Information System Analysis and Design

Course no: CSE 4109

Chapter 2

Emrana Kabir Hashi

Assistant Professor

CSE, RUET

✅ What is System Analysis?

- **System Analysis** is the process of **studying, understanding, and modeling** an existing or proposed system to identify its **components, processes, problems, and requirements** in order to design an improved system.

🔍 Main Goals of System Analysis

- Understand how the **current system** works
- Identify **problems or inefficiencies**
- Define **user requirements** for the new system
- Create models and documentation to guide system design

✅ What is System Analysis?

- 🛠️ **Key Activities in System Analysis**
- **Problem Identification**
  - What’s wrong with the current system?
- **Requirement Gathering**
  - Interview users, survey stakeholders, observe processes
- **Feasibility Study**
  - Is the solution technically and economically possible?
- **System Modeling**
  - Create **Data Flow Diagrams (DFDs)**, **Entity-Relationship Diagrams (ERDs)**, flowcharts
- **Requirements Documentation**
  - Prepare a clear **Software Requirements Specification (SRS)**
- **User Involvement**
  - Confirm findings and requirements with users

✅ What is System Analysis?

- 🧠 **Why System Analysis Is Important**
- Ensures the system **meets real user needs**
- Reduces the risk of **project failure**
- Helps avoid **costly redesigns** later
- Builds a **clear roadmap** for system design and development

✅ What is System Analysis?

- 📘 **System Analyst Role**
- A **System Analyst** is the person responsible for:
- Understanding the business problems
- Communicating with users and developers
- Creating technical and process documentation
- Acting as a bridge between business and technical teams

✅ What is System Analysis?

- 📌 **Example Scenario**
- A university wants to automate its manual student registration process:
- The analyst observes current steps
- Interviews students, staff, and admins
- Identifies delays and errors
- Defines the new system requirements
- Prepares models and documents for developers to build the system

✅ What is System Design?

- **System Design** is the process of **defining how a system will work**, based on the requirements gathered during **system analysis**. It involves **planning the architecture**, **interfaces**, **databases**, **inputs/outputs**, and **software components** to build an efficient and user-friendly system.
- 📌 **In simple terms**:
  System analysis defines *what* the system should do.
  System design defines *how* the system will do it.

✅ What is System Design?

- 🧩 **Objectives of System Design**
- Translate **user requirements** into a **technical solution**
- Define the **system architecture**
- Specify **hardware, software, database, and network needs**
- Plan **user interfaces** and **security features**

✅ What is System Design?

- **Types of System Design**
- **1. Logical Design**
- Focuses on **what** the system will do
- Describes **processes, data flows, inputs, and outputs**
- Uses tools like:
  - **Data Flow Diagrams (DFDs)**
  - **Entity-Relationship Diagrams (ERDs)**
- **Pseudo-code**

✅ What is System Design?

- **2. Physical Design**
- Focuses on **how** the system will be implemented
- Defines:
  - Hardware and software specifications
  - Database schema
  - Screen layouts and reports
  - Input/output methods
- Network setup

✅ What is System Design?

- 🧠 **Importance of System Design**
- Ensures the system is **efficient, reliable, and secure**
- Reduces future **errors, delays, and cost overruns**
- Bridges the gap between **analysis and coding**
- Ensures **usability** and **maintainability**

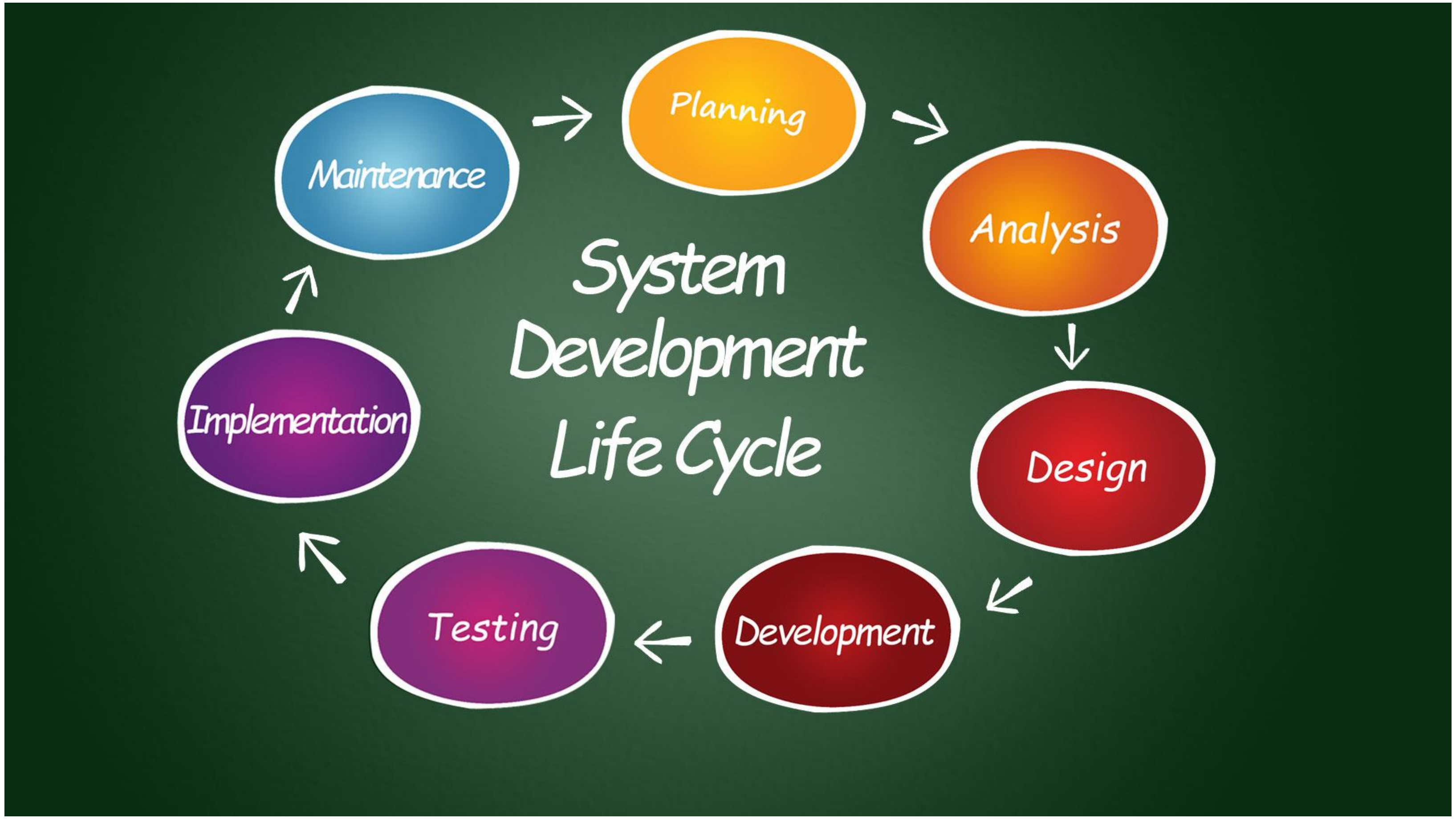
✅ What is System Design?

- 📌 **Example**
- In a **library management system**:
- **Logical design** defines:
  - Users (students, staff), books, borrowing process, return logic
- **Physical design** defines:
  - Database schema (Book table, User table)
  - Interfaces (issue form, search screen)
  - Technology (MySQL, Python, local network)

🎓 System Development Life Cycle (SDLC)

- ✅ **Definition:**
- The **System Development Life Cycle (SDLC)** is a structured process followed to develop, maintain, and replace information systems. It ensures that high-quality systems are delivered efficiently and effectively.
- There are typically **six to seven key stages**, depending on the model used. Below is a **standard 7-phase SDLC** with full explanations.

🎓 System Development Life Cycle (SDLC)



🎓 System Development Life Cycle (SDLC)

- 🔹 **1. Planning Phase**
- 🔍 **Purpose:**
- Understand the **problem or need**
- Determine whether the system is **feasible**
- Identify the **resources**, **budget**, and **timeline**
- 🔧 **Activities:**
- Preliminary analysis
- Feasibility study (Technical, Economic, Legal, Operational, Schedule)
- Resource planning
- Project scheduling
- **Output**: Project plan, feasibility report

🧭 Recognition of Need (Problem Identification Phase)

✅ **Definition:**

- The **Recognition of Need** phase is the initial step in the system development process where an organization **realizes** that a **problem exists** or an **opportunity for improvement** is available, which can be addressed by developing or upgrading an information system.
- 🔍 **Purpose of This Phase**
- Identify the **reason** for starting a new project.
- Understand what is **not working** or **can be improved**.
- Justify the **need for a new or modified system.**
- Serve as a **trigger** for initiating the SDLC process.

🧭 Recognition of Need (Problem Identification Phase)

- 📌 **Key Activities in the Recognition of Need Phase**

Activity	Description
Problem Identification	Detect inefficiencies, complaints, delays, or errors in the current system.
Opportunity Analysis	Discover chances to automate, improve, or innovate business processes.
User Feedback Collection	Gather opinions from users, managers, and stakeholders about existing issues.
Initial Investigation Request	Create a formal request for system analysis or improvement.
Management Approval	Get authorization from top management to explore the issue further.

🧭 Recognition of Need (Problem Identification Phase)

- 🔄 **Example Situations Where Need is Recognized**

Scenario	Need Identified
Customers complain about slow service	Need for an automated order processing system
Employees manually prepare daily reports	Need for a reporting and data visualization tool
Organization is expanding branches	Need for a centralized, scalable management system
Frequent errors in payroll calculations	Need for an accurate and secure payroll system

🧭 Recognition of Need (Problem Identification Phase)

- 🎯 **Outcome of This Phase**
- A **formal statement of need or problem**
- A decision to proceed to the **preliminary investigation or feasibility study**
- The beginning of project documentation

🧭 Recognition of Need (Problem Identification Phase)

- 📝 **Importance of Recognition of Need**
- Helps avoid unnecessary or irrelevant projects
- Aligns system development with **organizational goals**
- Ensures that **resources** are used to solve **real problems**
- Establishes the **foundation** for successful system analysis

✅ Feasibility Study Phase

- 📘 **Definition:**
- The **Feasibility Study** is a structured and systematic evaluation of a proposed system to determine whether it is **practical, achievable, and worth investing in** before proceeding with full-scale development.
- It helps answer the question:
  “Should we proceed with this project?”

✅ Feasibility Study Phase

- 🎯 **Objectives of Feasibility Study**
- Assess if the project is **technically possible**, **economically viable**, and **operationally acceptable**
- Minimize risk of **failure**
- Assist in **decision-making** for system approval
- Recommend whether to proceed, modify, or abandon the project

✅ Feasibility Study Phase

- 📌 **Importance of Feasibility Study**
- Prevents **resource wastage** on unworkable projects
- Reduces **technical and financial risks**
- Helps in gaining **management approval**
- Builds a strong **foundation** for project success

🎓 System Development Life Cycle (SDLC)

- 🔹 **2. System Analysis Phase**
- 🔍 **Purpose:**
- Understand **what the system should do**
- Gather and analyze **requirements** from stakeholders
- 🔧 **Activities:**
- Requirements gathering (interviews, surveys, observation)
- Process modeling (DFD, flowcharts)
- Identifying system constraints
- Documentation of functional and non-functional requirements
- **Output**: Software Requirement Specification (SRS)

🎓 System Development Life Cycle (SDLC)

- 🔹 **3. System Design Phase**
- 🔍 **Purpose:**
- Define **how the system will work**
- Design the architecture, database, interface, and processes
- 🔧 **Activities:**
- Logical design (ERD, process models)
- Physical design (hardware/software specs)
- Database design (normalization, schema)
- UI/UX design (screen layout, navigation)
- **Output**: Design Document Specification (DDS), ERD, DFD

🎓 System Development Life Cycle (SDLC)

- 🔹 **4. Development (Coding) Phase**
- 🔍 **Purpose:**
- Convert design into an actual **software product**
- 🔧 **Activities:**
- Programmers write code in chosen language
- Use of version control tools (e.g., Git)
- Follow design and coding standards
- **Output**: Working software modules

🎓 System Development Life Cycle (SDLC)

- 🔹 **5. Testing Phase**
- 🔍 **Purpose:**
- Ensure the system is **bug-free**, reliable, and performs as expected
- 🔧 **Types of Testing:**
- **Unit testing** – Test individual components
- **Integration testing** – Check combined modules
- **System testing** – Test entire system behavior
- **User Acceptance Testing (UAT)** – Done by end-users
- **Output**: Test report, error log, approved system

🎓 System Development Life Cycle (SDLC)

- 🔹 **6. Implementation Phase**
- 🔍 **Purpose:**
- Install and deploy the system in the **real working environment**
- 🔧 **Implementation Strategies:**
- **Direct cutover** (immediate switch)
- **Parallel running** (old and new run together)
- **Pilot** (test in a small area first)
- **Phased** (gradual rollout)
- **Output**: Live system in user environment

🎓 System Development Life Cycle (SDLC)

- 🔹 **7. Maintenance Phase**
- 🔍 **Purpose:**
- Fix bugs, improve performance, and make updates over time
- 🔧 **Types of Maintenance:**
- **Corrective** – Fix bugs
- **Adaptive** – Adjust to new environment
- **Perfective** – Improve features
- **Preventive** – Avoid future issues
- **Output**: System updates, support logs

🎓 System Development Life Cycle (SDLC)

- 🧠 **Why SDLC Is Important?**
- Ensures **systematic development**
- Improves **quality and efficiency**
- Helps in **risk management**
- Provides **clear documentation**
- Facilitates **team coordination**

🎓 System Development Life Cycle (SDLC)

📁 Deliverables by Stage

Stage	Main Deliverable
Planning	Feasibility Report, Project Plan
Analysis	Requirement Specification Document
Design	Design Documents, ERD, UI Prototypes
Development	Source Code, Compiled Programs
Testing	Test Cases, Bug Reports
Implementation	Installed System, User Manual
Maintenance	Patches, Updates, Change Logs

✅ When is a Project Terminated?

- 🔹 **1. Successful Completion**
- ✅ All objectives and deliverables have been met
- ✅ The client has formally accepted the outcome
- ✅ Project is closed through a structured handover
- 📌 Example: A software system is delivered, tested, accepted, and goes live

✅ When is a Project Terminated?

- 🔹 **2. Project Termination by Cancellation (Early Termination)**
- Occurs when the project is **stopped before completion** due to:
- 🚫 **a. Technical Failure**
- System or product cannot be built as intended
- Technologies chosen are no longer viable
- Example: New hardware requirements cannot support the software.
- 🚫 **b. Budget Overruns**
- Project costs exceed available budget
- Not financially sustainable
- Example: A system projected at $100,000 exceeds $200,000 with no end in sight.

✅ When is a Project Terminated?

- 🚫 **c. Scope Creep or Misalignment**
- Requirements keep changing
- Project no longer aligns with business goals
- Example: The client's priorities shift mid-project, making the current scope irrelevant.
- 🚫 **d. Organizational Change**
- Company undergoes restructuring, merger, or change in strategy
- Project is no longer needed
- Example: A bank project is cancelled after a merger with another company.

✅ When is a Project Terminated?

- 🚫 **e. Risk or Legal Issues**
- Legal, ethical, or compliance risks arise
- Data privacy, regulatory, or contractual problems
- Example: Health data system violates new data privacy regulations.
- 🚫 **f. Lack of Stakeholder Support**
- Key stakeholders or sponsors withdraw support or funding
- Example: Management loses interest or new leadership discontinues the project.

✅ When is a Project Terminated?

- 🧾 **Key Activities During Project Termination**
- **Formal closure meeting** with stakeholders
- **Handover** of final deliverables and documentation
- **Final report** summarizing performance, costs, outcomes
- **Lessons learned** documentation
- **Release of resources** (team, tools, space)
- **Archive** of documents and records

✅ When is a Project Terminated?

Reason for Termination	Description	Example
Completion	Project goals met successfully	New website delivered and accepted
Technical failure	Cannot meet technical requirements	App can't run on targeted devices
Budget issues	Costs exceeded plan	Hardware too expensive
Scope change	Goals no longer match business needs	Client priorities changed
Organizational restructuring	Project no longer fits new direction	Merged company cancels duplicate project
Legal/compliance issues	Regulatory or ethical problems arise	Privacy law violations in health app
Stakeholder withdrawal	Key sponsors no longer support it	Funded project loses backing mid-way

✅ Reasons Why a New System May Not Meet User Requirements

- **1. Incomplete or Incorrect Requirements Gathering**
- Requirements were not fully understood or captured
- Users’ needs were not properly analyzed
- **2. Poor Communication Between Developers and Users**
- Misunderstandings about what users want
- Lack of user involvement during development
- **3. Changing Requirements (Scope Creep)**
- Users change their minds during development
- New needs arise after the design phase

✅ Reasons Why a New System May Not Meet User Requirements

- **4. Technical Limitations**
- System design or technology cannot support certain features
- Performance or usability issues
- **5. Inadequate Testing**
- Testing did not cover real user scenarios
- Bugs or defects left unresolved
- **6. Lack of User Training or Documentation**
- Users don’t know how to use the system properly
- Misuse leads to perception that system is inadequate

✅ Consequences of Not Meeting User Requirements

- User dissatisfaction
- Decreased productivity
- Increased support and maintenance costs
- Potential project failure or need for system redesign

✅ How to Address This Issue

- **Early and continuous user involvement** in requirements and testing
- **Clear, detailed, and validated requirements** documentation
- **Effective communication** between all stakeholders
- **Iterative development** and prototyping for feedback
- **Comprehensive testing** including user acceptance testing (UAT)
- Provide **proper training** and support materials

✅ Considerations for Candidate Systems

- **1. Functionality**
- Does the system fulfill all the **required features** and **business processes**?
- Does it support future expansion or integration?
- **2. Usability**
- Is the system **user-friendly**?
- How much training will users need?
- Is the user interface intuitive and accessible?
- **3. Reliability**
- How stable is the system?
- Does it have a track record of **uptime and error-free performance**?
- **4. Performance**
- Can the system handle the expected **load and volume** of data?
- Is the response time adequate for user needs?

✅ Considerations for Candidate Systems

- **5. Scalability**
- Can the system grow with the organization?
- Does it support adding users, features, or increased data easily?
- **6. Compatibility**
- Is the system compatible with existing **hardware, software, and network infrastructure**?
- Does it support necessary integrations (e.g., with other enterprise systems)?
- **7. Cost**
- What are the **initial costs** (purchase, development)?
- What are the **ongoing costs** (maintenance, licensing, upgrades)?
- **8. Security**
- Does the system provide adequate **data protection** and **access controls**?
- Is it compliant with relevant security standards and regulations?

✅ Considerations for Candidate Systems

- **9. Maintainability**
- How easy is it to **update, fix bugs, or modify** the system?
- Is good documentation provided?
- **10. Vendor Support and Reputation**
- Does the vendor provide reliable **technical support and training**?
- What is their reputation in the industry?
- **11. Legal and Regulatory Compliance**
- Does the system comply with industry-specific regulations (e.g., GDPR, HIPAA)?
- Are there licensing or intellectual property considerations?
- **12. Implementation Time**
- How long will it take to fully implement the system?
- Does it fit the organization’s timeline and deadlines?

🏛️ Political Considerations for a Candidate System

- When choosing or developing a candidate system, it’s crucial to consider the **political environment** within the organization because systems impact power, influence, and decision-making. Ignoring political factors can lead to resistance, conflicts, or failure.
- 🔑 **Key Political Considerations**
- **Stakeholder Influence and Power**
  - Identify who holds power related to the system (e.g., managers, department heads).
  - Understand their interests, support, or opposition to the system.
  - Assess if the system shifts power or control between groups.
- **Resistance to Change**
  - Determine which groups or individuals might resist the new system.
  - Consider how the system affects their roles, job security, or authority.
- **Conflicting Departmental Interests**
  - Different departments may have competing needs or priorities.
  - Analyze how the system satisfies or threatens these competing interests.

🏛️ Political Considerations for a Candidate System

- **Impact on Organizational Politics**
  - Evaluate if the system reinforces or disrupts existing political alliances or conflicts.
  - Consider potential winners and losers from the system adoption.
- **Decision-Making and Control**
  - Clarify who will have control over the system’s data, configuration, and access.
  - Political battles can arise over control and governance.
- **Support from Key Leaders**
  - Gauge the commitment of influential leaders and sponsors.
  - Their support can help overcome political barriers.
- **Communication and Negotiation Needs**
  - Plan how to communicate benefits and address concerns.
  - Anticipate negotiation or compromise to gain political buy-in.

🏛️ Political Considerations for a Candidate System

- 📝 **Example**
- Suppose a new **Enterprise Resource Planning (ERP)** system is a candidate for adoption:
- The **finance department** may gain better control and visibility, increasing their power.
- The **sales department** may resist if the system limits their flexibility.
- IT might want centralized control, which could upset decentralized units.
- Top management’s support is critical to settle conflicts.

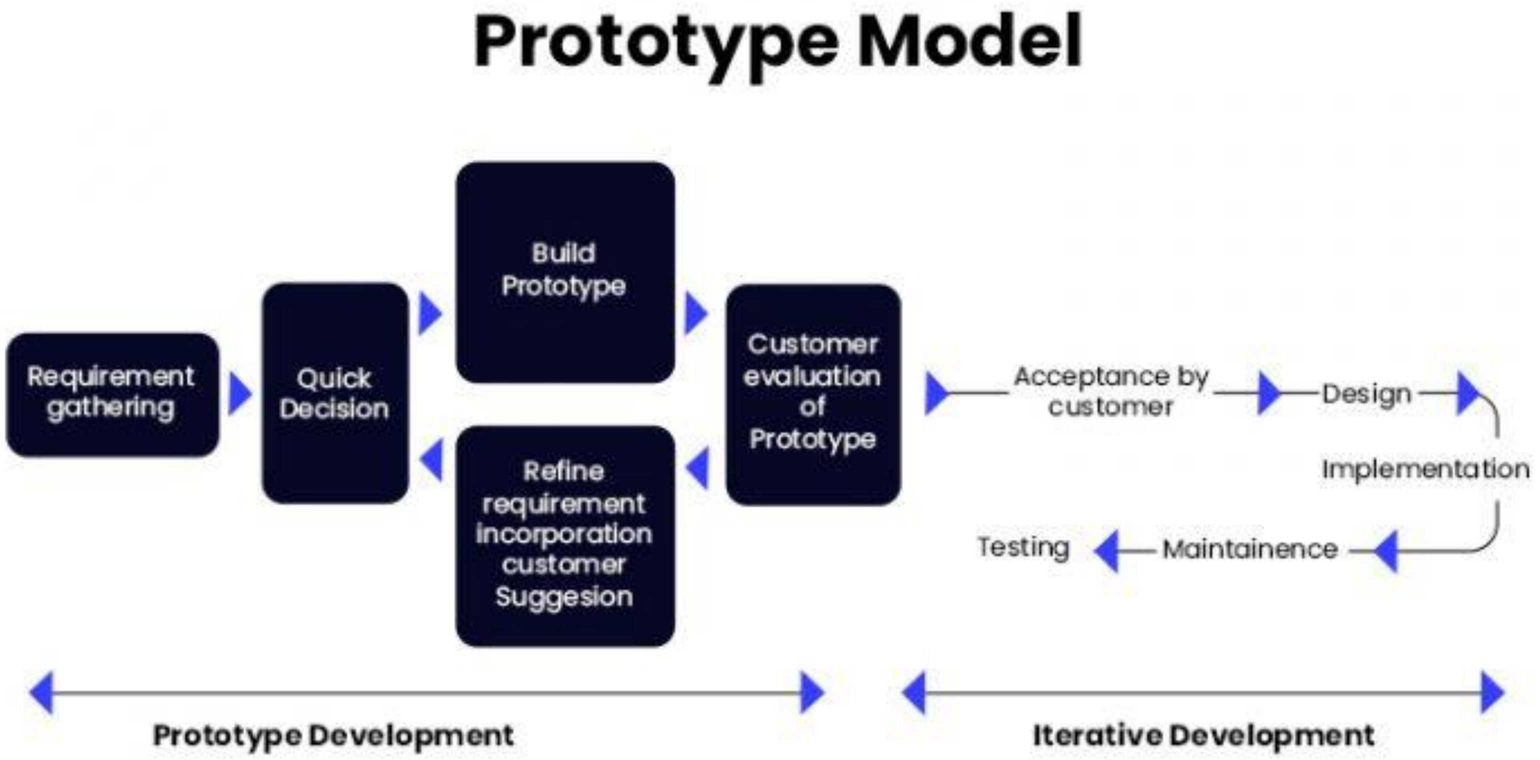
🏛️ Political Considerations for a Candidate System

- ⚙️ **How to Handle Political Considerations**
- Perform a **political stakeholder analysis** early.
- Engage and involve influential stakeholders in system evaluation.
- Communicate transparently to reduce fears and misunderstandings.
- Build coalitions and manage conflicts proactively.

📘 System Development Life Cycle (SDLC) with Prototyping

- **Prototyping** is an iterative approach where a preliminary version (prototype) of the system is built quickly to gather user feedback. This prototype is refined repeatedly until the final system meets user requirements.
- ✅ **Advantages of Using Prototyping in SDLC**
- Early user involvement improves requirements accuracy.
- Reduces risk of system rejection.
- Helps discover requirements that users may not initially express.
- Speeds up development by identifying problems early.
- Improves communication between users and developers.

📘 System Development Life Cycle (SDLC) with Prototyping



📘 System Development Life Cycle (SDLC) with Prototyping

- 🌀 **SDLC Phases with Prototyping**
- **1. Planning**
- Define project goals, scope, budget, and schedule.
- Identify feasibility and resources.
- **2. Requirements Analysis**
- Gather detailed requirements from users and stakeholders.
- Understand what the system must do.
- **3. Prototyping**
- **Build a quick, working model (prototype)** of the system or key components.
- Prototype may be paper-based or a simple software demo.
- Users interact with the prototype and provide feedback.
- Identify missing or unclear requirements early.

📘 System Development Life Cycle (SDLC) with Prototyping

- **4. Refinement and Iteration**
- Refine the prototype based on user feedback.
- Repeat prototyping cycles until the prototype meets expectations.
- Helps clarify requirements and reduces misunderstandings.
- **5. System Design**
- Once prototype and requirements are finalized, design the full system architecture, database, interfaces, and modules.
- **6. Development (Coding)**
- Develop the complete system according to design specifications.

📘 System Development Life Cycle (SDLC) with Prototyping

- **7. Testing**
- Test the system for defects, performance, and usability.
- Validate it against user requirements.
- **8. Implementation (Deployment)**
- Install the system in the live environment.
- Train users and provide documentation.
- **9. Maintenance**
- Monitor system performance.
- Fix bugs and update the system as needed.

📘 System Development Life Cycle (SDLC) with Prototyping

SDLC Phase	Description	Prototyping Role
Planning	Define goals and feasibility	-
Requirements Analysis	Gather user needs	Initial prototype reveals missing needs
Prototyping	Build and refine prototype	Core activity, iterative feedback loop
System Design	Design full system based on refined requirements	Finalizes design from prototype insights
Development	Code the full system	-
Testing	Validate system	Test final product, prototype helps test cases
Implementation	Deploy and train users	-
Maintenance	Ongoing support	-

Thank you!