COMPUTER APPLICATIONS IN PHARMACY

UNIT 1

TOPIC :

- **Concept of Information Systems and Software :** Information gathering, requirement and feasibility analysis, data flow diagrams, process specifications, input/output design, process lifecycle, planning and managing the project

Information Gathering

→ Information Gathering is the process of collecting relevant data and insights from various sources to understand the requirements of a system or project. It is the foundation of system analysis and helps in determining what a system must do to meet the needs of its users. It also ensures that all stakeholders' perspectives are considered during system development.

Information Gathering Strategies

An **Information Gathering Strategy** refers to the approach or plan used to collect relevant data from various sources. It involves three main components:

1. **Identifying Information Sources**
   Determining where and from whom the relevant information can be obtained.
2. **Choosing a Method of Gathering Information**
   Deciding how to collect the required data (e.g., interviews, questionnaires, observations).
3. **Using an Information Flow Model**
   This includes understanding how information flows within the organization, so the strategy aligns with existing processes.

Requirement Analysis

→ Requirement Analysis is the process of identifying and documenting the needs and expectations of users for a new or modified system. It is a critical phase in the system development life cycle (SDLC), as it determines what the system should do.

Objectives of Requirement Analysis

- To understand the needs of end-users and stakeholders.
- To define the functional and non-functional requirements of the system.
- To bridge the gap between users and system designers.
- To ensure clear, complete, and consistent documentation of system needs.

Types of Requirements

1. **Functional Requirements**
   - Describe what the system should do.
   - Example: "The system must generate monthly sales reports."
2. **Non-functional Requirements**
   - Define system attributes like performance, security, reliability.
   - Example: "The system should respond to a query within 2 seconds."
3. **User Requirements**
   - Describes the needs of end-users in simple language.
   - Example: "A pharmacist must be able to search for a drug by its brand name."
4. **System Requirements**
   - Detailed technical requirements for development.
   - Example: "The system shall use a SQL-based relational database."

Feasibility Analysis

→ Feasibility Analysis is a crucial step in project planning. It involves evaluating the practicality and benefits of a proposed system or solution. The purpose is to determine whether the project is viable before investing significant resources.

Types of Feasibility:

1. **Technical Feasibility**
   Examines whether the technology needed for the system is available, reliable, and suitable.
2. **Economic Feasibility**
   Assesses whether the expected benefits of the system outweigh its costs (cost-benefit analysis).
3. **Operational Feasibility**
   Evaluates if the proposed system will operate successfully in the current organizational environment and if users will accept it.
4. **Legal Feasibility**
   Ensures that the system will comply with all relevant laws, regulations, and contractual obligations.
5. **Schedule Feasibility**
   Determines whether the project can be completed within the desired time frame.

Objective of Feasibility Analysis

- To identify strengths and weaknesses of the proposed solution.
- To provide guidance on necessary improvements and risk areas.
- To evaluate alternatives and select the most viable option.
- To ensure that the project aligns with organizational goals and resources.

DATA FLOW DIAGRAM (DFD)

- A Data Flow Diagram (DFD) is a graphical representation that illustrates the flow of data through an information system. It shows how data enters the system, how it is processed, and where it is stored or exits the system.
- DFDs are typically the first step in understanding or designing a system and are used to visualize data processing across various system components.

Purpose of DFD:

- To create an overview of the system before design or development.
- To visualize how information flows in and out of the system.
- To help in system analysis, requirement specification, and communication among stakeholders.

Components of DFD:

Component	Symbol	Explanation
Process	Circle (◯)	Describes how data input is transformed into output (e.g., “Verify Order”)
Data Flow	Arrow (→)	Represents the movement of data between entities, processes, and data stores
Data Store	Two parallel lines (▭▭)	Storage area for data (e.g., "Order File")
Terminator	Rectangle (▭)	External entities that send or receive data (e.g., Vendor, Purchase Office)

Example System: Goods Receiving System

Entities (Terminators):

- **Vendor**
- **Inspection Office**
- **Purchase Office**

Data Flows:

- **Delivery Note**
- **Items Received Note**
- **Discrepancy Note**

Data Store:

- **Receiving Office Order File**

Data Elements in the Flows:

Delivery Note:

- Order No.
- Vendor Code
- Vendor Name & Address
- Item Name, Item Code
- Delivery Date
- Quantity Supplied, Units

Items Received Note:

- Order No.
- Item Name, Item Code
- Delivery Date
- Quantity Supplied, Units

Discrepancy Note:

- Order No.
- Vendor Code
- Vendor Name & Address
- Item Name, Item Code
- Delivery Date
- Quantity Supplied
- Excess/Deficiency
- No. of Days Late/Early

Receiving Office Order File:

- Order No., Order Date
- Item Name, Item Code
- Vendor Code, Name & Address
- Quantity Ordered
- Delivery Period

Processing Rules:

1. **Compare** order number in the delivery note with that in the order file:
   - If **no match**, **return** the item to the vendor.
2. **If order number matches**, then compare **item codes**:
   - If **no match**, **return** the item.
3. If both match, then **compare quantity delivered** with quantity ordered:
   - If **excess or deficient**, **send a discrepancy note** to the purchase office.
4. If all matches, then **compare delivery date** with expected delivery date:
   - If **late or early**, **send discrepancy note** to the purchase office.

Types of DFDs:

1. Logical DFD

- Focuses on the business and functional requirements.
- Shows what data is needed and what processes happen.
- Ignores the physical details like hardware, files, etc.

2. Physical DFD

- Describes how the system is implemented.
- Shows specific hardware, software, files, and manual procedures.
- Includes details such as screen designs, printers, and file names.

Process Specifications

- Process Specifications refer to the detailed explanation of how a particular process uses input data to produce the desired output in an information system.
- It explains what should be done, using the given input, to achieve a specific output — defining the rules, logic, and conditions associated with the process.

Or

- Process specification is a method of documenting, analyzing, and explaining the internal logic or rules that control how a process transforms data. It focuses on what the process does, rather than how it is physically implemented.

Purpose of Process Specifications:

- To **clearly define** the function of each process in a system.
- To support **system developers** during coding and testing.
- To provide **consistency** in data transformation logic.
- To ensure **accuracy** in system design and behavior.

Requirements of Process Specifications:

1. **Functional Requirements:**
    - Each process specification must **clearly describe the function** it performs.
    - It should define **what output** is produced from given input.
2. **Transformation Rules:**
    - The specification must include all **rules and logic** that govern how data is transformed.
    - These rules ensure the data flow follows the intended **business or operational process**.

3. **Policy Definition:**
    - All **underlying policies** (e.g., company rules, validation criteria, approval limits) related to the data transformation must be included.
    - These policies may be regulatory, procedural, or internal guidelines.
4. **Method Independence:**
    - It is **not mandatory** to describe **how** the transformation will be implemented (e.g., programming logic or algorithm).
    - The focus is only on **what should happen**, not **how it should be done** technically.

Example Scenario:

Process: Verify Delivery

- **Input:** Delivery Note (includes Order No, Vendor, Item Code, Quantity)
- **Output:** Approved Delivery or Discrepancy Report
- **Rules:**
    - If Order No not found → reject delivery.
    - If Item Code doesn't match → send item back.
    - If Quantity differs → generate discrepancy note.
    - If delivery is early/late → send report to Purchase Office.

Input/Output Design

→ Input/Output (I/O) Design is the process of designing how data enters (input) into the system and how results (output) are presented to the user. It is a crucial part of system design as it directly affects user interaction, data accuracy, and system usability.

Objectives of I/O Design:

- To ensure accurate, complete, and secure data entry.
- To provide user-friendly interfaces for both input and output.
- To minimize user errors and reduce unnecessary data entry efforts.
- To present output in a format that is clear, organized, and meaningful to users.
- To enhance system efficiency and decision-making.

Input Design

➢ Input design refers to the process of determining how data should be entered into the system in the most efficient, secure, and accurate manner.

Features of Good Input Design:

- **Easy to Use:** The input forms or screens should be simple and intuitive.
- **Accurate:** Use of validation checks to prevent incorrect data.
- **Consistent:** Data fields should follow standard formats.
- **Secure:** Sensitive information should be protected.
- **Minimized Data Entry:** Avoid asking for redundant or unnecessary data.

Input Design Techniques:

- **Forms Design:** Designing structured forms (manual or digital) for user data entry.
- **Screen Design:** Creating user-friendly data entry interfaces.
- **Input Validation:** Applying checks like range checks, data type validation, mandatory field checks.
- **Control Totals and Batch Controls:** Verifying input completeness for bulk data entries.

Examples of Input Elements:

- Text boxes (for name, address)
- Drop-down lists (for gender, category)
- Date pickers (for DOB, expiry date)
- Checkboxes/radio buttons (for yes/no, options)
- Barcode scanners (for inventory systems)

Output Design

- Output design involves deciding how the information produced by the system should be formatted, presented, and delivered to the user.

Features of Good Output Design:

- **Readable:** Clear fonts, proper layout, and well-organized data.
- **Relevant:** Contains only information required by the user.
- **Timely:** Delivered when the user needs it.
- **Accurate:** Reflects correct and processed data.
- **Appropriate Medium:** Could be on-screen, printed reports, emails, SMS, dashboards, etc.

Output Design Techniques:

- **Report Design:** Structured layout for printed reports.
- **Screen Output Design:** For dashboards, notifications, or real-time alerts.
- **Graphical Outputs:** Charts, graphs, diagrams to visualize trends or analysis.
- **Interactive Output:** User can sort/filter/search the data (e.g., in software dashboards).

Examples of Output Formats:

- Invoice
- Receipts
- Summary reports
- Detailed reports (like sales report, patient history)
- Alerts/Notifications

Process Life Cycle (System Development Life Cycle – SDLC)

→ The System Development Life Cycle (SDLC) is a structured framework that defines a sequence of steps or phases followed by system designers and developers to design, develop, test, deploy, and maintain an information system.

→ Each phase of the SDLC builds on the output of the previous phase, ensuring a logical and efficient development process.

Key Phases of the SDLC (Process Life Cycle):

1. Project Planning and Feasibility Study

- A high-level overview of the project is created.
- Defines the goals, scope, resources, and risks.
- Conducts feasibility analysis (technical, economic, operational).

2. System Analysis / Requirements Definition

- Gathers detailed information about what the users need.
- Analyzes the existing system (if any).
- Outputs a clear set of system requirements (functional and non-functional).

3. System Design

- Transforms requirements into a blueprint for development.
- Includes:
    - Screen layouts
    - Data flow diagrams (DFDs)
    - Database designs
    - Process logic (pseudocode)
    - Business rules

4. Implementation (Coding)

- Developers write the actual code based on design specifications.
- This is where the software is built using programming languages and tools.

5. Integration and Testing

- Combines all the components and tests the entire system.
- Identifies bugs, errors, and system failures.
- Ensures interoperability and system performance.

6. Acceptance, Installation, and Deployment

- Final product is delivered and installed on user systems.
- Users start working with the live system.
- Involves user training and documentation.

7. Maintenance and Support

- System is monitored and updated regularly.
- Fixes bugs, adapts to changing environments, and adds new features.
- This phase is ongoing and crucial for long-term success.

Planning and Managing the Project

→ Planning and Managing a Project involves defining the project's purpose, organizing necessary resources, assigning responsibilities, and ensuring that the project goals are achieved effectively, on time, and within the allocated budget. This phase is critical for project success and typically begins before system design or development.

Role of the SRO (Senior Responsible Owner) / Project Identifier:

The **SRO or project initiator** is responsible for starting the project and ensuring alignment with organizational goals. They must:

1. **Define and justify** the **need** for the project.
2. **Specify, quantify**, and **agree upon** the **desired outcomes and benefits**.
3. **Appoint a Project Manager** and, if necessary, establish a **Project Board** to oversee governance.

Responsibilities of the Project Management Team:

Once the project begins, the **Project Management Team** must:

1. **Plan** how to deliver the **expected outcomes and benefits**.
2. **Engage and manage stakeholders** effectively.
3. **Decide on a management strategy** for the entire delivery process.
4. **Identify and allocate resources** (staff, equipment, time, budget) and ensure they are available when needed.
5. **Develop a Business Case** to justify the project in terms of **cost, benefit, and risk** to support decision-making by the SRO or Project Board.

Responsibilities of the SRO During and After the Project:

- Ensure **Post-Project Reviews** are conducted to measure:
  - Achievement of **intended benefits**
  - Gaps between plan and actual outcomes
- Ensure the **Business Case is updated** to reflect **real-world outcomes.**
- Feed insights from reviews into **strategic planning** to identify:
  - Improvements
  - Changes
  - Future opportunities

Project Planning Checklist:

To manage a project successfully, a detailed **project plan** must be created and approved. This includes:

1. **Confirm the Scope and Purpose:**
   - Define **what the project will cover** and **what it will not**.
2. **Define Deliverables:**
   - List the **tangible outcomes** (e.g., software module, database, report system).
3. **Identify and Estimate Activities:**
   - Break down tasks and **estimate time and resources** required for each.
4. **Schedule Work and Resources:**
   - Allocate **team members and timelines** to each task using tools like Gantt charts.
5. **Identify Risks and Design Controls:**
   - Recognize **potential project risks** and create plans to **prevent or minimize them**.
6. **Document and Gain Approval:**
   - Write a formal project plan and get **approval from the Project Board or stakeholders**.