Roles of a System Analyst

A system analyst must be experienced in working with computers and be able to work with people of all descriptions.

1. **Consultant** (The Problem Solver)
   1. Analyzes existing systems
   2. Gathers requirements
   3. Proposes solutions

2. **Agent of Change** (The Facilitator)
   1. Bridges gaps: Acts as a connection between business departments (who need solutions) and technical teams (who build solutions)
   2. Manages implementation: Oversees the development and deployment of new systems, ensuring they meet specifications and integrate smoothly.
   3. Trains users: Develops training programs and documentation to help employees adapt to new software and processes.

3. **Supporting Expert** (The Technical Guide)
   1. Designs systems: Creates functional specifications, system architecture, and design documents for developers.
   2. Tests and monitoring
   3. Stays up-to-date

Qualities of a System Analyst

A great system analyst combines strong analytical and problem-solving skills with excellent communication, business acumen and technical knowledge to bridge the gap between IT and business needs.

Key qualities include —

1. Problem-solving
2. Communication
3. Strong personal and professional ethics
4. Self discipline and Self-motivated

(1) Problem Solving

Views the analysis of problems as a challenge and enjoys devising workable solutions.

(2) Communication

1. Capable of relating meaningfully to other people over extended periods of time.
2. Has enough computer experience to program and to understand the capabilities of computers.
3. Obtain information requirements from users.
4. Communicate what is needed to programmers.

(3) Personal and Professional Ethics

Involves prioritizing public welfare, honesty, integrity, unbiased judgement and competence. It focuses on protecting user safety and data privacy while delivering quality, reliable systems. It ensures transparency, avoiding conflicts of interest and respecting colleagues and clients.

Design Methodology **

A design methodology is a structured frameword, process or set of principles guiding designers from problem identification to solution.

There are mainly three design methodologies:

1. SDLC (Software Development Life Cycle)
2. Agile
3. UML (Object Oriented SAD)

SDLC

A cost-effective and time-efficient process that development teams use to design and build high-quality software.

SDLC follows a,

- Systematic and orderly approach
  - There is only way forward
  - There is no undo button

SDLC Characteristics:

1. Developed through the use of specific cycle of activities by analyst and user.

2. Each phase has unique user activities and certain deliverable.

SDLC Prerequisites:

Users should know what they need. (clear SRS)
Translation: No repeated changes.
SRS: Software Requirement Specification.

Q When to use SDLC?

1. When the requirements are clear and detailed.
2. When project scope, goals, budget and resources are defined.

Q Where to use SDLC?

1. Developing healthcare and Government systems.
2. Developing banking systems (like ATM)

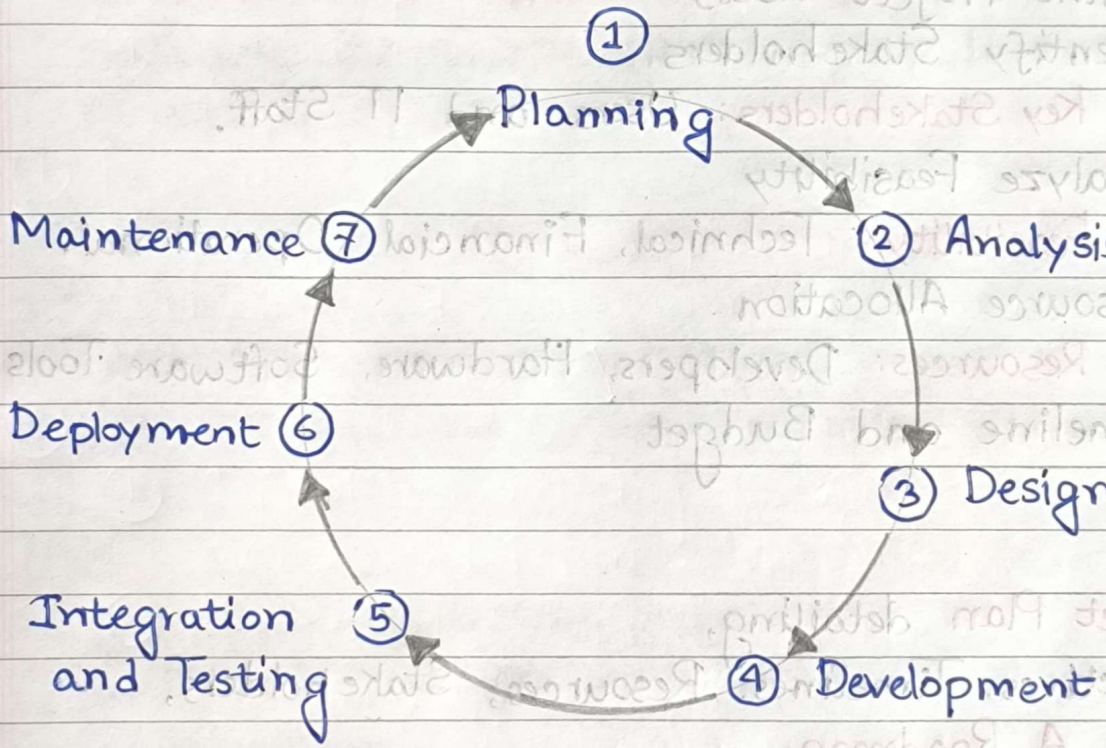

fig. name : SDLC.

1. Planning Phase
2. Analysis Phase
3. Design Phase
4. Implementation / Development Phase
5. Integration and Testing Phase
6. Deployment Phase
7. Maintenance Phase

① Planning Phase

Objective:

Establish the project's,

1. Goals
2. Feasibility
3. Resources
4. Schedule

Tasks:

1. Define Project Goals
2. Identify Stakeholders
   - Key Stakeholders: Users and IT Staff.
3. Analyze Feasibility
   - Feasibility: Technical, Financial, Operational.
4. Resource Allocation
   - Resources: Developers, Hardware, Software Tools.
5. Timeline and Budget

Outcome:

A Project Plan detailing,
Objectives, Timelines, Resources, Stakeholders,
forming A Roadmap.

② Analysis Phase

Objective:

Gather requirements to understand what the system should do.

Tasks:

1. Gathering Requirements (via interviews, surveys)
2. Document Requirements

Requirements

Functional

① Front-end (Presentation layer)
- Buttons, Search bar
- Forms and Menus

Non-functional

① Security Protocols
② Scalability (Load Balancing)
③ Usability (Better UX)

Outcome:

A Requirement Specification Document (SRS) outlining the functional and non-functional needs of the system.

③ Design Phase

Objective:

Define the architecture, data structures, user interface and components of the system.

Tasks:

1. System Architecture Design
2. Database Design
3. UI Design
4. Data Flow Diagrams (DFD)
5. System Security Design
6. Technical Specifications
   1. Technology Stack
      - Programming Languages (Frameworks)
      - Database Management Systems

Outcome: A Design Document including system architecture, database schema, UI mockups, and DFDs.

④ Development Phase

Objective:

Build the system according to the design specifications.

Tasks:

1. Front-end Development
2. Back-end Development
3. Database Implementation
4. Integration (Link front-end and back-end)
5. Security Implementation
   - Authentication
   - Access Control

Outcome:

A function system with core features implemented and ready for testing.

⑤ Testing Phase

Objective:

Ensure the system functions correctly and meets all requirements.

Outcome:

A bug-free functional system, ready for deployment.

Testing Types:

1. Unit Testing
2. Integration Testing
3. System Testing
4. UAT
5. Performace Testing
6. Security Testing
7. Regression Testing

⑥ Deployment Phase

Objective :

Deploy the system to the production environment.

Tasks :

1. Preparation for Deployment
   - Setup servers
   - Ensure Dependencies
2. Data Migration
   - Move data from legacy systems (if any)
3. Deployment
4. User Training
   - Provide training and guides for end-users.

Outcome :

A live system with trained users.

⑦ Maintenance Phase

Objective :

Keep the system running smoothly post-deployment.

Tasks :

1. Provide Bug fixes,
   System and Security updates,
   User Support
2. Monitor performance
   - Ensure the system handles peak loads.

Outcome :

A stable, secure system with continuous updates.