Anushka

★ MODULE 1:

- Distributed system: Collection of independent (autonomous) computers that appears to its users as a single coherent system.
- Network is the backbone as devices interconnected over the network.
- Heterogeneous systems: Dif devices, hardware, applications.
- Two aspects:
  - Independent computing elements
  - single system ⇒ Middleware: Queries get converted to format acceptable globally. Eases communication among dif systems over the network. Eg: JBDC drivers, APIs, etc
- See architecture diagram.
- Goals of distributed systems:
  - Making resources accessible: Access remote resources and share them in controlled and efficient way. Eg: printers, computers, storage facilities, data, files, web page. Clients are in bulk, so one server cannot handle, thus distributed system scales data, replicates it so that multiple devices can access. To remove system degradation due to bottleneck. The access is made efficient.
  - Distribution transparency: to hide the fact that its processes and resources are physically distributed across multiple computers. Types:

(1) Access transparency: Hide differences in data respresentation and how a resource is accessed. Eg: Dropbox - users can upload, download, or delete files without needing to understand the underlying infractructure like servers, replications or backup servers. It is basically using the functionalities available but not knowing underlying

Teacher's Signature:.....

structure and implementation.

(2) Location Transparency: Hide where resource is located. Different servers communicate with each other to process a query and build a path ~~from~~ but don't know where the devices are located. Eg: GPS navigation app.

(3) Relocation Transparency: Hide that a resource may be moved to another location while in use without affecting functionality of the system. Resources may be needed to transferred from one location to another. Eg: Smartphones - switch between Wifi networks seamlessly.

(4) Migration Transparency: Hide ~~that~~ that data or processes may be moved to another location without disrupting normal operation. Eg: Email service, upgrading smartphone - features like contacts are migrated from old to new phone.

(5) Replication Transparency: Hide that a resource is replicated Eg: Cloud-based note-taking app, email replication transparency. Dispersing data at different server locations while maintening data integrity. Replicas should be updated.

(6) Concurrency Transparency: Hide that a resource may be shared by several competitive users. Eg: Collaborative Document Editing, Shared Shopping List App.

(7) Failure Transparency: Hide the failure and recovery of a resource. Even if a server crashes, it will hide that, transfer query to dif server with same data and process the query. Eg: Online Banking System.
If one service is failing, others are still running.

→ Types of DC systems:

(1) Cluster Computing:

> Underlying hardware consists of a collection of similar workstations or PCs closely connected by means of a high ~~school~~ speed local-area network.

Teacher's Signature:.....

> Each node runs the same operating system.
> Request is diverted to required cluster which will process it
> Eg: Event Registration System
> Architecture diagram: Master node divides tasks and communicates with other machines. Problem if master fails, same category clubbing puts certain restrictions
> Characteristics:
- Proximity: Club similar (same processing power/applications) things/categories together. Group of interconnected nodes that are physically close to each other often within the same data centre / building
- Tightly coupled: Share common infrastructures, high-speed interconnect.
- Single administrative domain: Controlled by a centralized entity.
- Homogeneous resources: Same hardware & software components.

(2) Grid Computing
> Involves the coordinated use of resources from multiple networked computers to solve a problem or perform a task.
> Focus is on sharing computing power and resources across different administrative domains to achieve a common goal
> Resources include heterogeneous machines in terms of processing power, storage capacity and specialized software apps.
> Eg: Climate research project - To derive climate patterns, like give same project to dif groups for utilizing their individual advantages.
> Characteristics:
- Geographical Distribution: It involves use of distributed resources which are geographically dispersed, many of

Teacher's Signature:.....

them belong to dif organizations or institutions.

- Loosely coupled: Nodes may have dif hardware & software environments. You can add resources with change in requirement.
- Multiple admin domains: Often owned and controlled independently by dif organisations.
- Heterogeneous resources

(Real world eg of DC) – Cloud Computing Architecture: Three services available. Third party system provides their services for end-user to use. Eg: Google Doc.

– Distributed Information Systems:

> Network of interconnected and independent components or nodes that work together to achieve a common goal of managing, processing and disseminating info.

> Eg: Transaction Processing System

> A transaction is collection of operations on the state of object that satisfies ACID properties

- Atomicity: If you perform operations, it should be completed 100% or not at all.
- Consistency: Data integrity, data should be in valid state after the transaction.
- Isolation: Avoiding simultaneous write operations, do not interfere with other transactions.
- Durability: Changes should be updated regularly.

> A ~~TSP~~ TPS processes and manages transactions of an organisation such as sales, purchases and inventory updates.

> The processing and storage of transactional data are spread across multiple nodes.

> See architecture diagram, of TPS using TP manager, which fails.

> TP manager replaced with middleware for industrial applications.

Teacher's Signature:.....

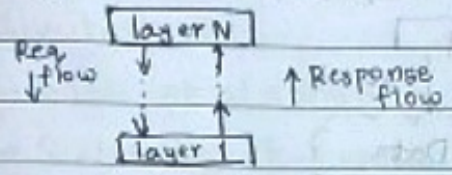
8

- Distributed Computing vs Distributed info systems
> Computing focuses on distributing computation tasks across multiple nodes or machines to solve a complex problem
• Goal is to achieve parallel processing & improved performance
• Deals with execution of algorithms, programs or processes
• Distributed processing of large datasets, parallel & grid computing. Eg: Apache Hadoop
> Information: focuses on distributing & managing info or data across multiple nodes / locations.
• Deals with storage, retrieval & management of data.
• Distributed databases, content delivery networks (CDNs) and distributed file systems.
• Eg: Amazon DynamoDB
- Distributed Pervasive system: (something available to you in real-word scenario)
• Refer to computing environments where computing devices are seamlessly integrated into the physical world and interact with users & other devices in pervasive manner.
• Involve the use of sensors, actuators & communication technologies.
• Eg: Smart home systems that control lighting, etc.
• Respond to user preferences, environmental conditions and security events
• Industrial IOT:
>Eg: Manufacting plants or industrial facilities where sensors and connected devices are distributed across production line.
• Healthcare systems:
>Eg. Remote patient monitoring that use wearable sensors to collect vital signs and health data.
• Smart Cities: Intelligent transportation systems.

Teacher's Signature:.....

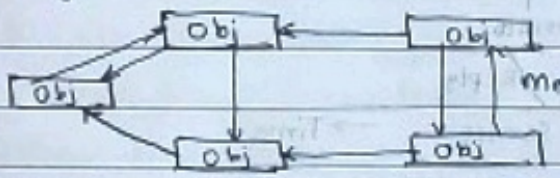
- Architectural Styles : to organize into logically different components and distribute those components over the various machines.

> Layered Architecture : Used for client-server systems.



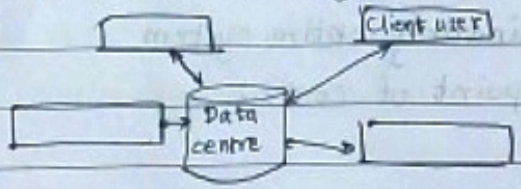
Eg: OSI model

> Object Based Architecture : Each object corresponds to what we have defined as component. These are connected through RPC mechanism.



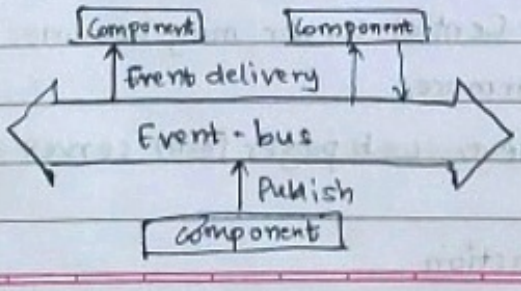
Eg: Social media apps : Objs ⇒ posts, comments, etc

> Data-centered Architecture : Processes communicate through a common (active passive) repository. Places a strong emphasis on data and the way it is stored, accessed and managed.



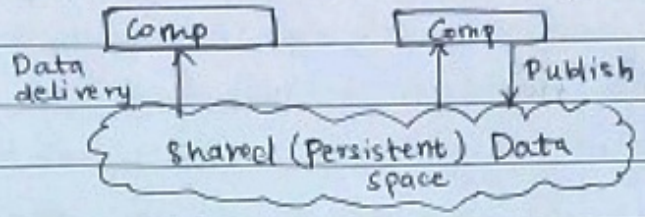
Eg: Dropbox, Google drive

> Event-based : Processes communicate through propogation of events which optionally also carry data. Middleware ensures only those processes that subscribed to those events will receive them. Processes are loosely ~~coup~~ coupled.



Teacher's Signature:.....

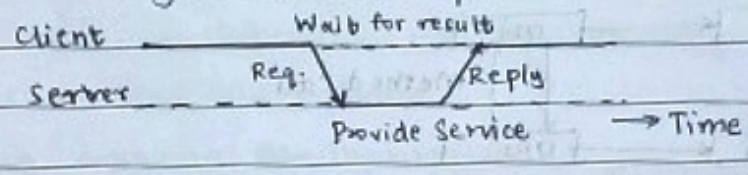
> Shared Data Space Architecture: Event based + Data centered
Processes are decoupled in time, i.e, they need not both be active when communication takes place. Increases flexibility.



Eg: Splitwise app

System Architectures

> Client-Server Architecture: Efficient unless & until msg does not get lost or computed.



- If msg gets lost, client will have to send same req again & again with increase in waiting time.

> Centralized Architecture:

- Design where one central node or server is responsible for managing and coordinating entire system.
- Central node acts as point of control
- Eg: Traditional DBMS.
- Advantages:
  - → Simple
  - → Centralized control over data consistency & integrity
- Challenges:
  - → Single point of failure
  - → Scalability issues: Central server may become bottleneck limiting overall performace.
- Eg: Web app to retrieve web pages from server
  - → User req
  - ⇄ Client-Server interaction

Teacher's Signature:.....

→ Server Processing
→ Data Retrieval
→ Response to client
→ Display on client

- Limitations:

→ Single pt of failure
→ Scalability challenges
→ Performance bottlenecks
→ High Network Dependency
→ Limited Fault Tolerance
→ Data Security concerns
→ Maintenance challenges
→ Geographical Dependency.
→ Resource utilization

> Application Layering : See architecture diagram
> Multi-layered Architecture : See architecture diagram
Eg: Any web-based app
> Decentralized Architectures:

- Involves distributing tasks and responsibilities across multiple nodes, avoiding reliance on a central authority
- Eg: Blockchain
- Nodes communicate with each other in a peer to peer fashion
- Structured peer-to-peer architecture. Nodes are organised in a structured manner to efficiently locate and retrieve info or resources. Each node has specific responsibility and there is a clear structure that guides how nodes are connected and how data is distributed across the network.

Eg: Chord.
→ Hash tables are used in this type of architecture

Teacher's Signature:

- Unstructered peer-to-peer architecture: Nodes in the network are not organised according to a specific structure.
→ Unstructured P2P networks are more flexible and dynamic, making them suitable for scenarios where nodes join ~~and~~

Teacher's Signature:.....

> Hybrid Architecture:

- Systems refer to a combination of different architectural styles or approaches within the same system.
- To achieve balance between efficiency, scalability, fault tolerance and other desired characteristics.
- Eg: Amazon Web Service (AWS), Retail Hybrid Architecture
- Customer can have a unified shopping experience, whether making purchase online or in-store while the business efficiently manages inventory and sales data across both channels.

> Edge Server System:

- Refers to the ~~develop~~ deployment of servers at the edge of a network, closer to end-users or devices, to reduce latency, improve performance, and enhance overall efficiency.
- A paradigm that brings computing resources closer to where data is generated or consumed.
- Eg: Content Delivery Networks (CDNs)
- CDN uses edge servers to deliver that content from a server located

> Collaborative Distributed System:

- Involves multiple entities or components working together to achieve a common goal, often in a decentralized manner.
- Emphasizes cooperation, information sharing and coordination, ~~etc~~

> Middleware:

- Software that provides common services and ~~appli~~ capabilities to facilitate communication & interaction among dif components or applications in distributed environment.

Teacher's Signature:.....

- Intermediary layer between the various distributed components, abstracting away the complexities of the underlying network and infrastructure.
- Addresses challenges like transparency, scalability and heterogeneity.
- Eg: RPC frameworks (DCOM, CORBA, Java RMI), msg-oriented middleware (eg: JMS, AMQP) and web services (REST)
- Middleware Functions:

(1) Communication Abstraction: Abstracts the communication details between dif components which helps in achieving interoperability.

(2) Heterogeneity Handling: Distributed systems often involve components running on dif platforms. Middleware provides a layer of abstraction that enables interoperability between heterogeneous components.

(3) Transparency: By hiding the complexity of the underlying distributed infrastructure. This includes location and access transparency

(4) Security: May involve encryption, authentication and authorization mechanisms.

(5) Scalability: By providing mechanisms for load balancing, dynamic resource allocation

(6) Naming and Directory Services: Provides naming and directory services to enable components to locate each other in a distributed environment.

(7) Event Handling: May offer event-driven communication mechanisms to facilitate the exchange of events or messages between distributed components. Useful for responsive and loosely coupled systems.

Teacher's Signature:.....

> Models of middleware: Can be categorized into dif models on the services it provides and the way it facilitates communication.

- MOM: Msg oriented middleware
  - Involves the passing of data between applications using a communication channel that carries self-contained units of info (msgs).
  - Msges are sent and received asynchronously.
  - See architecture diagram.
  - Uses an API to communicate through a msging client that is provided by the mom ~~valuer~~ vendor.
  - Applications are abstractly decoupled, senders and receivers are never aware of each other.
  - Characteristics:
    - → Communication is based on message passing.
    - → Supports publish/subscribe and pt to pt communication models
    - → Eg: JMS, AMQP, IBM MQ.
  - Limitations:
    - → Msg ordering: Challenge is guaranteeing strict msg ordering ~~en~~ which can be difficult in mom, especially in presence of network delays, failures or parallel processing of msgs.
    - → Latency: The use of asynchronous messaging in mom introduces a degree of latency due to the decoupling of msg senders and receivers.
    - → Complexity: mom systems can become complex when the no of distributed components and volume of msg increase. Managing the config, addressing and msg formats across a distributed system can be challenging.
    - → Scalability concern: Managing a large no of subscribers and publishers can lead to scalability challenges.

Teacher's Signature:.....

→ Guaranteed Delivery Overhead:

- ORB: Object Request Broker
  - Enables communication between distributed objects in a distributed computing environment.
  - It allows objects to request and invoke services from other objects in a network, regardless of the programming languages or platforms.
  - Eg: CORBA (Common ORB Architecture)
  - See architecture diagram
  - Limitations:
    - → Performance overhead. Marshalling & Unmarshalling will take time for large datasets.
    - → Deployment challenges: Deploying & configuring ORBs across dif platforms & environments.
    - → Network Overhead: Communication between objects involves network communication which can introduce latency.
    - → Resource utilization: Memory needed for maintaining object references, network bandwidth and communication.

★ DESIGN ISSUES AND CHALLENGES IN DISTRIBUTED APPLICATIONS:

- Communication: Involves designing appropriate mechanisms for communication among the processes in the network. Some mechanisms are RPC, RMI, ROI, etc. Eg: E-commerce
- Processes: Management of processes and threads at client/server; code migration.
  Eg: Ride Sharing system with components UI interface, location tracking, ride assignment, billing service.
  (1) Coordination and consistency: Eg: 2 rides accept same client
  (2) Load balancing: Optimized distribution of nodes is required.

(3) Fault tolerance : If one server fails, other nearby server should handle.
(4) Communication overhead : Delay due to multiple clients
(5) Scalability

- Naming service: Robust schemes for names, identifiers and addresses is essential for locating resources and processes in a transparent and scalable manner. Naming in mobile systems provides additional challenges bcoz naming cannot be easily tied to any static geographical topology.
  Eg: Cloud based documentation where users create and update documents simultaneously. Parameters.
  (1) Dynamic IP addresses
  (2) Service discovery
  (3) Consistency & Replication
  (4) Scalability
  (5) Security & Authentication
- Synchronization: Synchronization or coordination among the processes are essential. Mutual exclusion is a classical example of synchronization. Synchronizing physical clocks and devising logical clocks capture the essence of the passage of time.
- Data storage & access: Schemes for data storage and implicitly for accessing the data in a fast and scalable manner.
- Security: Aspects of cryptography, secure channels, access controls.
- Consistency and Replication: To avoid bottlenecks, provide fast access to data and provide scalability.
- Fault tolerance : requires maintaining correct and efficient operation inspite of any failures of links, nodes and processes.
- Scalability & modularity: Various techniques such as replication, caching and cache management and asynchronous processing help to achieve scalability.
- Transparency: Ability to relocate, replication, concurrency & failure transparency.

Teacher's Signature:.....

- DC Models:

(1) Minicomputer model: Extension of centralized time-sharing system. See diagram.

(2) Workstation model: Several workstations connected by communication network. Eg: Campus offices

(3) Workstation-server model: Network of personal workstations, each with its own serves to form a distributed system. Perform some processing tasks locally, while more intensive or centralized tasks are offloaded to server.

(4) Processor-pool model: Processors are pooled together to be shared by the users as they needed.

(5) Hybrid Model: Combination of dif computing paradigms or architectures to address specific requirements or challenges.

- Study hardware & software components

Teacher's Signature:.....

✱ Communication

> OSI MODEL

(1) Physical Layer: Deals with physical connection between devices, defining hardware specifications.

- Summary of sending an Email via OSI model: When you send an email, the OSI model is at work. The physical layer involves your computer and network hardware, the data link layer manages communication with local router, the network layer routes the email data across internet, transport layer ensures reliable email delivery, session layer establishes secure connection, presentation layer encrypts & encodes data and application layer encompasses your e mail client.

> RPC (Remote Procedure Call): Simplifies communication between programs running on different machines, making it as seamless as if they were right next to each other. See diagram.

- Client: Initializes RPC
- Stub (Client-side and serverside): Stubs are intermediary components that handle the communication between client & server.
- Client-side stub: Takes client's request & packages it into a format suitable for transmission over network.
- Server-side stub: Receives the message, unpacks it and invokes actual procedure or function.
- Marshalling / Unmarshalling: Converting data into format suitable for transmission over network.
- Transport Layer: Actual transmission of data between client and server.
- Servers: Program that hosts the actual procedure or service.

Teacher's Signature:.....

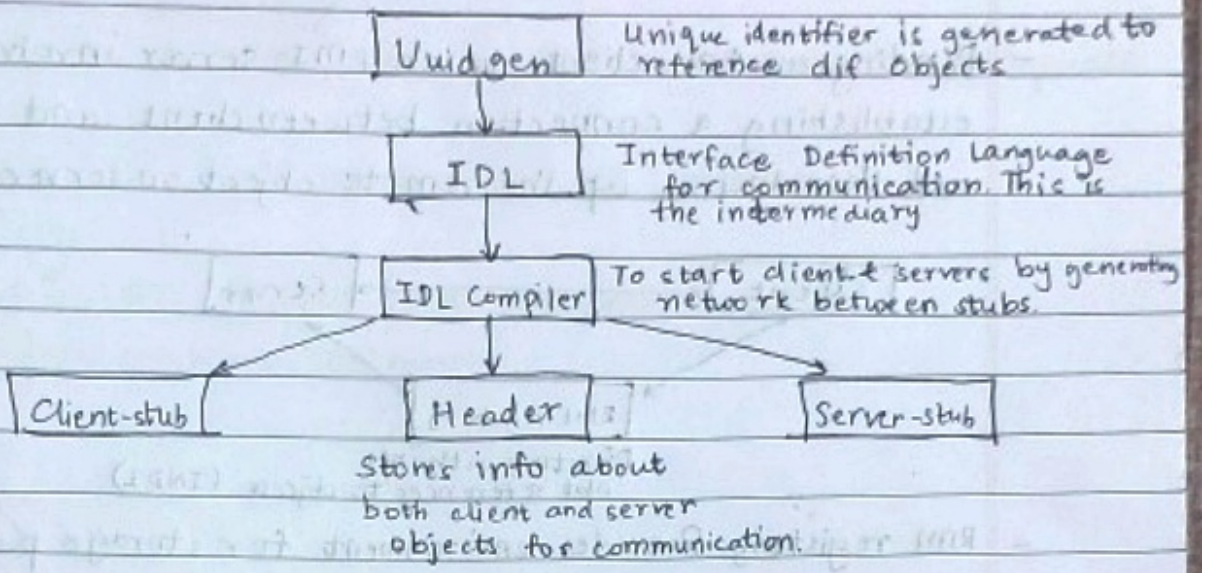
> RPC Using Parameter Passing:
Mechanism by which data is sent from the client to server, allowing the server to execute the requested procedure with the provided parameters. See diagram.

> RPC (Limitations):
- Latency & Performance: Impacts responciveness of the application, ~~data~~ leading to delays in updating the database
- Failure Handling: If server or network fails during RPC, client might not get a response.
- Security Concern: Sensitive data over the network during an RPC raises security concerns.
- Synchronous nature: Scenarios where server is performing resource-intensive tasks, synchronous nature can lead to delays.
- Compatibility issues: Dif platforms or programming languages may interpret data formats differently.
- Limited Language Support: Challenging to integrate RPC into applications written in dif. languages.

> Extended RPC: Asynchronous approach : DCE RPC
- Client sends a request to a server & doesn't wait for an immediate response.
- Client is free to perform its own tasks after the request is accepted.
- So the client does not become idle in asynchronous communication.

> See DCE RPC diagram and importance of all components like Unique ID gen, header, IDL compiler, IDL files, stubs.

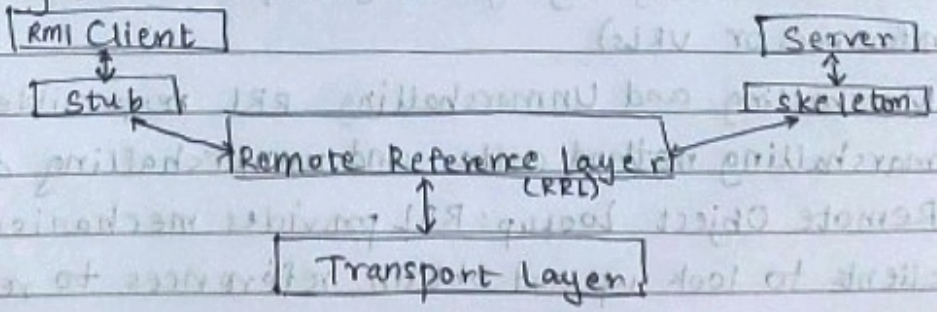
Teacher's Signature:.....



- See client-server binding diagram.

> Remote Object Invocation:

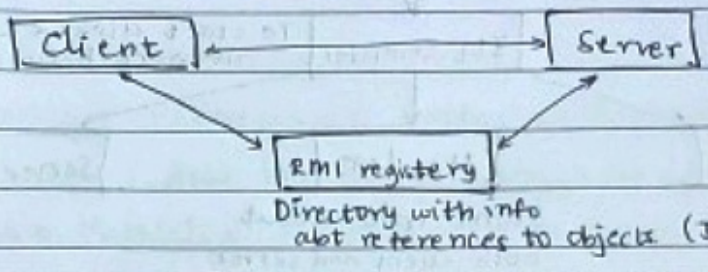
- Process of invoking methods or functions on objects that reside in a dif address space or remote machine.
- Its a mechanism that allows you to call methods on objects that are not co-located with the calling code.
- Eg. RMI: Mobile banking on mobile, when you log in & check balance, RMI allows client to interact with objects (account data) present on remote serve.
- RMI is a mechanism in Java that allows a Java program to invoke methods on objects that reside in a different JVM, typically on remote serve.



RMI Architecture

Teacher's Signature:.....

- Binding an RMI client to an RMI server involves establishing a connection between client and server and then looking up the remote object on server side.



- RMI registery: Provides environment for storage purpose.
- Remote Interface: Start by defining interfaces for the remote objects that you want to access remotely.
- Interfaces should extend the "java.rmi.Remote" interface.
- Remote Reference Layer:
  - Manages the communication between objects that reside in a different address spaces or on dif machines.
  - Responcible for handling the references to remote objects.
  - It is a sort of middleware which facilitates all communication between client and server throughout the entire process.
- Object Identifier:
  - RRL provides mechanisms for uniquely indentifying remote objects within a distributed system.
  - Involves assigning globally unique identifiers (like obj references or URLs)
- Marshalling and Unmarshalling: RRL responsible for marshalling method calls and unmarshalling data.
- Remote Object Lookup: RRL provides mechanisms for clients to look up and obtain references to remote objects.
- Stubs and Skeletons: Stubs act as proxies for remote objects. Skeleton receives incoming method invocation,

Teacher's Signature:.....

unmarchal them ~~data~~ and dispath them ~~service~~ to appropriate remote objects.

- Static RMI : Predefined and fixed interfaces.
  Eg: Contact list
- Dynamic RMI : Updation in Real time
  Eg: Stock prices, e-commerce.

Teacher's Signature:.....