Unit -6
System Architecture Development

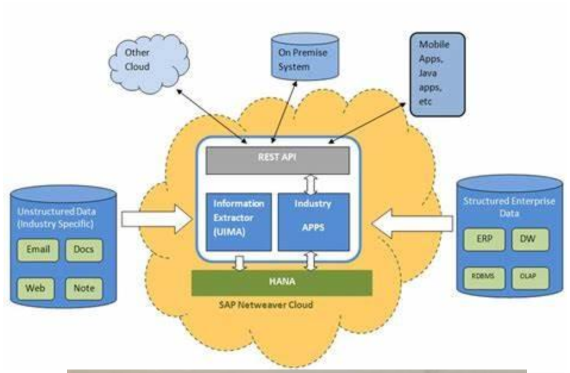
6.1 Introduction to system Architecture Development.

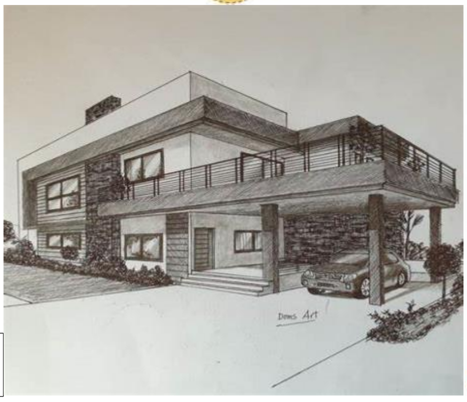
Architecture is a process and the product of planning designing and constructing building or any thing else structure. A good design can make a structure survive and be and admired for year. It is a task of joining(integrating) both art and science to make sure all the component come together in a good solution.

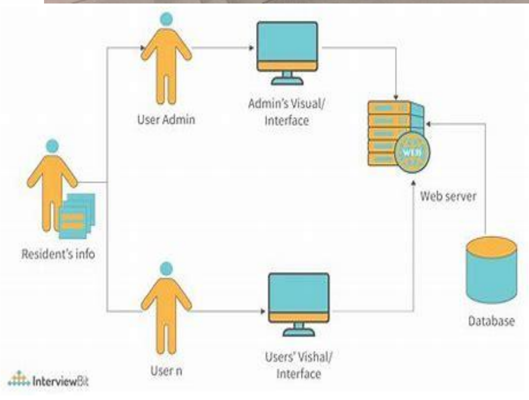
"Architecture is about the important stuff, What ever that is" **Ralph Johnson**

Important stuff means= In system Architecture we focus more on the structure more then implementation details.

" System/Software architecture is the structure of the S/W like blue print of building"
Let's analyze some architecture.


Other Cloud
On Premise System
Mobile Apps, Java apps, etc
REST API
Information Extractor (UIMA)
Industry APPS
HANA
SAP Netweaver Cloud
Unstructured Data (Industry Specific)
Email
Docs
Web
Note
Structured Enterprise Data
ERP
DW
RDBMS
OLAP



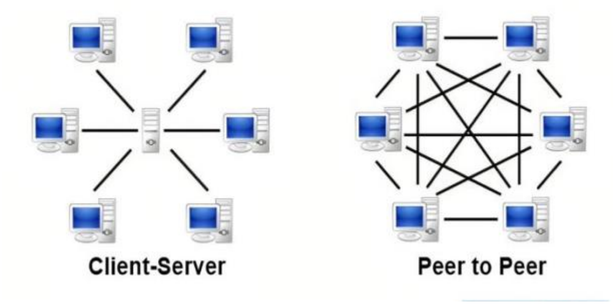


There are different models or approaches to design and develop a system architecture, depending on the nature, size, and complexity of the system.Some of the common models are the **waterfall model**, the **agile model**, the **iterative model**, and the **spiral model**.

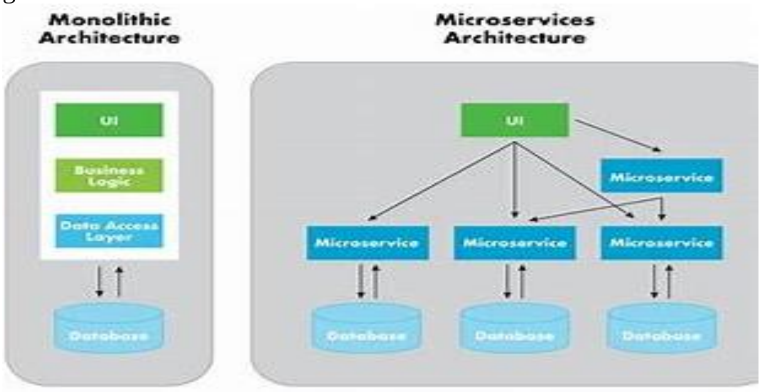
The term "system architecture model" typically refers to a conceptual or visual representation of the structure, components, and interactions within a system. This model provides a high-level overview of how different elements of a system work together to achieve certain objectives.

System architecture models are frameworks used to define and describe the structure, behaviour, and interaction of a system.Some common types of system architecture models are:

1. **Client -server architecture:**In this model, the system is divided into client and server components, where clients request services or resources from servers. This architecture enables scalability, centralization of data, and distributed processing. Variants include two-tier, three-tier, and n-tier architectures.
2. **Peer-to-Peer (P2P)** Architecture: P2P architecture distributes the workload and resources among interconnected nodes, allowing each node to act as both a client and a server. P2P systems facilitate decentralized communication, fault tolerance, and scalability



3. **Micro services Architecture:** Micro services architecture decomposes a system into a collection of loosely coupled, independently deploy able services. Each service is focused on a specific business capability and communicates with other services through lightweight protocols. This promotes flexibility, scalability, and ease of maintenance.
4. **Monolithic Architecture**: In contrast to micro services architecture, monolithic architecture consolidates all functionality into a single, self-contained application. While simpler to develop and deploy initially, monolithic architectures can become unwieldy and difficult to scale as the system grows.



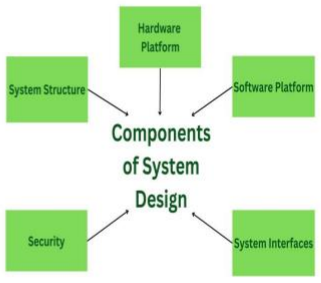
5. **Service-Oriented Architecture (SOA)**: SOA is an architectural style where software components are organized as services that can be accessed and reused across different applications and platforms. Services are loosely coupled and communicate via standardized protocols, enabling interoperability and agility.
6. **Event-Driven Architecture (EDA)**: In an event-driven architecture, system components communicate through the exchange of events. Events represent significant changes or occurrences within the system, triggering appropriate actions or responses. EDA enables asynchronous communication, scalability, and responsiveness to changing conditions.
7. **Component-Based Architecture:** Component-based architecture emphasizes the development and integration of reusable software components, which encapsulate specific functionality and can be assembled to create larger systems. This promotes code re usability, maintainability, and flexibility.
8. **Peer-to-Peer (P2P)** Architecture: P2P architecture distributes the workload and resources among interconnected nodes, allowing each node to act as both a client and a server. P2P systems facilitate decentralized communication, fault tolerance, and scalability.
9. **Hybrid Architecture**: Hybrid architectures combine elements from multiple architectural styles to meet specific requirements or address diverse use cases. For example, a hybrid cloud architecture may integrate on-premises infrastructure with public and private cloud services to optimize performance, cost, and security.

Components consider when designing the architecture of a system

It is the process of making high-level decisions about the organization of a system, including the selection of hardware and software components, the design of interfaces, and the overall system structure.

In order to design a good system architecture, it is important to consider all these components and to make decisions based on the specific requirements and constraints of the system.

Major Component of System architecture are:



Hardware Platform: Hardware platform includes the physical components of the system such as servers, storage devices, and network infrastructure.
Software Platform: Software platform includes the operating system, application servers, and other software components that run on the hardware.
System interfaces: System interfaces include the APIs and user interfaces used to interact with the system.
System Structure: System structure includes the overall organization of the system, including the relationship between different components and how they interact with each other.
Security: Security is an important aspect of system architecture. It must be designed to protect the system and its users from malicious attacks and unauthorized access.

Development of System architecture
The development of system architecture involves designing the overall structure and components of a system to meet specific requirements and objectives. This process typically includes several key steps:

Requirements Gathering: Understanding the needs and objectives of the system is the first step. This involves gathering requirements from stakeholders, users, and other relevant parties to define what the system should accomplish.

Analysis: Once requirements are gathered, they need to be analyzed to identify patterns, commonalities, and potential conflicts. This analysis helps in shaping the architecture to ensure that it effectively addresses all requirements.

Design: Based on the analysis, the system architecture is designed. This includes determining the overall structure of the system, defining components and their interactions, and specifying interfaces between different parts of the system.

Prototyping: In many cases, it's beneficial to create prototypes or proof-of-concepts to validate the design and ensure that it meets requirements. Prototyping helps in identifying potential issues early in the development process.

Implementation: Once the architecture is finalized, the system is implemented according to the design. This involves developing software components, configuring hardware, and integrating various parts of the system.

Testing: Testing is a crucial phase in system development to verify that the implemented system functions correctly and meets all requirements. This includes unit testing, integration testing, system testing, and user acceptance testing.

Deployment: After testing, the system is deployed into the production environment. This may involve installation, configuration, and migration of data from existing systems.

Maintenance and Evolution: Once the system is deployed, it requires ongoing maintenance and support to ensure its continued operation. Additionally, as requirements change or new features are requested, the system architecture may need to evolve over time.

Throughout the development process, it's important to consider various factors such as scalability, reliability, security, and performance to ensure that the system architecture meets both current and future needs.

Interface Design

Interface is the front end application view to which user interacts with the software or hardware. It determines how commands are given to the computer and how data is displayed on the screen.

Interface design is the process of creating a user friendly and intuitive interface for software, websites, or applications. It involves designing the layout, navigation, and visual elements to ensure a seamless user experience.

Significant of Interface Design

1. Attractive
2. User friendly
3. Simple to use
4. Easy to understand
5. **Convinenent for user**

Major importance of ID

1. Enhances usability
2. Boots engagement
3. Increase Adoption

What are the benefits of implementation of interface design?