - It requires a large number of iterations for obtaining the most accurate and complete solution.
-

Basic elements of DFD / Components of DFD :-

Symbols used in designing DFD are:

- Circle → Process
 - Open rectangle → Data store
 - Rectangle → Entity
 - Arrows → Data flow
-

Types

Logical DFD

- It shows how data flows in a system.
- It describes what data is moved from one entity to another.
- It focuses on what happens in the data flows.
- It is implementation independent.

Physical DFD

- It shows how data flow is actually implemented in the system.
 - It describes how data is moved from one entity to another.
 - It shows how the current system operates and how a system will be implemented.
 - It is implementation dependent.
-

Levels of DFD :-

There are mainly 3 levels in DFD:

- 0 – level
 - 1 – level
 - 2 – level
-

1. 0 – level :

- It shows the system as a single process with its relationship to external entities.
 - It represents the entire system as a single process with input and output data indicated by incoming/outgoing arrows.
 - It is also known as fundamental system model or context diagram.
 - It shows an abstract view.
-

2. 1 – level DFD :

- In level 1 DFD, we decompose the context level DFD (0 – level DFD) into multiple components and define each component in detail.
-

3. 2 – level DFD :

- In level 2 DFD, we decompose the 1 – level DFD into more detail and define each necessary detail about system functioning properly.
-

Data Dictionary :-

- A data dictionary contains metadata i.e data about the database.
- It contains information such as what is in the database, who is allowed to access it, where is the database physically stored etc.
- It plays an important role in building a database.

Example table:

Client_id	Client_name	Password	Contact_no	Email
1	Raju	1234	7777	raju@gmail.com
2	Rohan	Abcd	7766	rohan@gmail.com

Field Name | Datatype | Field Length | Constraints | Description

Client_id	number	10	Primary key	Client id, Auto generated
client_name	varchar	20	not null	Name of client
password	varchar	30	not null	Login Password
contact_no	number	10	not null	Contact of client
email	varchar	40	not null	Client email

-
- The data dictionary in general contains information about the following:
 - (i) Names of all tables in the database
 - (ii) Names of each field in the tables of the database
 - (iii) Constraints defined on tables
 - (iv) Physical information about tables like their storage location, storage method etc.
 - The users of the database normally don't interact with the data dictionary; it is only handled by the database administrator.
 - A data dictionary is also called a metadata repository.
-

Types

1. Active :

- It may happen that the structure of the database has to be changed like adding new attributes or removing older ones.
- If those changes are updated automatically in the data dictionary by the DBMS then the data dictionary is an active one.

- It is also known as integrated data dictionary.
-

2. Passive :

- When the DBMS maintains the data dictionary separately and it has to be updated manually then the data dictionary is a passive one.
 - It is also known as non-integrated data dictionary.
 - In this case, there is a chance of mismatch with the database objects and the data dictionary.
 - It gives the well structured and clear information about the database.
 - One can analyze the requirement, any redundancy like duplicate columns, tables etc.
 - It is very helpful for the administrator or any new DBA to understand the database. Since it has all the information about the database DBA can easily able to track any chaos in the database.
 - Since database is very huge and will have lots of tables, views, constraints, indexes etc. it will be difficult for anyone to remember. Data dictionary helps user by providing all details in it.
 - It is a valuable reference in any organization because it provides documentation.
-

Decision Tree :-

- Decision tree is a graph which uses a branching method in order to display all the possible outcomes of any decision.
 - It helps in processing logic involved in decision making and corresponding actions taken.
 - It is a diagram that shows conditions and their alternative actions within horizontal tree framework.
 - It helps analyst to consider the sequence of decisions and identifies the accurate decision that must be made.
-

Decision Table :-

- Decision table is just a tabular representation of all conditions and actions.
 - It is used for describing the complex problem in a precise manner which is easily understandable.
 - We can derive decision table from decision tree.
 - It helps testers to review the effects of combinations of different inputs and other software states that must correctly implement business rules.
-

Components of Decision table :-

(i) **Condition stub** : It is the upper left quadrant which lists all the condition to be checked.

(ii) **Action stub** : It is the lower left quadrant which lists all the action to be carried out to meet such condition.

(iii) **Condition Entry** : It is the upper right quadrant which provides answers to questions asked in condition stub quadrant.

(iv) **Action Entry** : It is the lower right quadrant which provides the appropriate action resulting from the answers to the conditions in the condition entry quadrant.

Example :

User Login & Register System

Condition | Rule 1 | Rule 2 | Rule 3 | Rule 4

Username | F | T | F | T

Password | F | F | T | T

Action | - | - | - | -

Output | Error | Error | Error | Login

Structured English :-

- The structured english uses logical construction and imperative sentences that are designed to carry out instructions for actions.
- In structured english, decisions are made using if – then – else statements.

(i) Decision (if else)

(ii) Loops

(iii) Case

(iv) Sequence of instructions

Feasibility Study :- Section B

- As the name suggests, feasibility study is a study to reveal whether a project is feasible or not.
- It is conducted in order to find answers to the following questions:
 - (i) Do we have required resources and technologies to build the project?
 - (ii) Do we receive profit from the project.

- It tells us whether a project is worth the investment.
 - After the feasibility study, the project may be accepted, accepted with modifications or rejected.
-

Types of feasibility study :-

1. Technical feasibility
 2. Economic feasibility
 3. Legal feasibility
 4. Operational feasibility
 5. Scheduling feasibility
-

1) Technical feasibility :-

- In technical feasibility, we check whether we have required technical resources (like hardware and software) to develop the project.
 - This feasibility study also analyzes technical skills and capabilities of technical team, existing technology can be used or not, maintenance and upgradation is easy or not for chosen technology etc.
-

Economic feasibility :-

- In economic feasibility study, cost and benefit of the project is analyzed.
 - In this feasibility study, a detail analysis is carried out to know what will be cost of the project including hardware and software resources required, design and development cost and so on.
 - It is also analyzed whether project will be beneficial for organization or not.
-

Legal feasibility :-

- In legal feasibility study we investigate whether the project is legal or not.
-

Operational feasibility :-

- In operational feasibility study, we examine whether the project satisfies the requirements identified in the requirement analysis phase.
- Assesses whether the organization's operations can support the project in real-world conditions.

Key aspects :-

- Process fit – Does the project align with existing workflows?
 - User adoption – Will employees / customers accept and adapt?
 - Training needs – What skills must employees acquire?
-

- Organizational impact : Will it improve efficiency or create disruption?
- Support System : Helpdesk, maintenance, upgrades.

Example :

- Implementing a hospital management system must ensure doctors and nurses can use it without disrupting patient care.
-

4) Legal feasibility :-

- Ensures that the project complies with laws and regulations.
-

5) Schedule feasibility :-

- Determines whether the project can be completed within the given timeframe.

Key aspects

- Timeline : Can deadline be realistically met?
- Dependencies : Vendor deliveries, government approvals, external contractors.
- Resource Availability : Are staff and equipment available on time?
- Project delays : Risks and contingency plans.

Example :

- A company bidding for Olympic stadium construction must ensure it can complete before the event begins.
-

6) Market feasibility :-

- Analyze market demand, competition, and customer acceptance for the project.

Key aspects :

- Market size : Is there enough demand?

- Target Audience : Who are the customers?
- Competitor Analysis : Who else provides similar products?
- Market trends : Is the industry growing or shrinking?
- Pricing Strategy : Can the product be competitively priced?
- Distribution channels : Online / offline, wholesale, retail, direct sales.

Example :-

- Before launching an EV scooter, a company must study consumer interest, competition pricing (Ola, Ather) and government subsidies for EV adoption.
-

Steps in feasibility analysis :-

1. Form a project team and appoint a project leader :-

- A group of specialists, analysts, and stakeholders are brought together to conduct the feasibility study.
- One person is chosen as the project leader to coordinate tasks and ensure smooth execution.

Purpose :-

- To ensure diverse expertise is available.
- To assign clear responsibilities and avoid confusion.

Example :-

- In an online banking system project, the project team may include:
 - Software engineers (technical knowledge)
 - Business analyst (requirements and costs)
 - Legal experts (compliance)
 - IT operations staff (infrastructure knowledge)
 - The project leader could be a senior IT manager responsible for managing deadlines and communication.
-

2. Prepare System Flowcharts :-

- A system flowchart is a diagram that shows how data flows through a system, including inputs, processes, storage, and outputs.

Purpose :-

- To understand the current system (if one exists).
- To visualize candidate systems for comparison.
- To identify redundancies and inefficiencies.

Example :-

For a university system, the flowchart may show:

- (i) Input : Student application forms
 - (ii) Processing : Verification of documents, merit list generation
 - (iii) Output : Admission confirmation, fee receipts
-

3. Enumerate Potential Candidate System :-

- List all possible alternative systems that could solve the problem or meet project objectives.

Purpose :-

- To ensure multiple options are available before making a decision.
- To avoid bias towards a single solution.

Example :-

For a library management project, candidate system may include:

1. Manual record-keeping with partial digital support.
 2. A custom-built library management software.
 3. A cloud-based library system subscription.
-

4. Describe and identify characteristics of candidate Systems :-

- Provide a detailed description of each potential system, focusing on its features, strengths, and weakness.

Purpose :-

- To understand how each system would function.
- To provide a fair basis for evaluation.

Example :-

Candidate systems for a hospital management system:

- System A (In-house software) : Fully customizable, requires skilled staff, high initial cost.
 - System B (Vendor package) : Faster to implement, lower cost, limited customization.
 - System C (Cloud solution) : Scalable, subscription-based, data security risks.
-

5. Determine and evaluate performance and cost effectiveness of each candidate system :-

- Measure how well each system meets performance requirements (speed, reliability, accuracy) against its cost.

Purpose :-

- To assess technical performance vs. financial investment.
- To eliminate impractical systems early.

Example :-

For an e-commerce platform:

- System A : Handles 10,000 orders/day but costs ₹50 lakh.
 - System B : Handles 8,000 orders/day but costs ₹20 lakh.
 - System C : Handles 12,000 orders/day but costs ₹70 lakh.
-

6. Weight System performance and cost data

- Assign weights (priority values) to factors like cost, speed, reliability, scalability, etc. then score each system accordingly.

Purpose :-

- To create an objective comparison between candidates.
- To balance trade-offs between performance and cost.

Example :-

For a student information system, weights may be:

- Performance = 40%
 - Cost = 30%
 - Flexibility = 20%
 - Security = 10%
-

7. Select the Best Candidate System :-

Definition :-

Choose the system that provides the best balance of cost, performance, scalability, legal compliance, and organizational fit.

Purpose :-

- To identify the most feasible option.
- To gain stakeholder approval.

Example :-

After weighted analysis, System B (Vendor package) may be selected for a hospital

management project because it provides moderate cost, good performance, and minimal implementation time.

8. Prepare and Report Final Project directive to Management :-

- Documentation the findings, analysis and final recommendations in a feasibility report and present it to top management.

Purpose :-

- To enable management to make a go / no-go decision.
- To provide a roadmap for project execution if approved.

Contents of final report :-

- Executive summary
- System alternatives considered
- Performance & cost evaluation
- Risks identified
- Final recommendation

Example :-

- A retail chain's feasibility report may recommend adopting a cloud-based POS (Point of Sale) system with reason like lower cost, scalability and faster rollout.
-

System Design :- (Section – B)

- System design is the process of defining the architecture, components, modules, interfaces and data of a system to satisfy specified requirements.
- Convert system requirements (what the system should do) into a blueprint for building the system (how it should be done).

Importance

- Provides a roadmap for developers.
 - Reduces errors and misunderstandings during implementation.
 - Ensures the final system is efficient, scalable and maintainable.
-

→ Process of System Design :-

Step 1 : Requirement Analysis

- Study and document the system requirements (from feasibility study & user interviews).

Example : For a university ERP system, requirements include student registration, fee payment.

Step 2 : Define System Objectives

- Clearly state what the system is expected to achieve.

Example : The ERP system should automate admissions, reduce manual errors, and provide real time data access.

Step 3 : Conceptual / System Design

- Identify major system components and how they interact.
- Focus on what the system will do, not implementation details.

Tools : Data flow diagrams, Entity relationship (ER) diagrams, Use case diagrams.

Example : Breaking the ERP into modules → Admission, Finance, Exams, Attendance.

Step 4 : Logical Design

- Translate the conceptual design into a logical representation.

Includes :

- (i) Database design (tables, relationships, keys)
- (ii) Input design (forms, data entry screens)
- (iii) Output design (reports, dashboards, statements)
- (iv) Process design (algorithms, data transformations)

Example : In ERP, a student table with attributes (Student_ID, Name, DOB, Course, fees)

Step 5 : Physical Design

- Focuses on the actual implementation details.

Includes :

- (i) Hardware configuration (servers, networks, storage)
- (ii) Software platforms (OS, DBMS, frameworks)
- (iii) File structures, indexing, system architecture (centralized, distributed, cloud)

Example : ERP system running on cloud based servers with a PostgreSQL database and web interface.

Step 6 : Prototype Development

- Create a working model or mockup of the system.
 - Helps stakeholders visualize the design.
 - Useful for gathering feedback and refining requirements.
-

Step 7 : System Security and Control Design

- Identify risks and security measures.

Includes : authentication, authorization, backups, audit trails, encryption.

Example : ERP require role-based access → only admin can update fee records, faculty can update grades, students can only view results.

Step 8 : Documentation

- Prepare design documentation for developers and stakeholders.

Includes :

- (i) System specifications
- (ii) Data Dictionary
- (iii) Design diagrams

Example : A software design document (SDD) containing all ER diagrams, flowcharts, and schema definitions.

Step 9 : Review & Approval

- The design is reviewed with stakeholders (management, users, technical team).
 - Changes are made before moving to development.
-

3. Stages of System Design

A. Logical Design

- A representation of the system independent of physical implementation.

Focuses on :

- (i) Data flow (how information moves)
- (ii) Processes (transformations and rules)
- (iii) Input / output formats
- (iv) Database structure (entities, attributes, relationships)

Tools used :

DFDs, ER diagrams, Normalization, UML diagrams.

Example :

Logical design of an ATM system includes – Customer inserts card → system verifies PIN
→ transaction is processed → receipt is generated.

B. Physical Design

- A representation of the system with specific details of hardware, software and networks.

Focuses on :

- (i) Hardware requirements (servers, processors, memory)
- (ii) Software platforms (databases, programming languages, middleware)
- (iii) Network architecture (LAN, WAN, cloud)
- (iv) File storage, backup methods, data access methods.

Example :

ATM system physically designed with :

- (i) Hardware : ATM machine, card reader, cash dispenser.
 - (ii) Software : Windows / Linux OS, SQL database.
 - (iii) Network : Secure VPN connecting to central bank servers.
-

(Section – B)

Hardware and Software Selection – Procedure & Phases

1. Introduction :-

- Hardware and software selection is the process of evaluating, comparing and choosing the most suitable computing resources (machines, devices, operating systems, application software etc.) that meet the requirements of a system.
-

2. General procedures for Hardware & Software selection :-

Step 1 : Requirement Analysis

- Clearly define what the system should achieve.
- Identify both functional requirements (what tasks the system should perform) and non-functional requirements (speed, scalability, security).

Example :- For a hospital management system, requirements include patient registration, billing, data security, 24/7 availability.

Step 2 : Prepare System Specifications

- Translate requirements into technical specifications.
- Hardware specs : Processor speed, RAM, storage, networking, backup systems.
- Software specs : Operating system, database, programming language, application modules.

Example : The system must support 500 concurrent users, 2TB storage, and run on Windows/Linux OS.

Step 3 : Identify potential vendors and Alternatives

- Collect information about vendors, suppliers, and open source alternatives.
- Sources : Internet research, vendor catalogs, trade shows.

Example : For databases → Oracle, MySQL, PostgreSQL, MongoDB are candidate options.

Step 4 : Evaluate Candidate Systems

- Compare each alternative based on:
 - (i) Performance : Processing speed, response time
 - (ii) Compatibility : With existing system
 - (iii) Cost : Initial + maintenance + upgrade
 - (iv) Support & maintenance : Vendor reputation, availability updates
 - (v) Scalability : Can it grow with the organization?
-

Step 5 : Cost Benefit analysis

- Estimate total cost of ownership.
 - (i) Direct costs : Purchase, installation, training, maintenance
 - (ii) Indirect costs : Downtime, staff learning curve
 - Compare costs against expected benefits (efficiency, increased revenue, reduced errors).
-

Step 6 : Benchmark Testing and Prototyping

- Conduct pilot runs, demos, or benchmark tests.
 - Check real-world performance before large scale adoption.
-

Step 7 : Select the Best Option

- Use weighted scoring models (assign importance to cost, performance, flexibility, etc.).
 - Select the option with the higher overall score and alignment with business goals.
-

Step 8 : Negotiate with vendor

- Discuss pricing, licensing terms, warranty, after-sales support, upgrade policies.
 - Ensure Service Level Agreements (SLAs) are clear.
-

Step 9 : Implementation and Installation

- Procure and install hardware/software.
- Train users and IT staff.
- Migrate existing data (if applicable).

3. Phases in hardware and software selection :-

Phase 1 : Planning Phase

- Define objectives, scope and budget.
- Identify stakeholders and form a selection committee.

Example : A university forms a committee to select ERP software for admissions.

Phase 2 : Requirement Phase

- Gather requirements from users, management, and technical staff.
- Prioritize critical vs optional features.

Example : A hospital doctors demand fast patient record access; management emphasizes low cost.

Phase 3 : Evaluation Phase

- Identify possible alternatives.
- Evaluate based on performance, cost, compatibility, vendor reputation, scalability, and security.

Example : Compare laptops from Dell, HP, and Lenovo for company staff.

Phase 4 : Selection Phase

- Perform cost benefit analysis.
- Use decision matrices and weighted scoring models.
- Select the system that best meets requirements.

Example : A company chooses cloud ERP instead of on-premise ERP due to lower upfront cost.

Phase 5 : Implementation & Review Phase

- Install and configure hardware/software.
- Train employees.
- Review performance after deployment and resolve issues.

Example : After installing new billing software, a retail chain checks if customers are billed faster than before.

Software Maintenance

Section – B

- Software maintenance refers to all activities carried out after software is delivered to the customer and deployed, to correct faults, improve performance or adapt to changes.
 - Software is not static → it evolves due to user needs, technology changes, bug fixes, and new regulations.
 - IEEE defines it as: “the modification of a software product after delivery to correct faults, to improve performance, or to adapt the product to a changed environment.”
-

→ Maintenance or Enhancement :

A. Maintenance :

- Purpose – To preserve and keep the system working as intended.
- Includes:
 - (i) Corrective maintenance : Fixing defects (bugs).
 - (ii) Adaptive maintenance : Modifying software to run in a changed environment (eg, OS upgrade, new hardware).
 - (iii) Preventive maintenance : Changes made to prevent future problems (refactoring, updating libraries).
 - (iv) Perfective maintenance : Improving performance, efficiency, or maintainability without adding new features.

B. Enhancement :

- Purpose – To extend or upgrade the system’s capabilities.
- (i) Adding new features requested by user or stakeholders.

- Improving the user interface for better experience.
- Supporting new business requirements.

Example: Adding a mobile App module to an existing web application.

→ Primary activities of a Maintenance procedure :

Maintenance is not just fixing bugs – it follows a structured procedure with several key activities:

1. Identification of Need

- Trigger : User reports, system monitoring, or environment changes.
 - Activities :
 - Logging change request or issue.
 - Classifying the request (maintenance vs enhancement).
-

2. Analysis

- Assess the nature, scope and impact of the requested change.
 - Key tasks :
 - Understand the problem or requirement.
 - Perform feasibility analysis (cost, time, resources).
 - Assess risks and dependencies.
-

3. Design (Modification Planning):

- Plan how the change will be implemented.
 - Activities :
 - Design the modification in detail.
 - Ensure consistency with existing system architecture.
 - Prepare test plans for the updated software.
-

4. Implementation

- Make the actual code changes.
 - Activities :
 - Modify source code, configuration, or documentation.
 - Follow coding standards and version control.
 - Update system documentation to reflect the change.
-

5. Testing

- Verify that the modification works and does not break existing functionality.
 - Types of tests:
 - Unit testing : Test modified modules.
 - Regression Testing : Ensure unchanged parts still work.
 - System Testing : Validate overall behavior.
-

6. Documentation

- Updates all user manuals, technical documents, and design docs.
 - Ensure future maintainers can understand the change.
-

7. Release & Deployment

- Deliver the updated version to users.
 - Activities :
 - Installation or patch distribution.
 - Communicating changes to stakeholders.
-

8. Maintenance Review

- Was the change effective?
- Did it introduce new issues?
- Are further changes required?

Conversion and Post Implementation Review

Section – B

- Conversion is the process of changing from the old system to the new system.

It includes:

- Installing the new system.
- Migrating data.
- Training users and IT staff.
- Ensuring a smooth transition with minimum disruption.

Conversion is critical, because even a well defined system can fail if not implemented properly.

Objectives

- Ensure business operations continue smoothly.
 - Avoid / minimize data loss.
 - Provide training to users.
 - Detect and solve unexpected issues quickly.
-

Types

1. Parallel conversion

- Old and new systems run together for a period.
 - Users compare results until new system is proven reliable.
 - Very safe (backup always available).
 - Expensive (double effort and cost).
-

2. Direct conversion (Big Bang approach)

- Old system is completely stopped and new system starts immediately.
- Fast, low cost.
- High risk (if new system fails, no backup).

3. Phased conversion

- New system is introduced in stages/modules.
 - Example: First implement payroll module, then attendance, then reporting.
 - Lower risk, manageable.
 - Slower, requires careful planning.
-

4. Pilot conversion

- New system is implemented in a small area (department/branch) first.
 - If successful → expanded to the whole organization.
 - Limits risks, provides testing ground.
 - Longer time to implement organization wide.
-

→ Post Implementation Review

- Post implementation review is a formal evaluation conducted after the system has been in use for some time (eg, 3–6 months).
 - Purpose : To determine whether the system meets its objectives and identify improvements.
-

Objectives of PIR

- (i) Verify whether the system is working as expected.
 - (ii) Evaluate whether user requirements are satisfied.
 - (iii) Assess system performance, reliability, and usability.
 - (iv) Identify strengths and weaknesses of the new system.
 - (v) Recommend corrective actions or enhancements.
 - (vi) Ensure ROI (return on investment) is being achieved.
-

Activities of PIR

1. System Evaluation

- Check system performance (speed, accuracy, downtime).
- Compare actual results with expected goals.

2. User Feedback

- Collect feedback from end users, managers, IT staff.
 - Identify usability issues, training needs, or satisfaction levels.
-

3. Error and problem Analysis

- Review errors, bugs, or failures reported during initial usage.
 - Evaluate how quickly issues are resolved.
-

4. Cost Benefit analysis

- Compare actual cost vs estimated costs.
 - Evaluate tangible and intangible benefits.
-

5. Documentation Review

- Verify that user manuals, technical documents, and training materials are complete and accurate.
-

→ Database Design (Section – B)

- Database design is the process of structuring data logically and physically so that it supports the required information system.
 - Store data efficiently, avoid redundancy, maintain integrity and ensure easy retrieval.
 - Good design = reliable performance, security, scalability.
-

→ Stages of Database design :

A. Requirement Analysis

- Understand what data needs to be stored.
 - Collect information from users, documents, processes.
 - Define : entities, attributes, relationship, queries, reports.
-

B. Conceptual Design

- Build high-level data model (independent of DBMS).
 - Usually represented using an Entity relationship (ER) diagram.
 - (i) Entity = object (eg, Student, Course)
 - (ii) Attribute = property (eg, Name, Roll no).
 - (iii) Relationship = association (eg, Student enrolls in Course)
-

C. Logical Design

- Convert conceptual model → Relational schema (tables, keys, relationships).
 - Define :
 - Primary keys, foreign keys.
 - Normalization (1NF, 2NF, 3NF, BCNF).
 - DBMS specific but still independent of physical storage.
-

D. Physical Design

- Decide how data will be stored physically.
 - Includes:
 - Indexing
 - File organization
 - Storage structures
 - Clustering
 - Focus on performance and efficiency.
-

E. Implementation

- Create tables, constraints, views, stored procedures in DBMS.
- Load data.
- Test queries and transactions.