Department of ICT

System Analysis and Design (SYAD-2432)

CHAPTER 5: SYSTEM DESIGN

2nd Lt. Lencho A.

Masters in Computer Science and Engineering

E-Mail: **lenchoajema@gmail.com**

March, 2025

CHAPTER 5: SYSTEM DESIGN

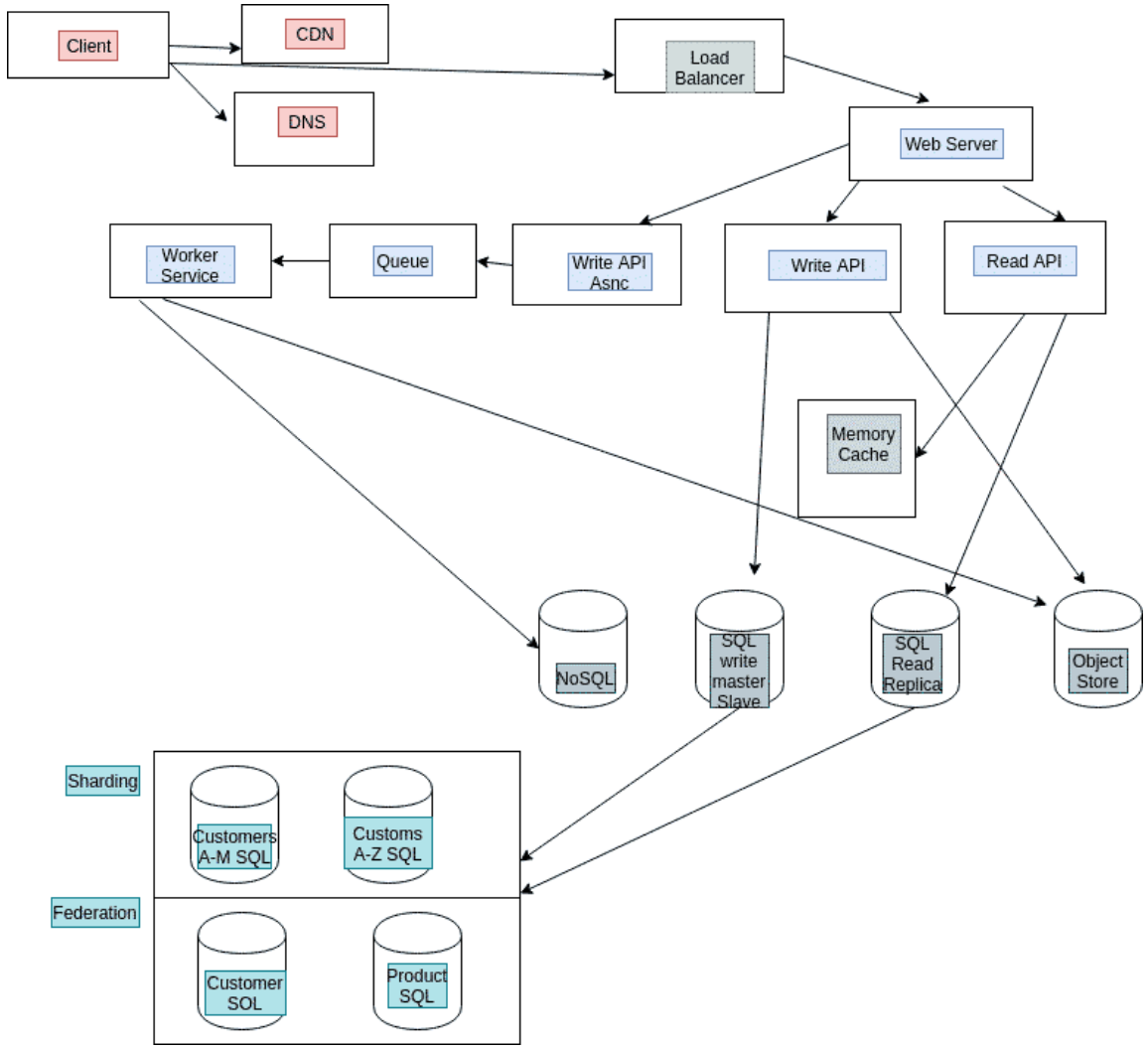
- Inputs to system design
- Outputs from system design
- Types of system design
  - Logical design
  - Physical design
- Design considerations
- Design strategies
  - Top-down strategies
  - Bottom-up strategies

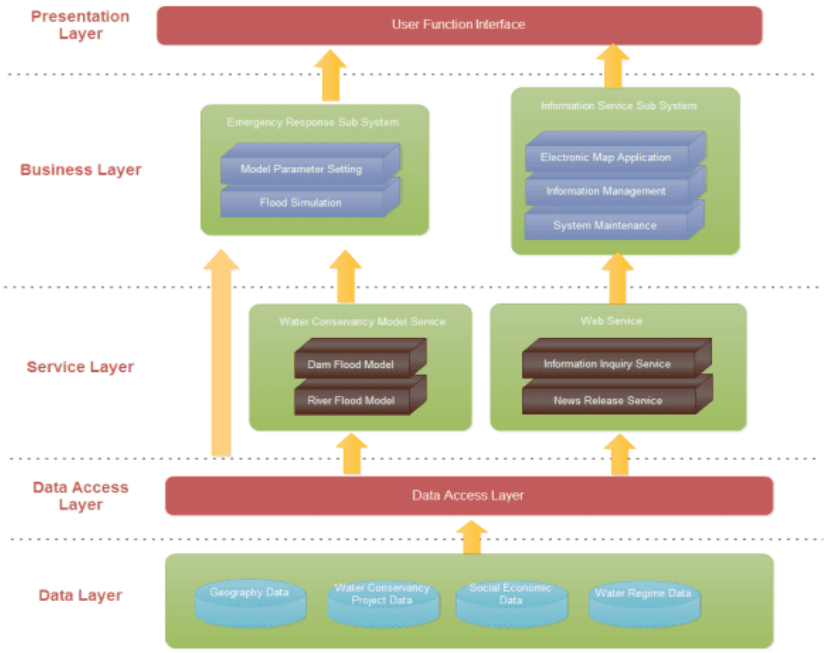
Chapter 5: System Design

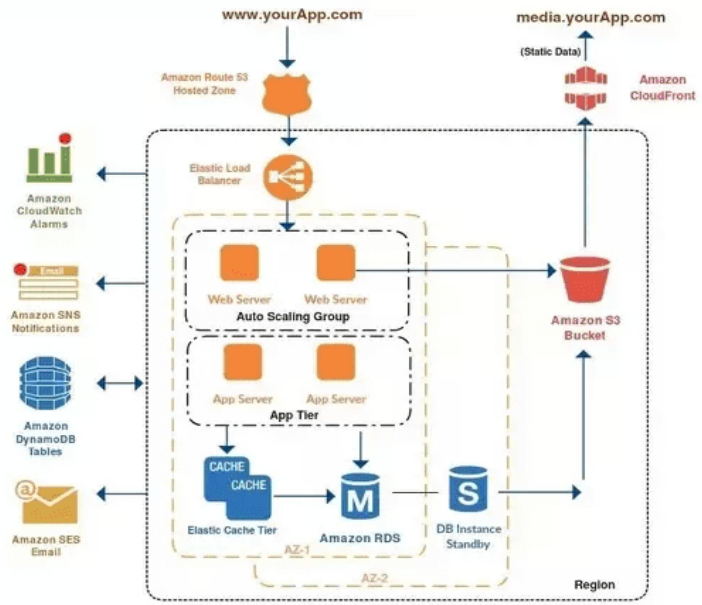
- System design is a crucial phase in the development of any software or information system.
- It involves creating a blueprint for the system, considering various inputs and factors to ensure an effective and efficient design.
- This chapter focuses on understanding the inputs, outputs, types, considerations, and strategies involved in system design.

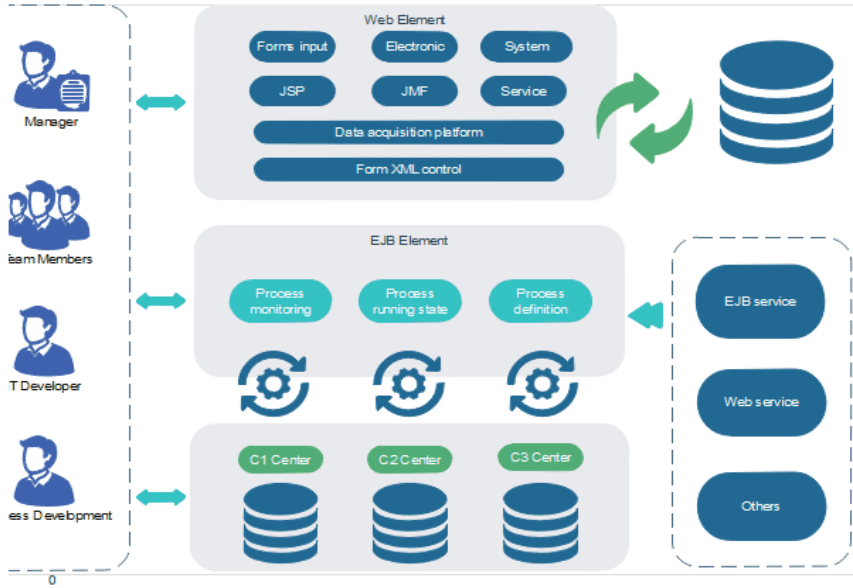
1. Introduction to System Design

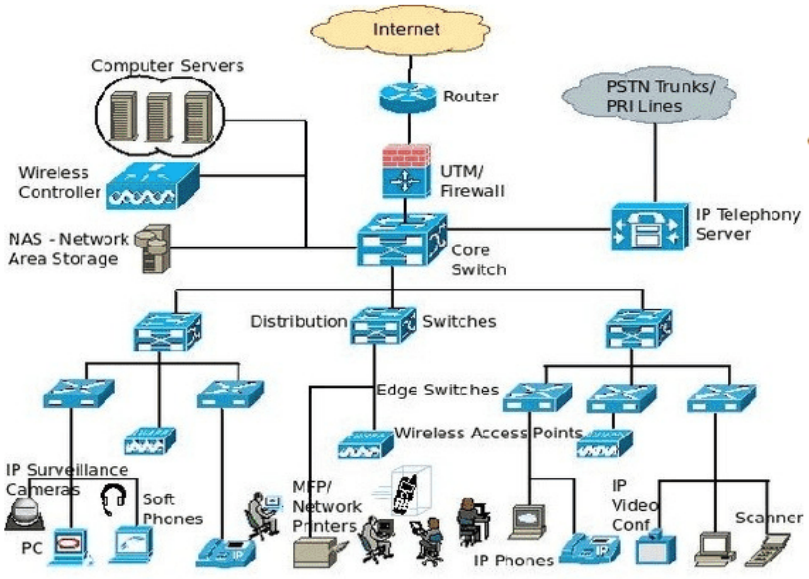
- **System Design** involves defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements.
  - It's crucial in engineering and software development for creating reliable, scalable, and maintainable systems.

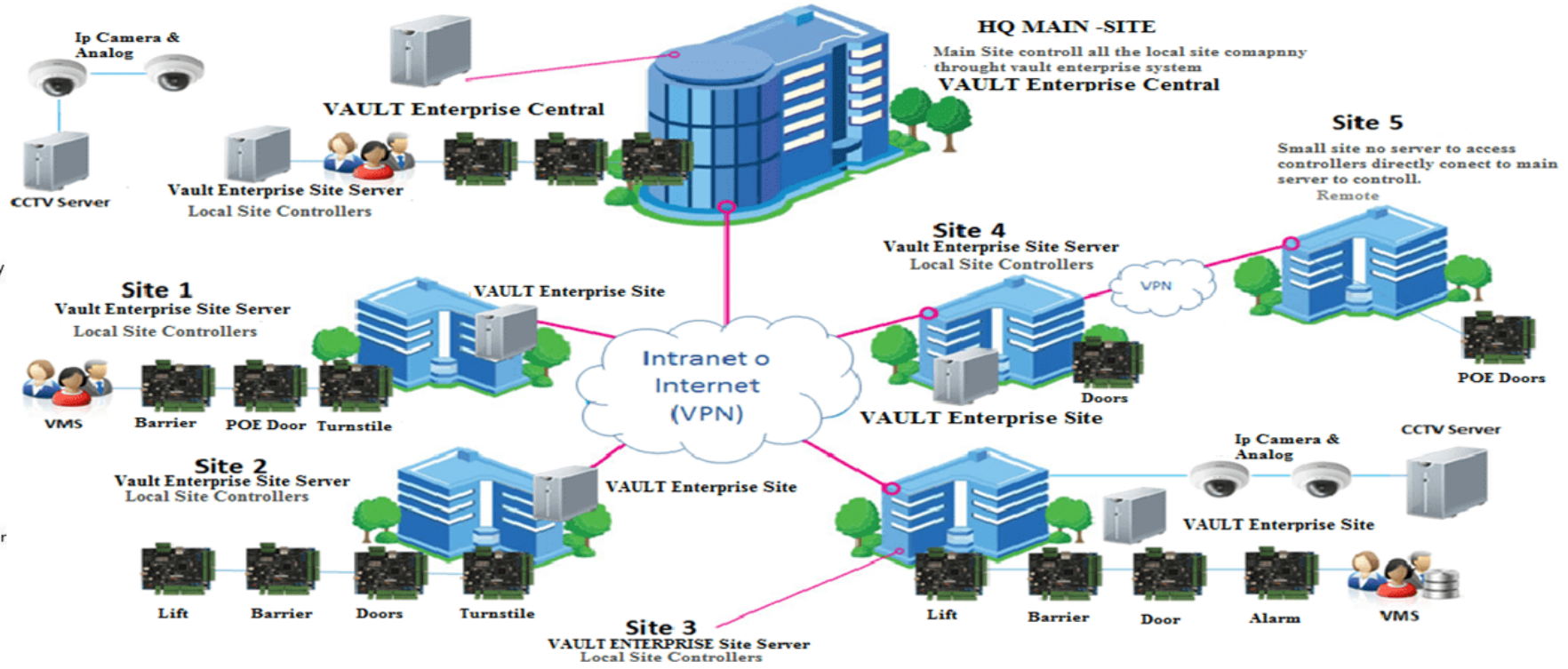



Presentation Layer
User Function Interface
Business Layer
Emergency Response Sub System
Model Parameter Setting
Flood Simulation
Information Service Sub System
Electronic Map Application
Information Management
System Maintenance
Service Layer
Water Conservancy Model Service
Dam Flood Model
River Flood Model
Web Service
Information Inquiry Service
News Release Service
Data Access Layer
Data Access Layer
Data Layer
Geography Data
Water Conservancy Project Data
Social Economic Data
Water Regime Data


www.yourApp.com
media.yourApp.com
(Static Data)
Amazon Route 53 Hosted Zone
Amazon CloudFront
Amazon CloudWatch Alarms
Elastic Load Balancer
Amazon SNS Notifications
Web Server
Web Server
Auto Scaling Group
Amazon S3 Bucket
Amazon DynamoDB Tables
App Server
App Server
App Tier
Amazon SES Email
CACHE
CACHE
Elastic Cache Tier
Amazon RDS
DB Instance Standby
AZ-1
AZ-2
Region


Manager
eam Members
T Developer
ess Development
Web Element
Forms input
Electronic
System
JSP
JMF
Service
Data acquisition platform
Form XML control
EJB Element
Process monitoring
Process running state
Process definition
C1 Center
C2 Center
C3 Center
EJB service
Web service
Others


Internet
Computer Servers
Router
PSTN Trunks/ PRI Lines
Wireless Controller
UTM/ Firewall
NAS - Network Area Storage
Core Switch
IP Telephony Server
Distribution Switches
Edge Switches
Wireless Access Points
IP Surveillance Cameras
Soft Phones
PC
MFP/ Network Printers
IP Phones
IP Video Conf
Scanner


Ip Camera & Analog
CCTV Server
VAULT Enterprise Central
Vault Enterprise Site Server
Local Site Controllers
HQ MAIN -SITE
Main Site controll all the local site comapnny throught vault enterprise system
VAULT Enterprise Central
Site 5
Small site no server to access controllers directly conect to main server to controll.
Remote
Site 4
Vault Enterprise Site Server
Local Site Controllers
VPN
POE Doors
Doors
VAULT Enterprise Site
Site 1
Vault Enterprise Site Server
Local Site Controllers
VAULT Enterprise Site
VMS
Barrier
POE Door
Turnstile
Intranet o Internet (VPN)
Site 2
Vault Enterprise Site Server
Local Site Controllers
VAULT Enterprise Site
Lift
Barrier
Doors
Turnstile
Site 3
VAULT ENTERPRISE Site Server
Local Site Controllers
Ip Camera & Analog
CCTV Server
VAULT Enterprise Site
Lift
Barrier
Door
Alarm
VMS

Inputs to System Design

- Inputs to system design are essential for creating a well-designed system that meets the requirements and expectations of users and stakeholders. Let's explore some examples of inputs in more detail:
- Requirements from users and stakeholders: These are the needs, functionalities, and features that users and stakeholders expect from the system.
  - For example, in the case of a customer relationship management (CRM) system, the inputs could include requirements such as contact management, lead tracking, and sales reporting.
- Business needs: The objectives and goals of the organization play a significant role in system design.
  - For instance, if a company wants to improve its supply chain management, the inputs could include requirements for inventory tracking, supplier management, and order fulfillment.

Inputs to System Design

- Technological constraints: The system design must consider the technological limitations and constraints imposed by the chosen technology stack, infrastructure, or compatibility requirements.
  - For example, if the organization uses a specific database management system, the design inputs should account for its capabilities and limitations.
- Existing infrastructure: The design process should take into account the organization's existing systems, software, and hardware infrastructure.
  - Understanding the current environment helps in ensuring compatibility and integration with the new system.
  - Inputs related to the existing infrastructure could include database systems, networking equipment, and software interfaces.
- Available resources: The availability of resources, such as skilled personnel, budget, and technological resources, influences the system design.
  - For instance, if the organization lacks expertise in a particular programming language, the design inputs may need to consider alternatives or include training programs.

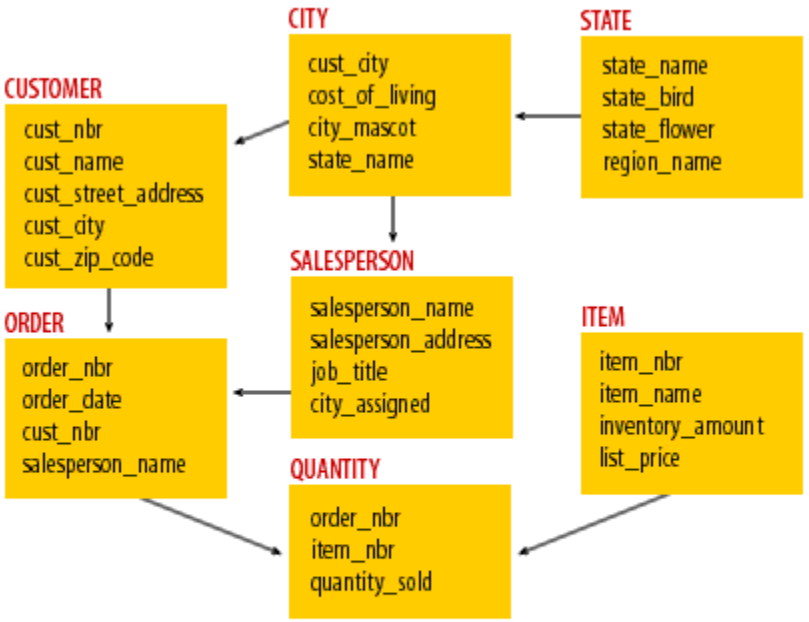
Outputs from System Design

- The outputs of system design are tangible deliverables that serve as the foundation for the development and implementation of the system. Let's explore some examples of outputs in more detail:
- - System specifications: System specifications document the functional and non-functional requirements of the system.
  - This includes detailed descriptions of the system's features, user interactions, and data processing.
  - The specifications serve as a reference for developers and stakeholders throughout the development process.
- - Architectural diagrams: Architectural diagrams provide a visual representation of the system's structure, components, and their relationships. These diagrams can include high-level diagrams, such as system block diagrams or data flow diagrams, as well as detailed diagrams, such as class diagrams or component diagrams. They help in understanding the overall system design and its various components.

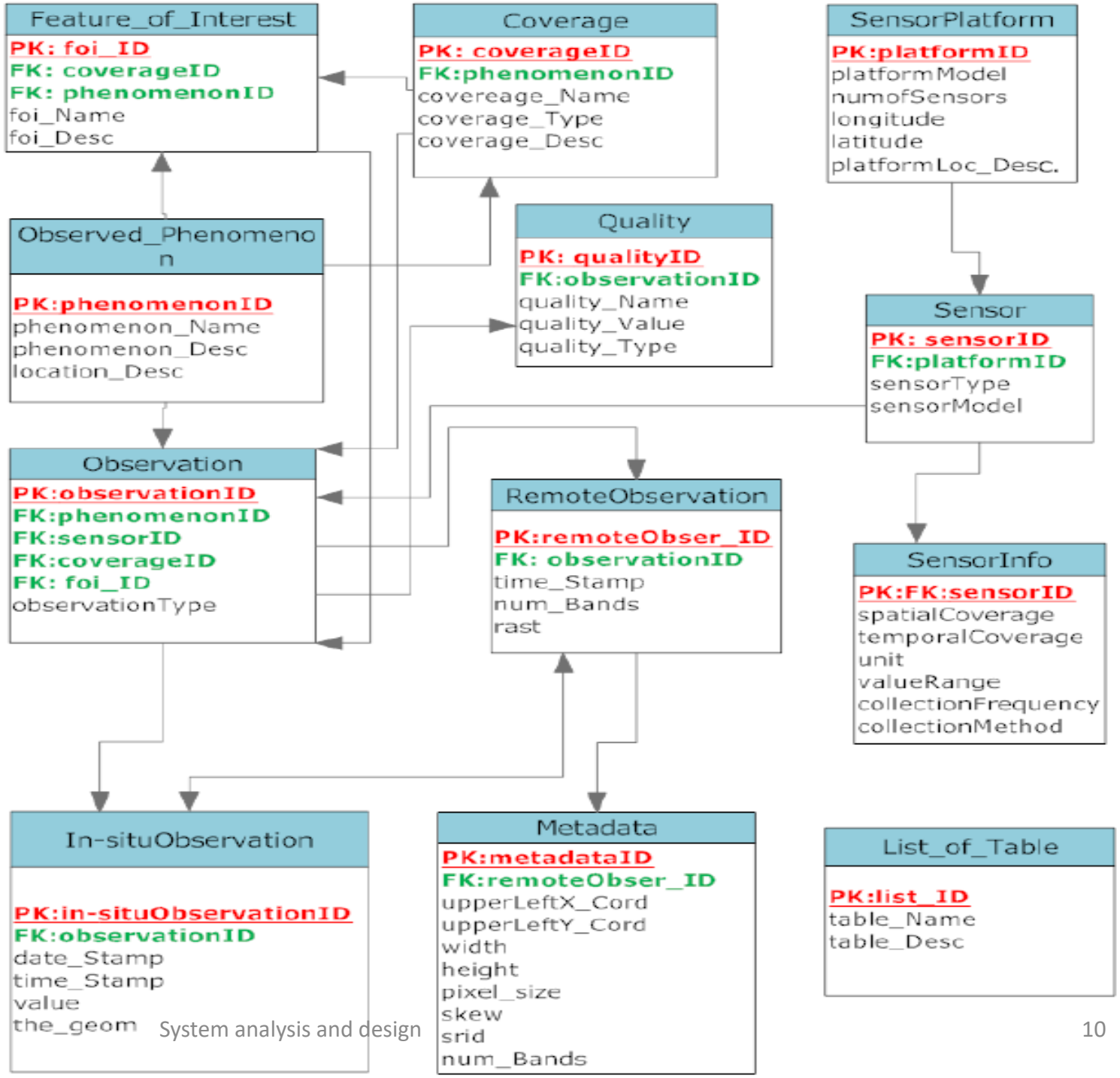
Outputs from System Design

- - Data models: Data models depict the structure, relationships, and attributes of the system's data entities.
  - Examples include entity-relationship diagrams (ERDs) or database schema diagrams.
  - Data models play a crucial role in designing the database and ensuring proper data organization and storage.
- - Interface designs: Interface designs focus on the visual and interaction aspects of the system.
  - They include wireframes, mockups, or prototypes that depict the layout, navigation, and user interactions.
  - Interface designs help in visualizing the user experience and ensuring usability and intuitive interaction with the system.
- - Documentation: Comprehensive documentation captures the design decisions, system configurations, and other relevant information necessary for the development and maintenance of the system.
  - This documentation can include design documents, user manuals, system administration guides, and technical specifications.
  - Documentation is crucial for future reference, system maintenance, and knowledge transfer.

Logical Design

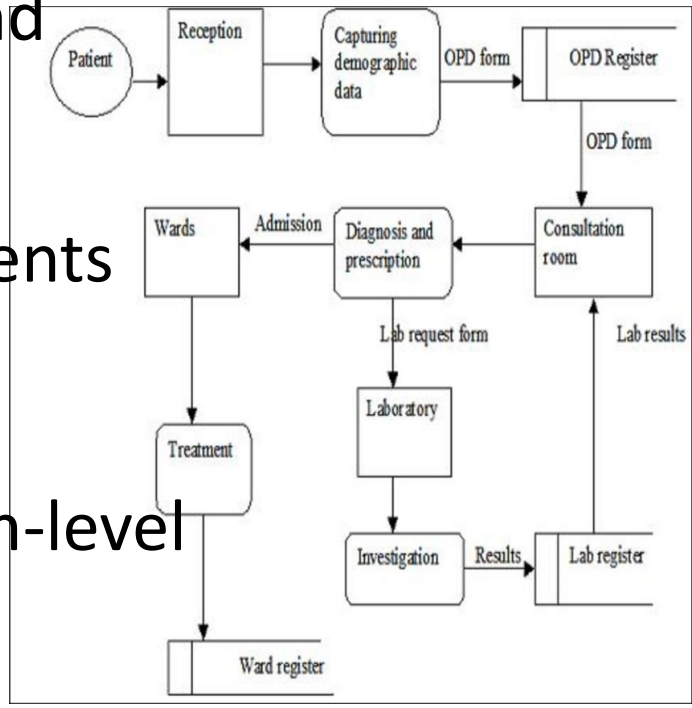


First, here is a very well done, non-redundant database design that will make efficient use of Oracle storage.



Logical Design

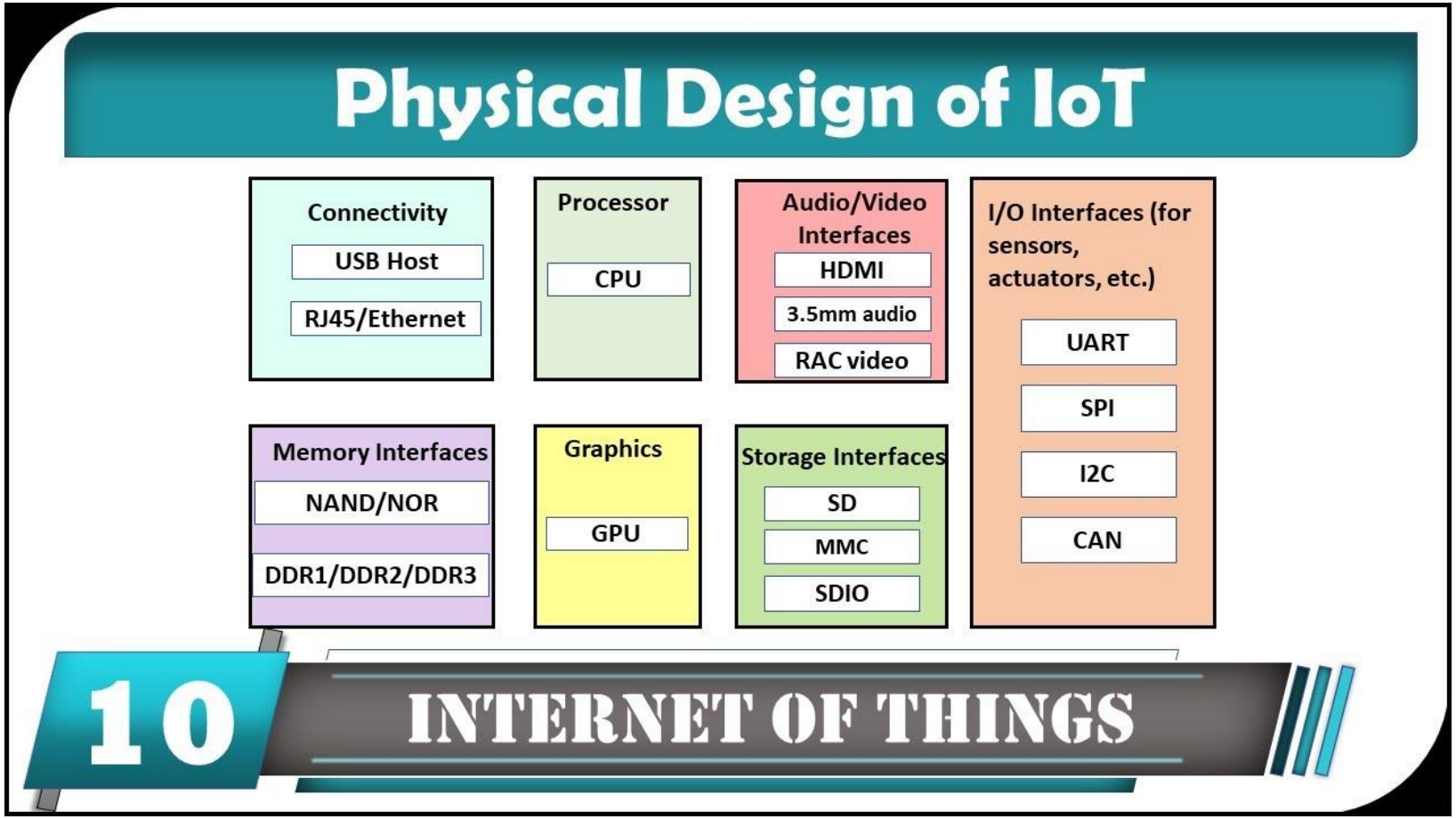
- Logical design focuses on defining the functionality and behavior of the system without considering specific implementation details.
- It focuses on creating a conceptual model that represents the system's structure, processes, and interactions.
- Some examples of elements in logical design include:
- Conceptual models: Conceptual models provide a high-level representation of the system, emphasizing its ***core concepts, relationships, and processes***.
- For example
  - a conceptual model of an e-commerce system could include entities like *customers, products, and orders, along with their relationships*.



Logical Design

- Process flows: Process flows illustrate the sequence and flow of activities and information within the system.
  - These diagrams show how data moves through various processes, highlighting the inputs, outputs, and decision points.
  - Process flows help in understanding the system's workflows and identifying potential bottlenecks or areas for optimization.
- Logical components and relationships: Logical design involves identifying the major components or modules of the system and defining their relationships and interactions.
  - This helps in understanding the system's structure and ensuring that the components work together seamlessly.
  - For example, in a banking system, logical components could include customer management, account management, and transaction processing.

Physical Design



Physical Design

- Physical design involves translating the logical design into a concrete system configuration.
- It takes into account the specific technologies, platforms, and infrastructure that will be used to implement the system.
- Examples of elements in physical design include:
- Hardware: Hardware design involves determining the required physical components, such as servers, storage devices, and networking equipment, to support the system.
  - For example, a web application may require web servers, database servers, load balancers, and firewalls.
- Software: Software desgn focuses on selecting the appropriate software components, frameworks, and programming languages that will be used to develop the system.
  - It includes decisions on programming languages, application frameworks, libraries, and third-party software integrations.

Physical Design

- Network infrastructure: Network design encompasses planning and designing the network architecture necessary for the system's operation.
  - This includes ***determining network topology, protocols, security measures, and bandwidth requirements***.
  - Network design ensures reliable communication between system components and external systems.
- Database design: Database design involves designing the database schema, defining the tables, relationships, and constraints that will store and manage the system's data.
  - It also includes decisions on indexing, partitioning, and data retrieval strategies.
  - Database design ensures efficient data storage, retrieval, and manipulation.

Design Considerations

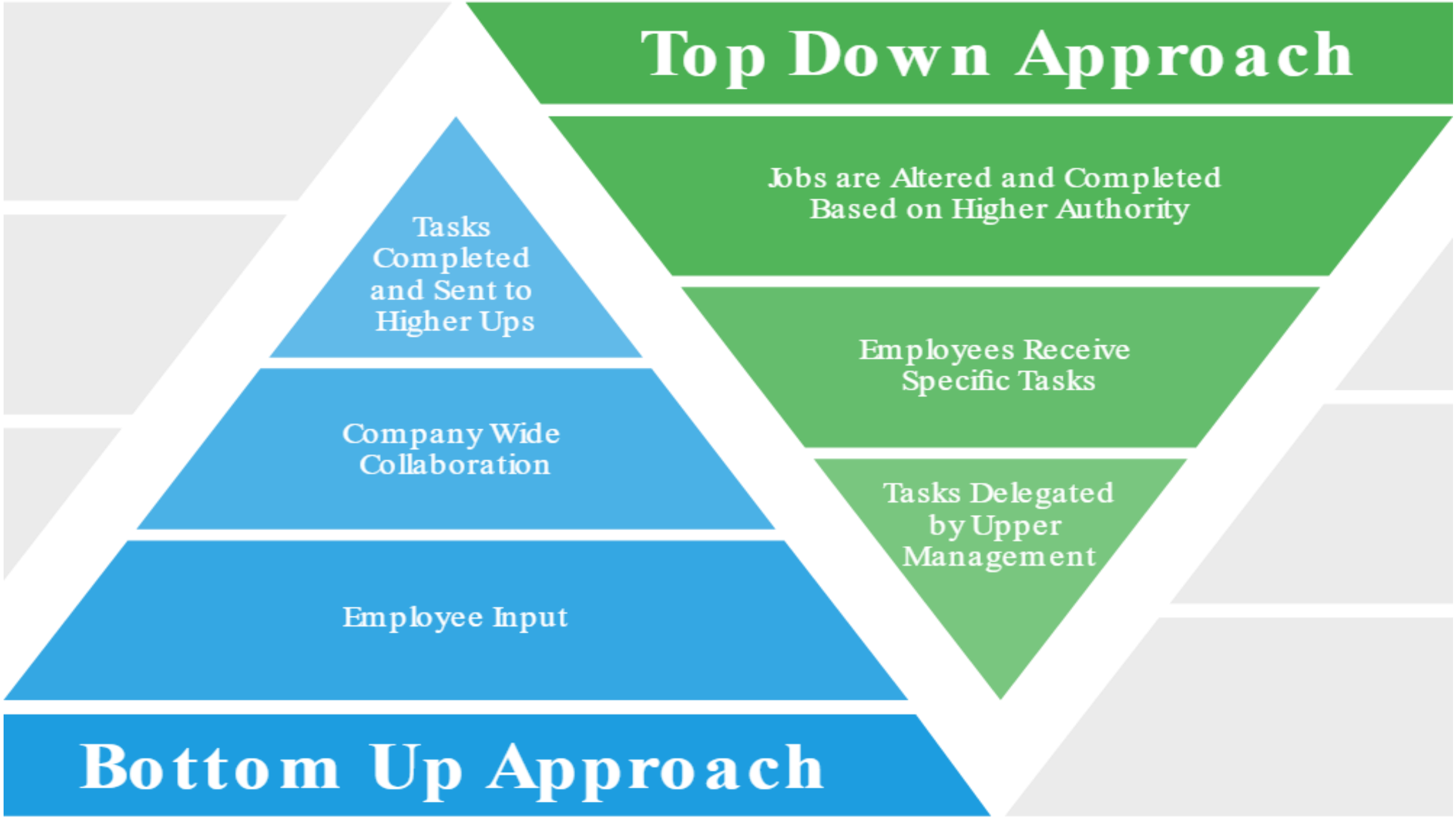
- Design considerations are crucial factors that need to be taken into account during the system design process.
- Considering these factors helps in creating a system that meets its objectives, is robust, and satisfies the requirements of stakeholders.
- Let's explore some key design considerations in more detail:
- Functionality: Designing a system that meets the required functionality is a primary consideration.
  - It involves understanding the needs of the users and stakeholders and ensuring that the system provides the necessary features and capabilities.
- Performance: Performance considerations involve designing a system that performs ***optimally in terms of speed, responsiveness, and resource utilization***.
  - It includes optimizing algorithms, data structures, and system configurations to achieve the desired performance levels.
- Scalability: Scalability refers to the system's ability to handle increased data volume, user load, or processing requirements.
  - Designing for scalability involves considering factors like distributed computing, load balancing, and data partitioning to ensure that the system can handle growth without compromising performance.

Design Considerations

- Reliability: Reliability focuses on building a system that is robust and minimizes downtime.
  - It involves designing for fault tolerance, error handling, and backup and recovery mechanisms to ensure system availability and data integrity.
- Security: Security considerations involve protecting the system and its data from unauthorized access, breaches, and data loss.
  - It includes implementing authentication, authorization, encryption, and other security measures to safeguard sensitive information and prevent security vulnerabilities.
- Usability: Usability focuses on designing a system that is easy to use, intuitive, and provides a positive user experience.
  - It involves considering factors like user interface design, navigation, error handling, and accessibility to ensure that the system is user-friendly.
- Maintainability: Maintainability involves designing a system that is easy to update, maintain, and troubleshoot.
  - It includes considerations like modular design, code readability, documentation, and logging to facilitate system maintenance and future enhancements.
- Cost-effectiveness: Designing a system that is cost-effective involves considering budget constraints, resource utilization, and return on investment.
  - It requires making design choices that balance functionality, performance, and other considerations with the available resources.

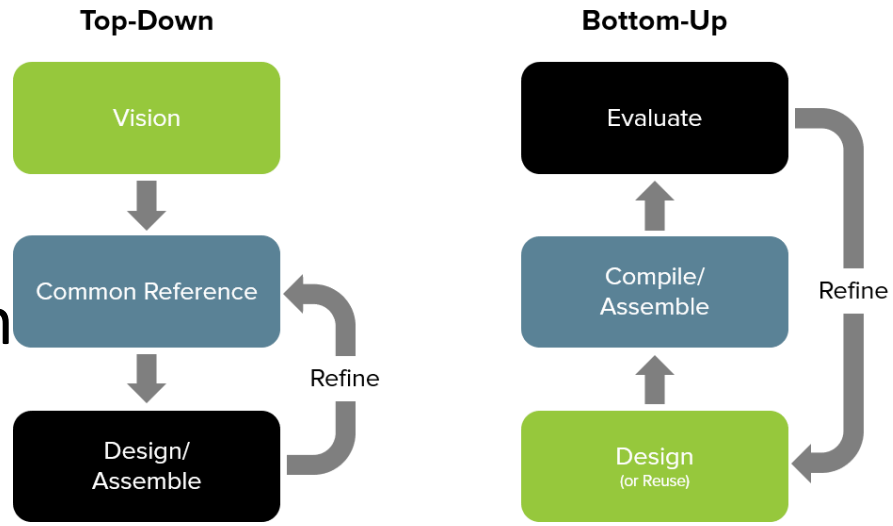
Design Strategies

- Design strategies are overarching approaches that guide the overall system design process.
- They provide a structured and systematic framework for designing an effective system.
- Let's explore the two main design strategies in more detail:


Top Down Approach
Jobs are Altered and Completed Based on Higher Authority
Employees Receive Specific Tasks
Tasks Delegated by Upper Management
Tasks Completed and Sent to Higher Ups
Company Wide Collaboration
Employee Input
Bottom Up Approach

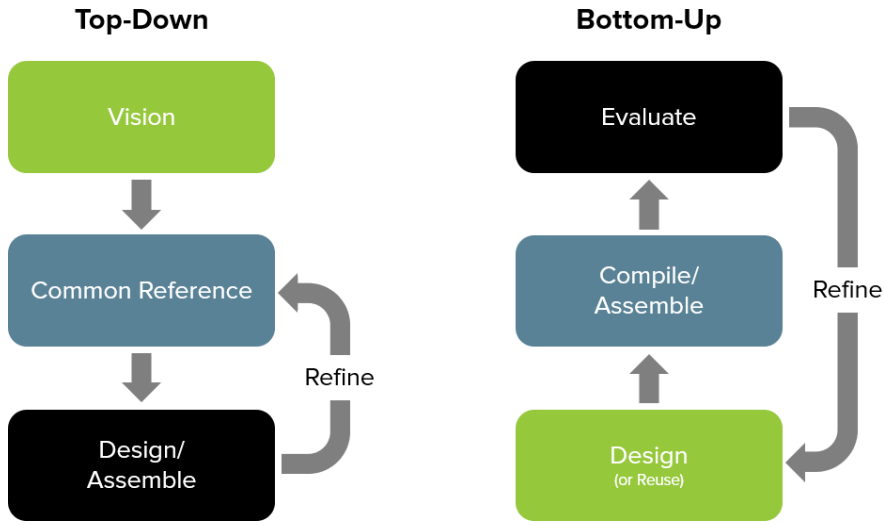
Top-Down Strategies

- Top-down design starts with a high-level view of the system and gradually breaks it down into smaller components and subsystems.
- This strategy involves the following steps:
- Understanding the overall structure: The design process starts by gaining a clear understanding of the system's goals, objectives, and overall architecture.
  - This includes identifying the main functionalities, system boundaries, and external dependencies.



Top-Down Strategies

- Decomposition into detailed levels: Once the high-level structure is established, the system is decomposed into smaller components and subsystems.
  - Each component is analyzed and further broken down into more detailed levels, creating a hierarchical structure.
- Iterative refinement: The design is refined iteratively at each level of decomposition.
  - This involves defining the interfaces, interactions, and relationships between the components, ensuring coherence and compatibility between the different parts of the system.



Bottom-Up Strategies

- Bottom-up design starts with individual components and builds up to form a complete system.
- This strategy involves the following steps:
- Identifying and designing building blocks: The design process begins by identifying the individual components or modules that will be part of the system.
  - Each component is designed in detail, considering its functionality, interfaces, and interactions.

Bottom-Up Strategies

- Integration: The designed components are then integrated to create a larger system.
  - This involves connecting the components, ensuring their proper interaction and compatibility.
  - Integration may require designing communication protocols, data exchange formats, or middleware.
- Incremental development: Bottom-up design often follows an incremental development approach, where the system is built gradually.
  - This allows for early validation of individual components and the flexibility to add new components or modify existing ones as the system evolves.

Difference between top-down and bottom-up strategies

Top-down strategy:

* Start with a broad goal or vision.

* Break down the goal into smaller tasks or objectives.

* Assign tasks to individuals or teams.

* Monitor progress and make adjustments as needed.

Bottom-up strategy:

* Start with individual or team tasks.

* Identify how the tasks contribute to the overall goal.

* Work together to develop a plan to achieve the goal.

* Communicate progress and collaborate with others to make adjustments as needed.

S. No.	TOP DOWN APPROACH	BOTTOM UP APPROACH
1.	In this approach We focus on breaking up the problem into smaller parts.	In bottom up approach, we solve smaller problems and integrate it as whole and complete the solution.
2.	Mainly used by structured programming language such as COBOL, Fortran, C, etc.	Mainly used by object oriented programming language such as C++, C#, Python.
3.	Each part is programmed separately therefore contain redundancy.	Redundancy is minimized by using data encapsulation and data hiding.
4.	In this the communications is less among modules.	In this module must have communication.
5.	It is used in debugging, module documentation, etc.	It is basically used in testing.
6.	In top down approach, decomposition takes place.	In bottom up approach composition takes place.
7.	In this top function of system might be hard to identify.	In this sometimes we can not build a program from the piece we have started.
8.	In this implementation details may differ.	This is not natural for people to assemble.

9.	**Pros-** **•Easier isolation of interface errors** **•It benefits in the case error occurs towards the top of the program.** **•Defects in design get detected early and can be corrected as an early working module of the program is available.**	**Pros-** **•Easy to create test conditions** **•Test results are easy to observe** **•It is suited if defects occur at the bottom of the program.**
10.	Cons- •Difficulty in observing the output of test case. •Stub writing is quite crucial as it leads to setting of output parameters. •When stubs are located far from the top level module, choosing test cases and designing stubs become more challenging.	Cons- •There is no representation of the working model once several modules have been constructed. •There is no existence of the program as an entity without the addition of the last module. •From a partially integrated system, test engineers cannot observe system-level functions. It can be possible only with the installation of the top-level test driver.

Examples of top-down and bottom-up strategies:

- To illustrate the difference between top-down and bottom-up strategies, let's consider the design of a web application for a restaurant.
- In a top-down approach, the design process may start with identifying the main functionalities of the system, such as menu management, online ordering, and table reservations.
- The system's high-level architecture is established, and then each functionality is further decomposed into more detailed components.
- For example, menu management may involve designing components for menu creation, dish categorization, and pricing.
- The design is refined iteratively, ensuring that all the components fit together seamlessly.

Examples of top-down and bottom-up strategies:

- In a bottom-up approach, the design process may begin by identifying the individual components that will be part of the system, such as user authentication, database management, and payment processing.
- Each component is designed in detail, considering its functionalities and interfaces.
- The components are then integrated to form the complete system, connecting them through well-defined interfaces.
- This approach allows for focusing on the detailed design and implementation of each component and ensuring their proper integration.

Summary

- System design is the process of defining the architecture, components, and interfaces of a system.
- It is a complex process that requires careful consideration of a number of factors, including requirements, constraints, goals, performance, scalability, security, usability, maintainability, and design strategies.
- By following the principles and strategies outlined in this chapter, you can increase the chances of success in your system design projects.

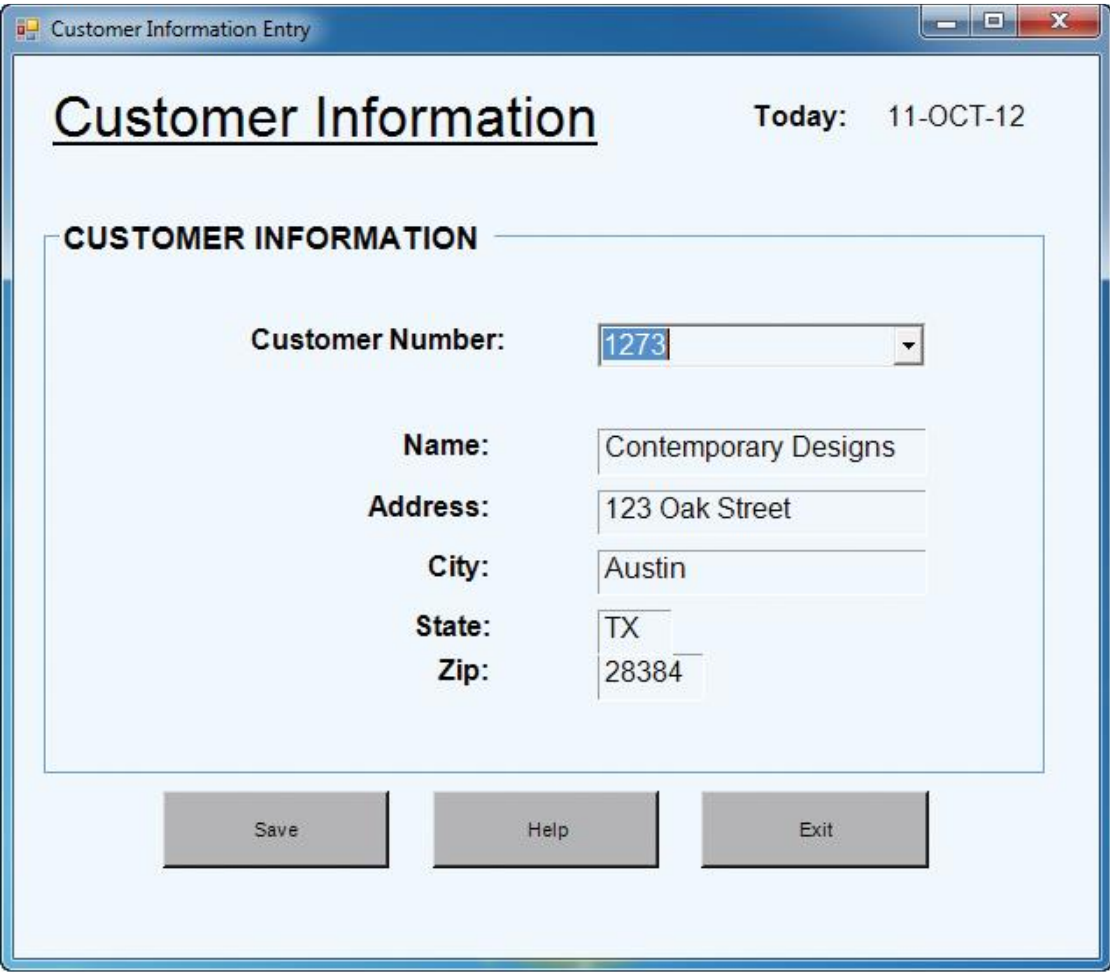


TABLE 8-2: Guidelines for Designing Forms and Reports

Guideline	Description
Use meaningful titles	Clear and specific titles describing content and use of form or report
	Revision date or code to distinguish a form or report from prior versions
	Current date that identifies when the form or report was generated
	Valid date that identifies on what date (or time) the data in the form or report were accurate
Include meaningful information	Only needed information displayed
	Information provided in a usable manner without modification
Balance the layout	Information balanced on the screen or page
	Adequate spacing and margins used
	All data and entry fields clearly labeled
Design an easy navigation system	Clearly show how to move forward and backward
	Clearly show where you are (e.g., page 1 of 3)
	Notify user of the last page of a multipage sequence

Vague title

Difficult to read: information is packed too tightly

Pine Valley Furniture

CUSTOMER INFORMATION

CUSTOMER NO:	1273	
NAME:	CONTEMPORARY DESIGNS	
ADDRESS:	123 OAK ST.	
CITY-STATE-ZIP:	AUSTIN, TX 28384	
YTD-PURCHASE:	47,285.00	
CREDIT LIMIT:	10,000.00	
YTD-PAYMENTS:	42,656.65	
DISCOUNT %:	5.0	
PURCHASE:	21-JAN-12	22,000.00
PAYMENT:	21-JAN-12	13,000.00
PURCHASE:	02-MAR-12	16,000.00
PAYMENT:	02-MAR-12	15,500.00
PAYMENT:	23-MAY-12	5,000.00
PURCHASE:	12-JUL-12	9,285.00
PAYMENT:	12-JUL-12	3,785.00
PAYMENT:	21-SEP-12	5,371.65
STATUS:	ACTIVE	

No navigation information

No summary of account activity

Easy to read: clear, balanced layout

Clear title

Pine Valley Furniture

Pine Valley Furniture

Page: 2 of 2

Detail Customer Account Information

Today: 11-OCT-12

Customer Number: 1273

Name: Contemporary Designs

DATE	PURCHASE	PAYMENT	CURRENT BALANCE
01-Jan-12			0.00
21-Jan-12	(22,000.00)		(22,000.00)
21-Jan-12		13,000.00	(9,000.00)
02-Mar-12	(16,000.00)		(25,000.00)
02-Mar-12		15,500.00	(9,500.00)
23-May-12		5,000.00	(4,500.00)
12-Jul-12	(9,285.00)		(13,785.00)
12-Jul-12		3,785.00	(10,000.00)
21-Jul-12		5,371.65	(4,628.35)
YTD SUMMARY	(47,285.00)	42,656.65	(4,628.35)

Help | Prior Screen | Exit

Summary of account information

Clear navigation information

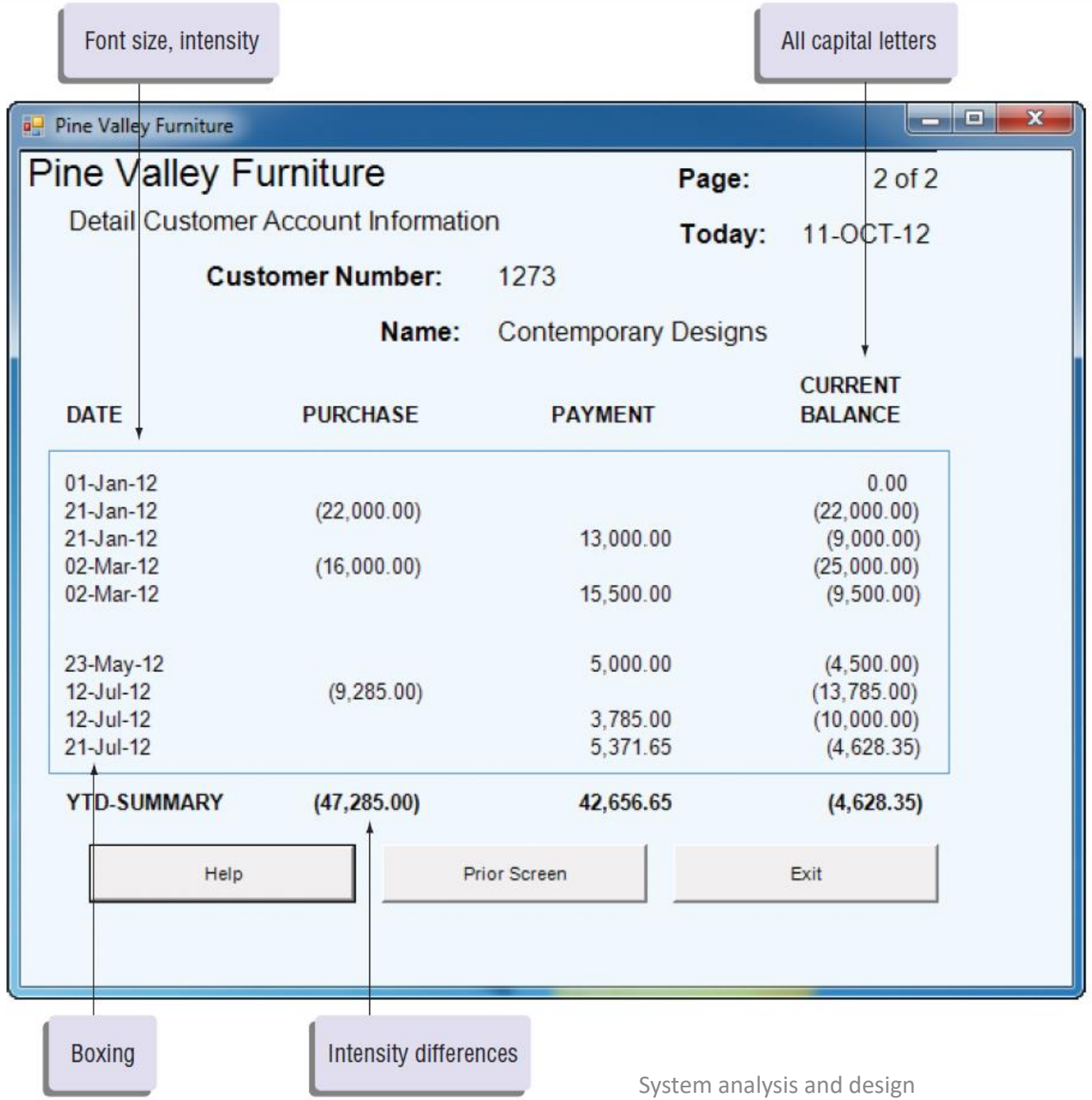


FIGURE 8-6
A form in which several types of highlighting are used.

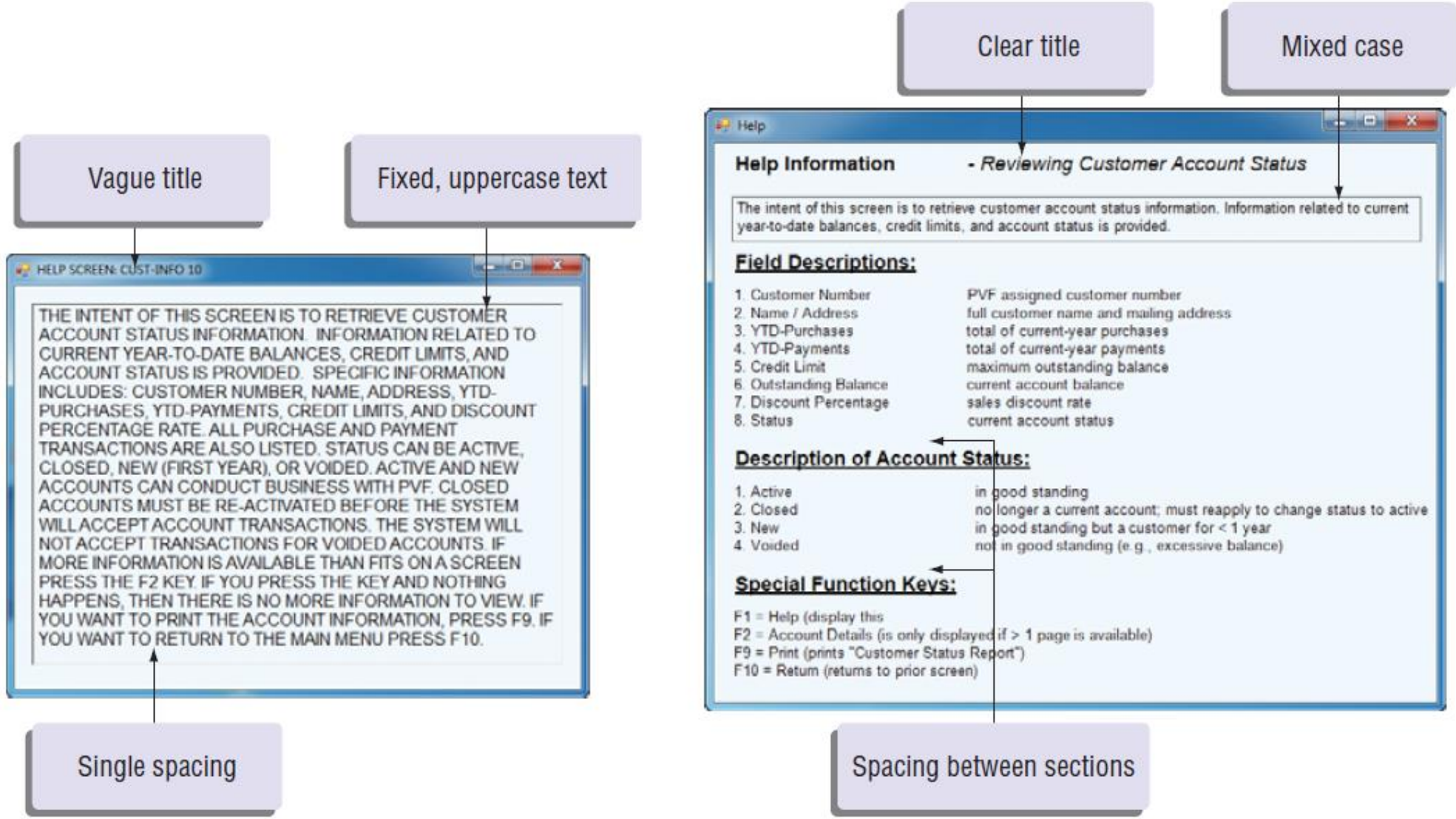


FIGURE 8-7
Contrasting two help screens from an application system at PVF: (A) A poorly designed help screen with many violations of the general guidelines for displaying text, (B) An improved design for a help screen.

TABLE 8-5: General Guidelines for Displaying Tables and Lists

Guideline	Description
Use meaningful labels	All columns and rows should have meaningful labels.
	Labels should be separated from other information by using highlighting.
	Redisplay labels when the data extend beyond a single screen or page.
Format columns, rows, and text	Sort in a meaningful order (e.g., ascending, descending, or alphabetical).
	Place a blank line between every five rows in long columns.
	Similar information displayed in multiple columns should be sorted vertically (i.e., read from top to bottom, not left to right).
	Columns should have at least two spaces between them.
	Allow white space on printed reports for user to write notes.
	Use a single typeface, except for emphasis.
	Use same family of typefaces within and across displays and reports.
	Avoid overly fancy fonts.
Format numeric, textual, and alphanumeric data	Right-justify *numeric data* and align columns by decimal points or other delimiter.
	Left-justify *textual data*. Use short line length, usually 30 to 40 characters per line (this guideline is what newspapers use, and it is easy to speed-read).
	Break long sequences of *alphanumeric data* into small groups of three to four characters each.

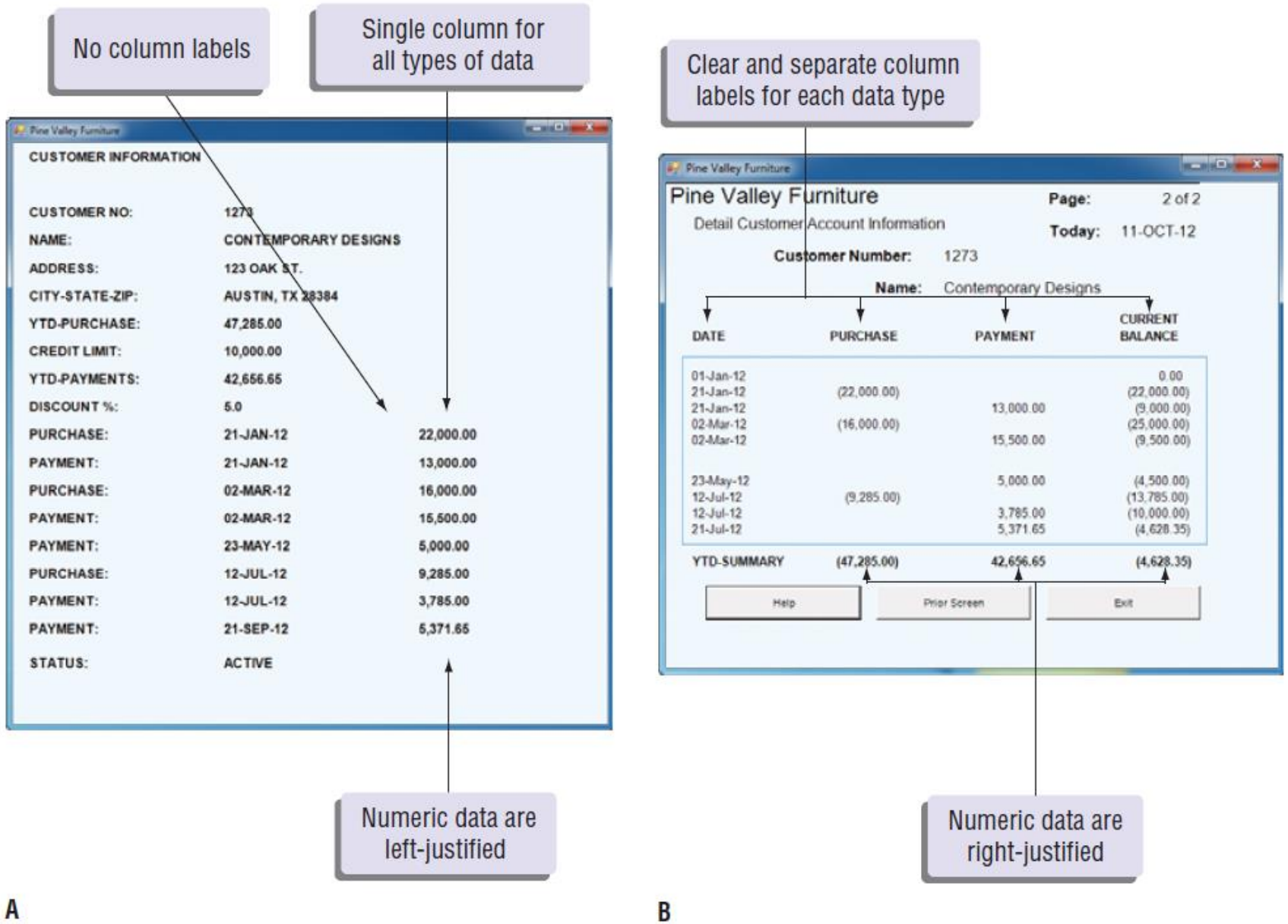


FIGURE 8-8
Contrasting two Pine Valley Furniture forms: (A) A poorly designed form, (B) An improved design form.

Place meaningful labels on all columns and rows

Alphabetic text is left-justified

Use a meaningful title

Box the table data to improve the appearance of the table

Pine Valley Furniture

Salesperson Annual Summary Report, 2012

January 10, 2013 Page 1 of 2

Region	Salesperson	SSN	Quarterly Actual Sales			
			First	Second	Third	Fourth
Northwest & Mountain						
	Baker	999-99-9999	195,000	146,000	133,000	120,000
	Hawthorne	999-99-9999	220,000	175,000	213,000	198,000
	Hodges	999-99-9999	110,000	95,000	170,000	120,000
Midwest & Mid-Atlantic						
	Franklin	999-99-9999	110,000	120,000	170,000	90,000
	Stephenson[1]	999-99-9999	75,000	66,000	80,000	80,000
	Swenson	999-99-9999	110,000	98,000	100,000	90,000
New England						
	Brightman	999-99-9999	250,000	280,000	260,000	330,000
	Kennedy	999-99-9999	310,000	190,000	270,000	280,000

[1]Sales reflect May 1, 2012–December 31, 2012.

Superscript characters can be used to alert reader of more detailed information

Sort columns in some meaningful order (names are sorted alphabetically within region)

Long sequence of alphanumeric data is grouped into smaller segments

Right-justify all numeric data

Try to fit table onto a single page to help in making comparisons

FIGURE 8-9
Tabular report illustrating good report design guidelines.