CLIENT/SERVER TECHNOLOGY AND WEB SERVICES

MODULE 5

PREPARED BY,

ANAGHA PRADEEP SNGIST

SYLLABUS

- WEB SERVICES HISTORY. WEB SERVER TECHNOLOGY- WEB SERVER, WEB SERVER COMMUNICATION, ROLE OF JAVA FOR CLIENT/SERVER ON WEB. WEB SERVICES- MICROSERVICES, APIS, API GATEWAY, AUTHENTICATION OF USERS/CLIENTS, TOKENS/KEYS FOR AUTHENTICATION, SERVICE MESH, MESSAGE QUEUES, SAAS, WEB SOCKETS.
- CLIENT/SERVER/BROWSER – SERVER TECHNOLOGY, CLIENT/SERVER TECHNOLOGY AND WEB APPLICATIONS, BALANCED COMPUTING AND THE SERVER’S CHANGING ROLE. THIN CLIENT COMPUTING - COMPUTING MODELS-COMPARISON-COMPUTING ENVIRONMENT. FUTURE OF CLIENT/ SERVER COMPUTING ENABLING TECHNOLOGIES, TRANSFORMATIONAL SYSTEM.

WEB SERVICES HISTORY

- The world wide web was developed by **Tim Berners-Lee** for his employer CERN or the European organization for nuclear research between 1989-1991.
- In 1990, he wrote the program for the **World Wide Web**. This program created the first web browser and HTML editor.
- It was the first program to use both **FTP and HTTP.**
- FTP (file transfer protocol) is used **to transfer data over a network**.
- Protocol is the **set of standards** that defines and controls connection, communication, and the transfer of data between two computer endpoints.
- It determines the **format and defines the terms of transmission**.
- HTTP is the protocol that supports **hyper-text documents.**
- Both of these protocols are necessary for communication over the internet or world wide web.

- The source code for the **World Wide Web** was made **public in 1993**, making it available to everyone with a computer.
- The technology continued to develop and between 1991-1994, extended from communication only between **scientific organizations, to universities** and, finally, to **industry.**
- By 1994, **computers could transfer data between each other** through a cable linking ports across various operating systems.
- The **first web server**, also written by Berners-Lee, **ran on Nextstep**, the operating system for NEXT computers.
- The other technology authored by Berners-Lee that is required for web communication is **URLS** (universal resource locators).

- These are the **Uniform Global Identifiers** for documents on the web allowing **for easily locating** them on the web.
- Berners-lee is also responsible for writing the **initial specifications for html**.
- The **First Web Server** was installed in the united states on December 12, 1991 and at SLAC (Stanford linear accelerator center), which is a U.S. Department of energy laboratory.
- In 1994, Berners-Lee created the **world wide web consortium (WWC)** to regulate and standardize the various technologies required for web construction and communication.
- It was created **to ensure compatibility between vendors or industry members** by having them agree on certain core standards.
- This ensures the ability for web pages to be intelligible between different operating systems and software packages.
- After 2000, the web exploded. Till date, there exist more than 110 million web sites on the world wide web.

WEB SERVER TECHNOLOGY

- The process for web communication works as follows: a computer **runs a web browser** that allows it **to request, communicate and display** HTML documents (web pages).
- **Web browsers** are the software applications that allow **users to access and view** these web pages and they run on individual computers.
- The most popular web browsers are internet Explorer, Mozilla, Firefox, And Safari (for MAC).
- After **typing in the URL** (or address) and pressing return, **the request is sent to a server** machine **that runs the web server**.
- The web server is the **program that delivers the files that make up web pages**. Every **web site or computer that creates a web site requires a web server**.
- The most popular web server program is **Apache**. The **server machine** then **returns** the **requested web page**

- Communication over the internet can be broken down into two interested parties: **clients and servers.**
- The machines **providing services** are servers.
- Clients are the machines used to **connect to those services**.
- Internet requires a web server program like Apache to render the search result intelligible in html.
- Web servers translate URL path components in **local file systems**.
- The URL path is dependent on the **server's root directory**. The root directory is the top directory of a file system that usually exists hierarchically as an inverted tree.
- URL paths are similar to UNIX like operating systems.

- The typical client request reads, for example, “http://www.Example.Com/path/file.Html”. This client web browser translates this request through an HTTP request and by connecting to “www.Example.Com”, in this case.
- The web server will then add **the requested path to its root directory path**.
- The result is located in **the server’s local file system or hierarchy of directories**. The **server reads the file and responds** to the browser’s request. The response contains the requested documents, in this case, web sites and the constituent pages.

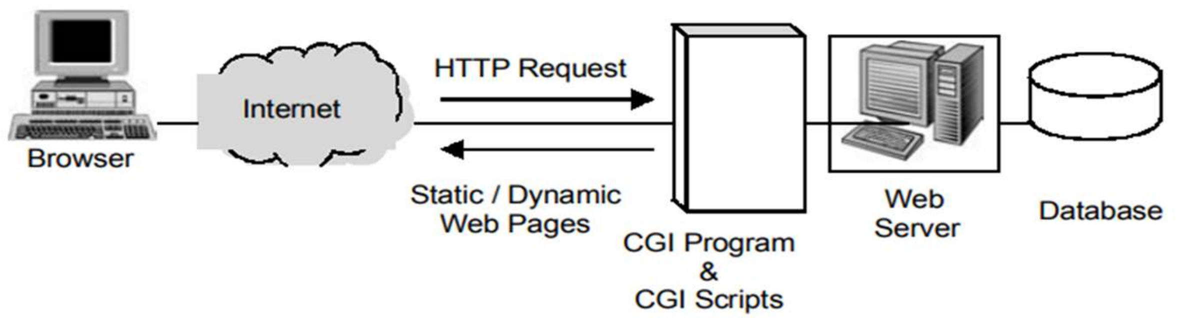


Fig. 8.1(*b*): Web Technology Today

- **WEB BROWSER (WEB CLIENT)**
  - The browser acts on behalf of the user.
  - The browser:
    - Contacts a web server and sends a request for information.
    - Receives the information and then displays it on the user's computer.
  - A browser can be **text-based or graphical** and can make the internet easier to use and more intuitive.
  - A graphical browser allows the user to view images on their computer

- **ACCESSING DATABASE ON THE WEB PAGE**
  - Step1: the user types in a URL or fills out a form or submits a search on a web page and clicks the submit button.
  - Step 2: the browser sends the user's query from the browser to the web server, which passes it on to a cgi script.[A common gateway interface (CGI)]
  - step 3: the CGI script loads a library that lets it talk to an SQL database server, and it uses that library to send SQL commands to the database server.
  - Step 4: the database server executes the sql commands and sends the request information to the cgi script.
  - Step 5: the CGI script generates an html document and writes the html document to the web server.
  - Step 6: the web server sends the HTML page back to the remote user.

- If the user has send some information to update a database.
- Then the CGI script will generate the appropriate SQL commands and send it to the database server.
- The database server will execute the SQL commands and then inform the user about the result of execution.

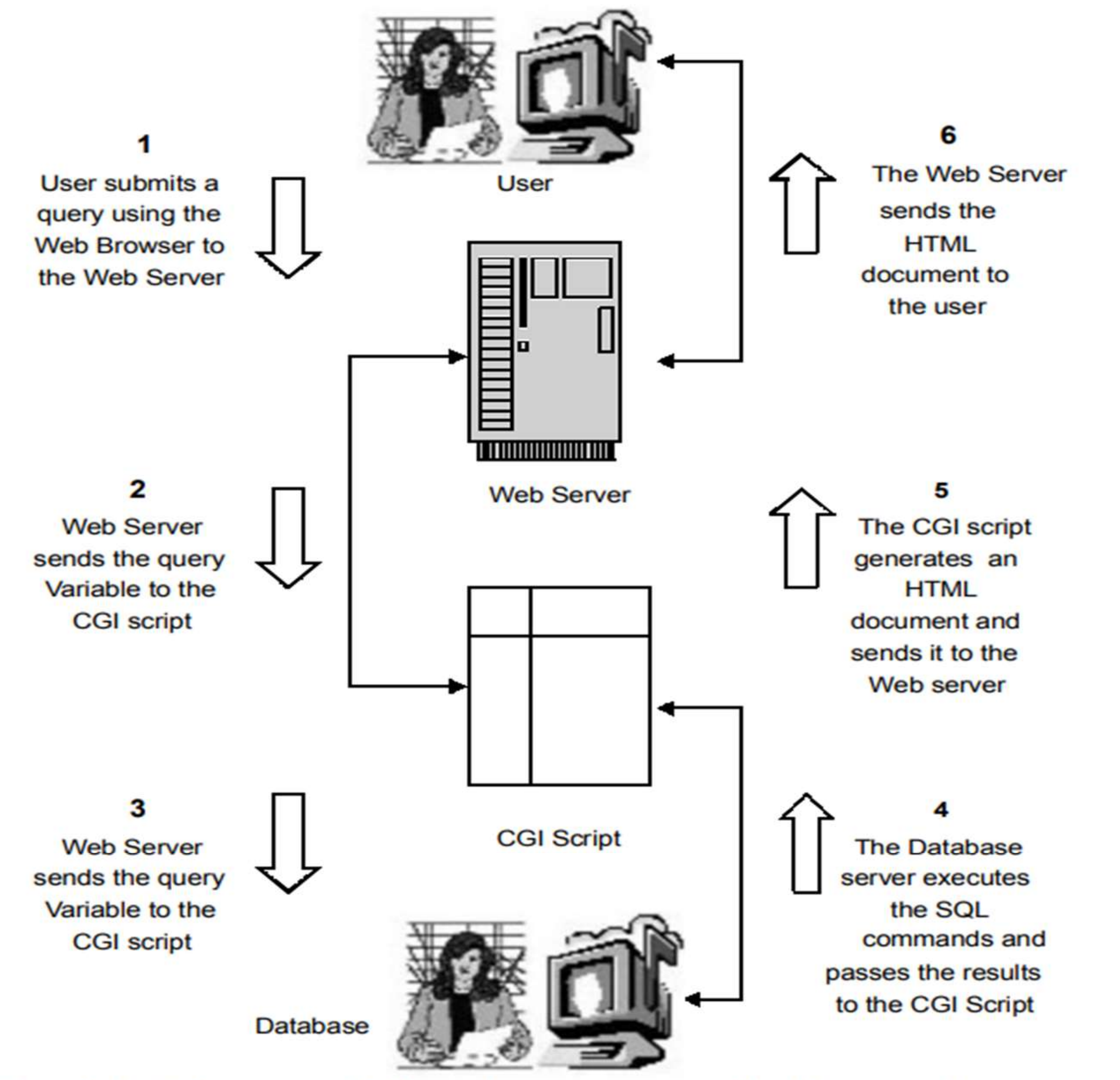

User
1
User submits a query using the Web Browser to the Web Server
6
The Web Server sends the HTML document to the user
Web Server
2
Web Server sends the query Variable to the CGI script
5
The CGI script generates an HTML document and sends it to the Web server
CGI Script
3
Web Server sends the query Variable to the CGI script
4
The Database server executes the SQL commands and passes the results to the CGI Script
Database

Fig. 8.2: Information Exchange between User and a Web-based Database

WEB SERVER

- **HTTP:** Every web server program operates by **accepting HTTP requests** from the client, and **providing an HTTP response** to the client.
- The HTTP response usually consists of an HTML document, but can also be a raw file, an image, or some other type of document (defined by mime-type**s); if some error** is found in client request or while trying to serve the request, a web server has to send an error response which may include some custom HTML or text messages to better explain the problem to end users.
- **Logging:** usually web servers have also the capability of logging some detailed information, **about client requests and server responses, to log files**; this allows the webmaster to collect statistics by running log analyzers on log files.

- In practice many web servers implement the following features also:
  - Authentication, optional authorization request (request of user name and password) before allowing access to some or all kind of resources.
  - Handling of not only static content (file content recorded in server's filesystem(s)) **but of dynamic content** too by supporting one or more related interfaces (SSI, CGI, SCGI, fastcgi, JSP, PHP, ASP, ASP.NET, server API such as NSAPI, ISAPI, etc.).
  - Https support (by ssl or tls) to allow secure (encrypted) connections to the server on the standard port 443 instead of usual port 80.
  - Content compression (i.E., By gzip encoding**) to reduce the size of the responses** (to lower bandwidth usage, etc.).
  - **Virtual hosting** to serve many web sites using one ip address.
  - Large file support to be able to serve files whose size is greater than 2 GB on 32 bit OS.
  - **Bandwidth throttling** to limit the speed of responses in order to not saturate the network and to be able to serve more clients.

- HTTP protocol provides the standardized rules for representing data, authenticating requests, and detecting errors. The purpose of protocols is to make data transfer and services user-friendly.
- Web servers must also be able to **manage static and dynamic content**.
- Static content exists as **a file in a file system**.
- Dynamic content is content (text, images, form fields) on a web page **that changes according to specific contexts or conditions.**
- Dynamic content is produced by some other **program or script** (a user-friendly programming language that connects existing components to execute a specific task) or **API** (application programming interface the web server calls upon).
- It is **much slower to load** than static content since it often has to be pulled from a remote database

- It provides a greater degree of **user interactivity and tailor responses to user requests**.
- To handle dynamic content, **web servers must support** at least any one of **interface like JSP** (java server pages**), PHP** (a programming language that creates dynamic web pages**), ASP** (active server pages) or **ASP.NET** (it is the successor of ASP).

WEB SERVER COMMUNICATION

- **Web servers** are one of **the end points in communication** through the world wide web.
- The **world wide web** is the **global structure** of electronically connected information.
- It refers to the global connections between computers that allow users to search for documents or web pages by requesting results from a web server.
- These documents are **hyper-text based** (written in HTML hypertext markup language), **allowing users to travel to other pages and extend their research** through links.
- They are delivered in a **standardized protocol, HTTP** (hypertext transfer protocol, usually written in lower case letters), making html documents intelligible across hardware and software variations.
- This information travels through **web servers and web browsers**.

- The communication initiates from a user request through a web browser.
- The request is delivered to a web server **in 'HTTP' format**.
- The **server then processes the request**, which can be anything from a general search to a specific task, and returns the results in the same format. The results are written in HTML, which is the language web pages are written in that supports high-speed travel between web pages.
- HTML is also essential for displaying many of the interactive features on web pages, such as linking web pages to other objects, like images.

ROLE OF JAVA FOR CLIENT/SERVER ON WEB

- Java is a programming language that has been developed **specifically for the distributed environment** of the internet. It resembles C++ language, but is easier to use.
- Java, like c++, is a language that is **multi-paradigm** and supports **object-oriented programming (oop)**, **procedural programming, and data abstraction**.
- Object-oriented programming is increasingly being used in client server technology
- It refers to a programming language model that is **organized by objects** rather than actions, data rather than logic.
- Oop **identifies objects** (sets of data) and defines their relationship to each other. These objects are then generalized into a class.
- Methods or sequences of logic are applied to classes of objects. Methods provide computational instructions and class object features provide the data that is acted upon.

- Users communicate with objects and objects with each other through specifically **defined interfaces called messages**. Java can **create applications** that are **distributed between clients and servers in a network.** Java source code (signaled by .
- Java extension) is compiled (source code transformed into object code) into byte code (signaled by .Class extension).
- A java interpreter then **executes this byte code format**.
- Java interpreters and runtime environment run on java virtual machines (jvms).
- Java is one of the most **well-suited languages** for working on the world wide web and the client server model is the primary models for working on distributed networks, of which the world wide web is just one.

- To understand how java is used in client server systems it becomes essential to understand the **major characteristics of java**.
- Syntactic of java are similarity to c and c++ languages, **java is simple, simpler, in fact, than the languages it emulates**.
- Java is also a **robust programming language, which means it creates software that will identify and correct errors and handle abnormal conditions intelligently.**
- Another major characteristic of java is that it is **object oriented programming**, which was described above.
- Oop is characterized by three properties also present in java programming: inheritance, encapsulation, and polymorphism.

- Inheritance is a major component in oop which defines a general class and then specializes these classes by adding additional details in the already written class.
- Programmers only have to write the new features since the specialized class inherits all the features of the generalized class.
- Multi threading is also an important characteristic of java that increases interactive responsiveness and real time performance.
- Threading is the way a program splits or forks itself into two or more tasks running simultaneously.
- This allows for thread based multi tasking.
- Multi threading creates the effect of multiple threads running in parallel on different machines simultaneously

- SOCKET-BASED CLIENT SERVER SYSTEMS IN JAVA
  - Java builds client server systems **with sockets**. Sockets are the **endpoints of two-way communication** between programs running in a network.
  - They are **software objects** that connect applications **to network protocols**, so they become intelligible. For example, in UNIX a program opens a socket that enables it to send and receive messages from the socket.
  - This **simplifies software development** because programmers only have to **change or specify the socket**, while the operating system is left intact. In client/server models, **the server contains sockets bound to specific port numbers.**
  - **Port numbers identify specific processes** that are to be forwarded over a network, like the internet, in the form of messages to the server.
  - The **server only has to monitor the socket and respond** when a client requests

- The server socket class in java allows for **servers to monitor requests for connections.**
- As the socket communicates over the network, java’s server socket class assigns a one port number to each client in the server and creates a socket object or instance.
- This server socket instance creates a connection and produces a thread or string of processes through which the server can communicate with the client through the socket.
- Java web servers are built according to this model.

- **Java's RMI system**
  - The other method for using java to build client server systems is RMI.
  - RMI stands for **Remote Method Invocation.**
  - By using java language and functions, programmers write object oriented programming, so **that objects that are distributed over a network** can interact with each other.
  - RMI allows **for client objects in JVMS to request services through a network from another computer (host or server**).
  - Client objects included with the request may **call** upon **methods in the remote computer** that change the results.
  - Methods are **programmed procedures attributed to a class** that are contained in each of its objects (or instances).
  - It is a characteristic of object-oriented programming

WEB SERVICES

- **Web service** is a standardized **medium to propagate communication between the client and server applications** on the WWW (world wide web). A web service is a **software module** that is designed **to perform a certain set of tasks**.
- The web service would be able to deliver functionality to the client that invoked the web service.
- A web service is a **set of open protocols and standards** that allow data to be exchanged between different applications or systems.
- Web services can be used by software programs written in a variety of programming languages and running on a variety of platforms to exchange data via computer networks such as the internet in a similar way to inter-process communication on a single computer.
- There are mainly two types of web services – SOAP (simple object access protocol and REST (representational state transfer).

COMPONENTS OF WEB SERVICE

- **SOAP (SIMPLE OBJECT ACCESS PROTOCOL)**
  - SOAP stands for “Simple Object Access Protocol.” It is a **transport-independent messaging protocol**. SOAP is built on **sending XML data in the form of SOAP messages**. A document known as an XML document is attached to each message. Only the structure of the XML document, not the content, follows a pattern. The best thing about web services and SOAP is **that everything is sent through HTTP,** the standard web protocol.
- **UDDI (UNIVERSAL DESCRIPTION, DISCOVERY, AND INTEGRATION)**
  - UDDI is a standard **for specifying, publishing and discovering a service provider’s online services**. It provides a specification that aids in the **hosting of data via web services**. UDDI provides a repository where **WSDL files can be hosted so that a client application can discover a WSDL file** to learn about the various actions that a web service offers. As a result, the client application will have full access to the UDDI, which serves as a database for all WSDL files.

- **WSDL (WEB SERVICES DESCRIPTION LANGUAGE)**
  - If a web service can't be found, it can't be used. The client invoking the web service should be aware of the location of the web service. Second, the client application must understand what the web service does in order to invoke the correct web service. The WSDL, or web services description language, is used to accomplish this. The WSDL file is another **xml-based file that explains what the web service does to the client application**. The client application will be able **to understand where the web service is located and how to use it by using the WSDL document.**

- The **web services architecture** consists of three distinct roles as given below :

1. **Provider** – the provider creates the web service and makes it available to client application who want to use it.
2. **Requestor** – a requestor is nothing but the client application that needs to contact a web service. The client application can be a .Net, java, or any other language based application which looks for some sort of functionality via a web service.
3. **Broker** – the broker is nothing but the application which provides access to the UDDI. The UDDI, as discussed in the earlier topic enables the client application to locate the web service.

MICROSERVICE

- Microservices is software architecture. From **multiple small components**, it builds a **large complex application and each small component performs a single function**. For example, it can do authentication, notification, or any payment processing.
- It is a method of **breaking large software applications into loosely coupled modules**, in which **each service runs a unique process and communicates through APIs**. It can be developed using messaging or event-driven APIs , or using non-http backed RPC mechanisms.
- The microservice defines an approach to the architecture that divides an application into a pool of loosely coupled services that implements business requirements. It is next to **service-oriented architecture (SOA)**.

- The most important feature of the microservice-based architecture is that it can perform **continuous delivery** of a **large and complex** application.
- Microservice helps in breaking the application and build a **logically independent smaller applications.**
- Microservices are **designed to cope with failure and breakdowns of large applications**. Since multiple unique services are communicating together, it may happen that a particular service fails, but the **overall larger applications remain unaffected by the failure of a single module.** For ERP, CRM, automobile, banking and other financial services microservices is the most suitable architecture.

- Microservices are all about small, self-contained services. The advantages are that these services are **highly maintainable and testable, loosely coupled with other services, independently deployable and developed by small, highly productive developer teams**. Some service types for a microservices architecture may include the following examples.
- **API gateway**
  An API (application programming interface) is the intermediate system that allows for two services to talk to each other. In the case of microservices, **an API gateway is like an entry point: it accepts requests from clients, collects the services needed to fulfill those requests from the backend and returns the correct response.**

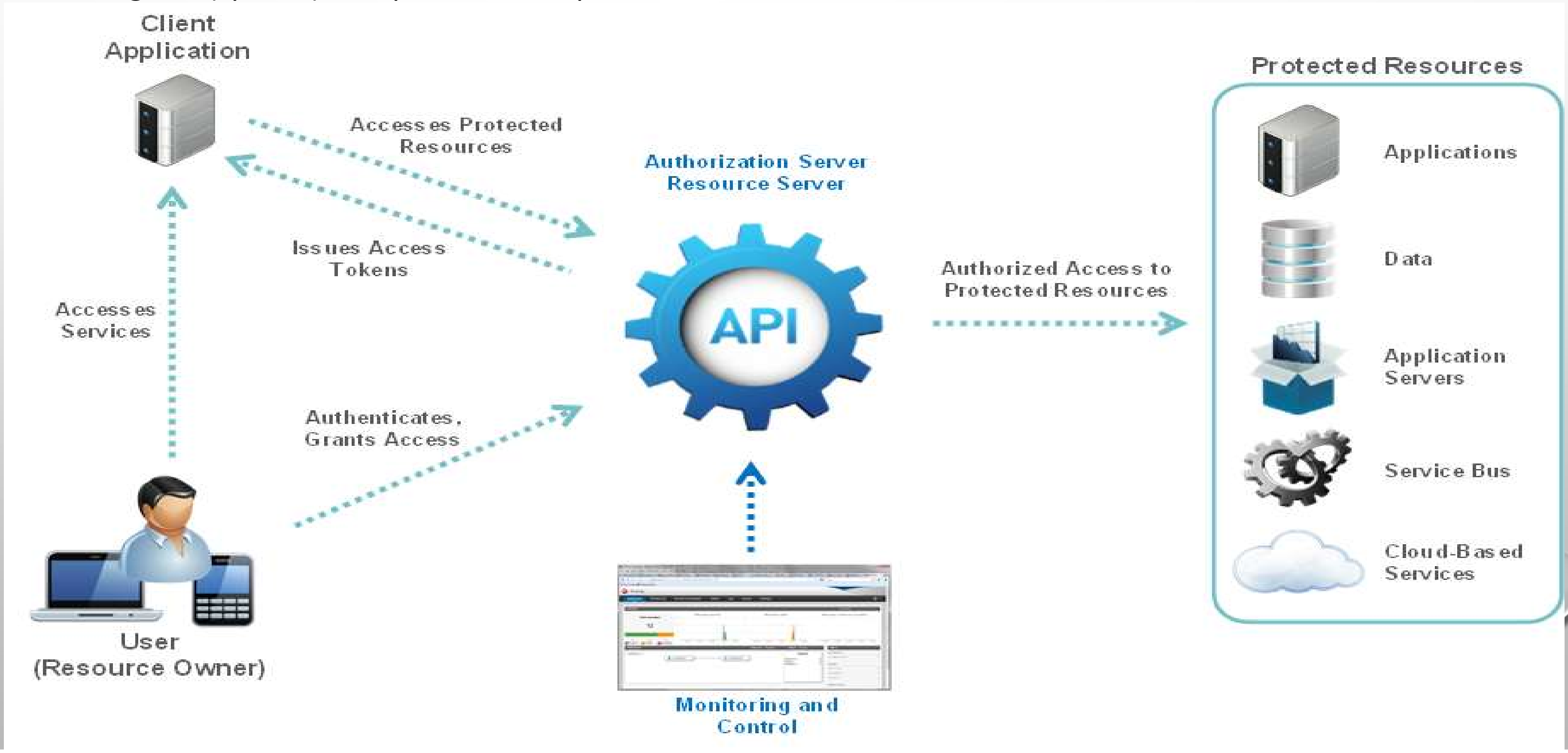
- **Benefits of a microservices architecture**
- Microservices are showing lots of success in modern cloud native businesses. Businesses benefit from using this structure for a number of reasons.
- **Flexibility:** because each service is **independent,** the **programming language can vary between all microservices** (although it's prudent to standardize as much as possible on one modern programming language).
- **Faster deployment:** not only are microservices easier to understand for most developers, but they're also **faster to deploy**. Change one thing in the code for a monolithic structure, and it affects everything across the board. Microservices are **independently deployed and don't affect other services.**

- **Scalability:** if you run everything through one application, it's hard to manage the massive scale of services as an application grows. Instead, **a microservices architecture allows a team to modify the capabilities of an individual service instead of redeploying an entire system.** This even applies to the scale of each service within an application. If a payment service, for example, is seeing more demand than other services, that microservice can be scaled as needed.
- **Isolated failures:** with a monolithic architecture, a failure in one service can compromise the entire application. **Microservices isolate each component, so if one service fails or has issues, the rest of the applications can still function**, although likely in a degraded state.
- Ultimately, microservices architecture saves teams time, offers more granular modifications to each service and scales with the need of every individual service and interface as a whole.

API GATEWAY

- An API stands for **Application Program Interface**. It is a **set of instructions, protocols, and tools for building software applications. It specifies how software components should interact.**
- The API gateway **is a server**. It is a single entry point into a system. API gateway **encapsulates the internal system architecture**. It provides an API that is tailored to each client. It also has other responsibilities such as **authentication, monitoring, load balancing, caching, request shaping and management,** and **static response handling**.
- API gateway is also responsible for **request routing, composition,** and **protocol translation**. All the requests made by the client go through the API gateway. After that, the API gateway routes requests to the appropriate microservice.

- The API gateway handles the request in one of the two ways:
    - It routed or proxied the requests to the appropriate service.
    - Fanning out (spread) a request to multiple services.



- Advantages of API gateway
- The most important advantage of API gateway is that it encapsulates the internal structure of the application.
- Rather than invoking the specific service, the client directly talks to the API gateway.
- It reduces the number of round trips between client and application.
- It simplifies the client code.
- It reduces coding efforts, makes the application more efficient, decreases errors all at the same time.
- It provides each kind of client with a specific api.
- Disadvantages
- It requires routing rules.
- There is a possibility of a single point of failure.
- Risk of complexity due to all the API rules are in one place.

AUTHENTICATION OF USERS/CLIENTS

- Traditional methods rely on **single-level authentication** with username and password to grant access to the web resources.
- Users tend to keep easy passwords or reuse the same password on multiple platforms for their convenience.
- The fact is, there is always a wrong eye on your web activities to take unfair advantage in the future.
- Due to the rising security load, **two-factor authentication** (2fa) come into the picture and introduced **token-based authentication**.
- This process reduces the reliance on password systems and added a **second layer** to security.

TOKENS/KEYS FOR AUTHENTICATION

- Token-based authentication is a **two-step authentication** strategy to enhance the security mechanism for users to access a network.
- The users once register their credentials, **receive a unique encrypted token** that is valid for a specified session time. During this session, **users can directly access the website or application** without login requirements. It **enhances the user experience by saving time and security** by adding a layer to the password system.
- A token is stateless as it does not save information about the user in the database. This system is **based on cryptography** where once **the session is complete the token gets destroyed**. So, it gets the advantage against hackers to access resources using passwords.
- The most friendly example of the token **is OTP** (one time password) which is used to verify the identity of the right user to get network entry and is valid for 30-60 seconds. During the session time, the **token gets stored in the organization's database and vanishes when the session expired**.

SERVICE MESH

- A service mesh has **proxies on both the server and client endpoints to secure any communication between both terminals**.
- A *service mesh* is a dedicated network layer that **provides secure service-to-service communication within and across infrastructure**, including on-premises and cloud environments.
- Service meshes are often used with **a microservice architectural pattern**, but can provide value in any scenario where complex networking is involved.
- A service mesh uses **lightweight containers to provide a transparent infrastructure layer over a microservice-based app**. It provides operational features to improve app **resilience, observability, and security** by allowing the app's owners to focus on the business logic.

- Using a service mesh, which is a **configurable, low-latency infrastructure layer** in software, is still a fairly **new approach**. It was born out of the need for **organizations to effectively manage the rapidly growing number of microservices** they are developing as they build applications.
- For example, an e-commerce application app that typically has microservices architecture**, with front-end and back-end components, needs services to communicate securely to support customer transactions**. Such apps can include shopping cart and shipping services.
- A **service mesh can bind all the services in a kubernetes cluster together so they can communicate with each other**. It enables secure, service-to-service communication by creating a **transport layer security** (TLS)-encrypted gateway. Service mesh features include traffic management, observability, and security.

- A typical service mesh has two main architectural components, **a data plane and a control plane**.
- Data plane
- The data plane is responsible for tasks such **as checking, routing, authentication, and authorization.** It **translates, forwards, and observes** all network **packets flowing to and from service instances**. A service mesh may include more than one data plane.
- Control plane
- The control plane in a service mesh **provides policy and configuration for all the data planes** in the service mesh. Unlike a data plane, the control plane doesn't touch any packets or requests in the system. But it can turn **all the data planes into one distributed system**.
- The control plane of a service mesh is **usually human-operated through a command-line interface** (CLI), web portal, or some other kind of user interface.

BENEFITS OF A SERVICE MESH

- A service mesh provides benefits for all organizations, ranging from security to improved application resiliency. Some of the benefits of a service mesh include;
- Service discovery
- Application health monitoring
- Load balancing
- Automatic failover
- Traffic management
- Encryption
- Observability and traceability,
- Authentication and authorization,
- Network automation

- Traditional applications work like this: A client sends HTTP requests and responses to and from a web server. That server, in turn, communicates with an application server and perhaps a database. But if an organization wants to change or upgrade any function, it must upgrade the entire app.
- With microservices, **each application function runs in its own container**. These containers need to communicate with each other across the network, which acts as an operating system of sorts. The **advantage of using microservices is that they can be rolled out or upgraded slowly or carefully without having to upgrade the whole app at once.**

- A top benefit of a service-mesh architecture for application developers is that it provides teams more flexibility in how and when they test and release new functions or services. For example, as explained below, service mesh **supports canary deployments, A/B testing, and green-blue testing**.
- **Canary deployments**
  - With a canary deployment or canary rollout, application developers can send a new version of code into production and send a proportion of users to the new version while the rest remain on the current version.
  - For example, developers could **introduce just 10 percent of a new service to start and rely on the service mesh to confirm that the service is working as intended before expanding the service further.** Through this approach, the developers can confidently introduce more of the upgrade into the service mesh over time.

- **A/B testing**
- The service mesh also lets organizations road test new ideas. For example, say a retailer wants to use a new web design for a black friday campaign. It can use the service mesh to test the new design months ahead of time to gather user feedback**. The retailer could test the design with 5 percent of customers** and, if it resonates, make it available to all customers on black friday.
- **Green-blue testing**
- Developers can also use the service mesh to conduct green-blue testing, which is an engineering process for **testing new versions of a service.** It involves running two **identical production environments and monitoring for errors or unwanted changes in user behavior** as more traffic is moved from the old version of the service to the new.

- A service-mesh architecture allows organizations to control the sharing of services among the many disparate, front-end and back-end clusters in an application, whether those clusters are located in the cloud or in on-premises environments.
- What is a sidecar proxy?
- A service mesh is, essentially, **a mesh of network proxies**. App development teams **implement** the **service mesh using sidecar proxies**, which are additional containers that proxy all connections to the containers where the services live, such as in a container orchestrator like kubernetes, also known as k8s.
- As its name suggests, a sidecar proxy in a service mesh runs alongside a service or instances, such as a kubernetes pod. Sidecar proxies enforce policies and collect telemetry in the data plane. **They can handle communications, monitoring, and security-related issues between microservices.**

Differences	API Gateway	Service Mesh
Existing Functionalities	•Useful for API requests between the client and server •Compatible with external API requests and internal calls.	•Handles internal communications •Best for improving the portability of service-to-service calls
Digital Transformation Functionality	•Longer app delivery cycles •Zero security risks	•Useful for effective microservices management •Speeds up the network's delivery cycle •Security risks due to open loopholes when the program runs alone
Operations	Reroutes API calls present outside of the given application environment	Works within the application architecture
Maturity Level	API gateway is mature technology. It has been a core developer tool for several years now	Service mesh is still a relatively new technology and requires more research and development
Security Processing	Automated security protocol	Manual security systems
Usage Complexity	Application complexity remains constant with endpoints remaining unchanged	Business scalability can be complex as each update comes with new endpoints

MESSAGE QUEUES

- Message queues provide tone benefits for modern application development. let's check out a few **advantages** you can leverage while using message queues.
- Message queues strive best at allowing application decoupling. A single application can be divided into different microservice components. Each component runs independently. **These microservices can communicate using message queues without being tightly coupled** in a monolith architecture. **A producer and consumer aren't necessarily needed to be connected simultaneously**. Each service is self-contained.
- Decoupled microservices provide application **resilience and reliability. If a receiver is not able to process a message at the time it is received, the message can be stored in a queue and processed later.**
- It creates **fault tolerance** models**. The message will be re-deliver when the destination node is back online.** This is necessary **when your application is experiencing high traffic and when down for maintenance.**

- A message queue has **the capacity to receive a large number of messages while the consumer is still processing them accordingly.** This creates **scalability advantages**. You can easily scale **to handle more messages as the system grows**. Producers are only assigned to create messages, and consumers take and deliver the messages to the intended destination. **This application handles increasing loads without requiring significant changes to the architecture.**
- Decoupling your application and **allowing the message queue to handle messages asynchronously improves the overall performance**. Consumers and producers continue with their tasks without waiting for each other**. Each component pushes messages into a queue and retrieves them when ready.** This allows the producer to free up bandwidth after delivering messages. The consumers always process new incoming messages in the background without clogging the application, and thus performance is improved.

- Message queuing provides **a more straightforward monitoring approach**. You can get transaction **insights such as sent, received, failed, rejected, and retried messages**. This provides monitoring statistics to allow you to understand performance and how data flows in your system.

CLIENT/SERVER TECHNOLOGY AND WEB APPLICATIONS

- Desktop applications run on a single computer, while web applications run on the internet.
- Since the invention of the web, developers have been trying to design web applications that **demonstrate the speed and interactivity of applications running on the client machine of a LAN**.
- Web applications are accessed by web browsers and run over a network, like the internet. They are popular because of the **power dominance of web browsers serving as thin clients.**
- Thin clients are client applications or devices that simply initiate requests and provide input.
- They do very **little of the processing**, letting the **server handle the heavy lifting** by forwarding requests and contacting different nodes and networks.
- In traditional client server computing, every application contained **a client program that provided the user interface (UI) through which users would interact** with/make requests from the applications.

- Each client program had to be installed individually on each user workstation. Web applications are different.
- Web applications **dynamically generate web documents**.
- These are HTML/XHTML (hypertext markup language/extensible hypertext markup language) documents transmitted over the internet through HTTP (hypertext transfer protocol).
- These documents or pages make up the web site.
- Using a **standard server side scripting language like javascript** allows for more interactive features in the user interface.
- Usually, each page is delivered as a static document, but the sequence in which they are presented is interactive.

- The **web browser**, acting as "**universal client**", interprets and displays the dynamic web pages.
- Web application UIs offer a variety of features. For example, **the drag and drop fea**ture allows for virtual objects to be moved from location through the mouse. This can create all sorts of actions, from copying to associating two objects to create a new action
- By using application specific technologies like java, javascript, dhtml (dynamic html), and flash, all sorts of graphic and audio interactive features may be added to a UI

1ST GENERATION WEB APPLICATIONS

- The applications that are available now are typified by the technology used/presented in the 1st generation web application
- Companies **are replacing human resources manuals and maintenance manuals** with browsers connected over intranets or the internet to a server which contains the latest information sought.
- A **primary limitation** of first generation applications is that there **is no database management system, connected to the web server** and the software does not keep the track of **who is requesting information or of the last request from that use**r.
- It is **a stateless connection**.

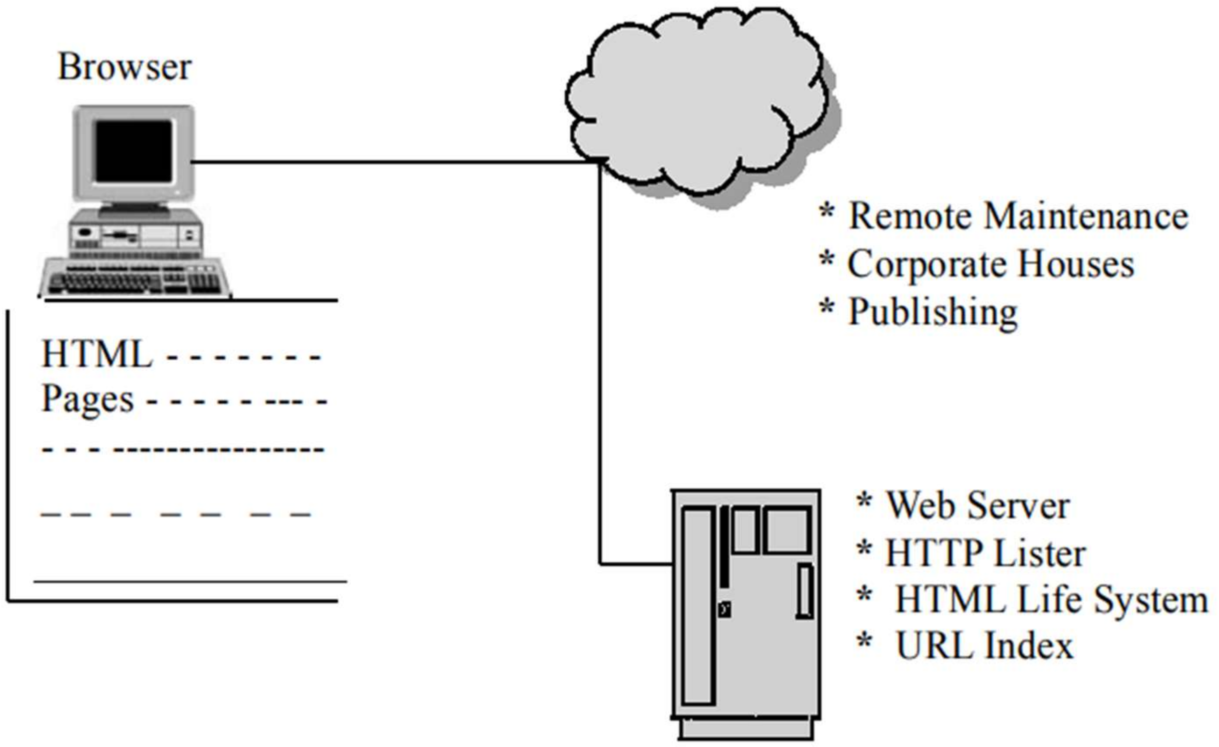


Fig. 8.3: Architecture of Ist Generation Web Application

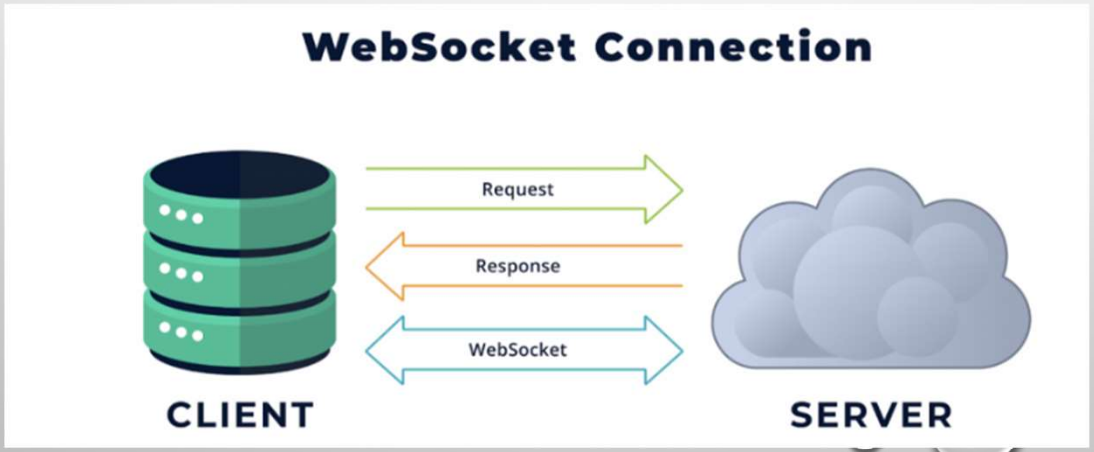
2ND GENERATION WEB APPLICATIONS

- Several things will define this newer generation that are given as below:
  - support for active clients via downloadable applets (software components),
  - live DBMS links that enable the server to know who you are from page to page, and visit to visit, and
  - True interactivity between the client and server (indicated by the two-headed arrow).
- 2nd generation web applications **have live DBMS connections on the server.**
- Today what is mostly available for such support are SQL DBMS engines from companies like SYBASE, INFORMIX, IBM and oracle.
- A **problem** with these engines is that SQL supports traditional **business data types** such as **alphanumeric, date and time**. No major SQL product supports extended and complex data types such as voice, maps or video at this time

- Sun describes its hot-java browser/java language technology as “simple, object oriented, distributed, interpreted, robust, secure, architecture-neutral, portable, high performance, multi-threaded and dynamic.”
- Html will be extended to handle embedded java, activex and other component technologies.

WEBSOCKET

- WEBSOCKET IS A COMMUNICATIONS PROTOCOL THAT ALLOWS FOR A CONSTANT FLOW OF DATA OVER A SINGLE TCP CONNECTION. IN THIS WAY, WEB APPLICATIONS MAY ENJOY REAL-TIME COMMUNICATIONS BETWEEN A CLIENT AND A SERVER.



- What websockets do is essentially extend the conversation between the client and a server. The conversation, instead of ending with that query, can continuously update in real-time and in a seamless, automatic way.
- Websockets provide a faster, continuous stream of data. HTTP is more suited for one-time data pulls.
- The websocket protocol allows for persistent communications in real-time.
- By using a bi-directional approach, the communication can stay open to constantly receive new updates.
- Developers can then build applications that make the most of real-time data. With the emergence of blockchain networks, websockets provide a key element in bringing distributed ledger data to a set of users for various functionalities.

BALANCED COMPUTING AND THE SERVER'S CHANGING ROLE

- In balanced computing, processing is shared between clients, servers, and any other devices connected via a network. This distribution is inherent in the design since applications are already spread out on different machines and connected through web services.
- Load balancing can also be achieved by building **Service-oriented Applications (Soas) where components run on different nodes in multiple locations duplicate services** on multiple nodes.
- This **duplicating of services on multiple nodes** also improves reliability since there is no single point of failure that will topple the entire application. Through balanced computing, platforms can take maximum advantage of computing and display capabilities on each platform.

- Balanced **computing not only distributes the processing load, but changes the role of the server as well.** Instead of computing so heavily, the server primarily directs traffic. Given **rich clients and decent internet connectivity**, users directly contact databases instead of requiring server intervention. Rich clients are applicants in user computers that retrieve data through the internet.
- Another method used to reduce demands on the server uses a connecting device to **redirect previously server-side processing to the client**. This depends on the client device capability and internet connectivity.

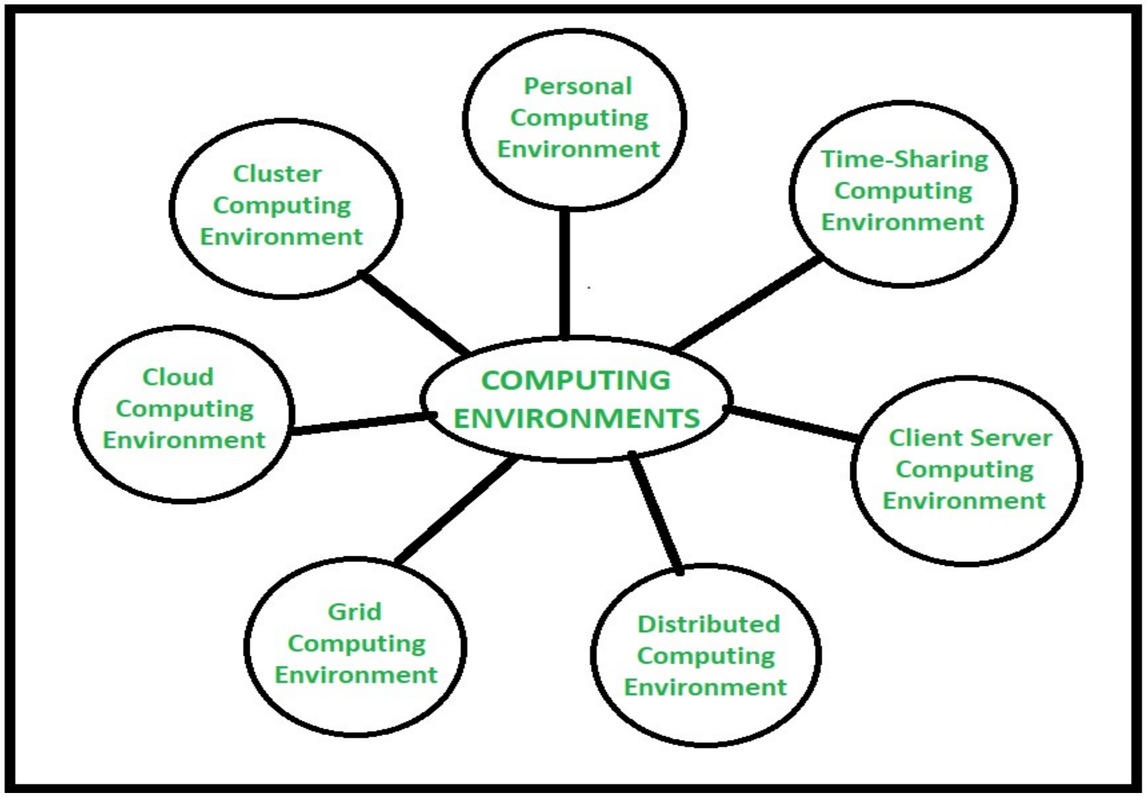
- User experience and development
  - Balanced distributed computing models improve **user experience and expand development.** Developers can focus on user experience by examining devices and customizing features.
  - For example, different **user interfaces can be customized for different departments** that require different resources to perform their function. **Different roles would have their own user interface.**
  - Well-defined user roles and profiles that are stored on user machines make more efficient use of server computation. **It allows the server to pull customized responses based on the identity/role of the user.**
  - To further reduce server demand**, clients can communicate directly with databases by requesting information for the user role profile**. This eliminates the web server as middleman for the request and computation on the output side.

- The **decentralization of distributed browser-server models** also improves security and protects privacy.
- For example, **data repositories are often located in a different location from the server**.
- This makes it **more difficult for external attackers to find**.
- It also makes it less accessible to internal attackers. Also, it is safer for user profiles to be stored on individual machines, rather than on a central database

COMPUTING ENVIRONMENT

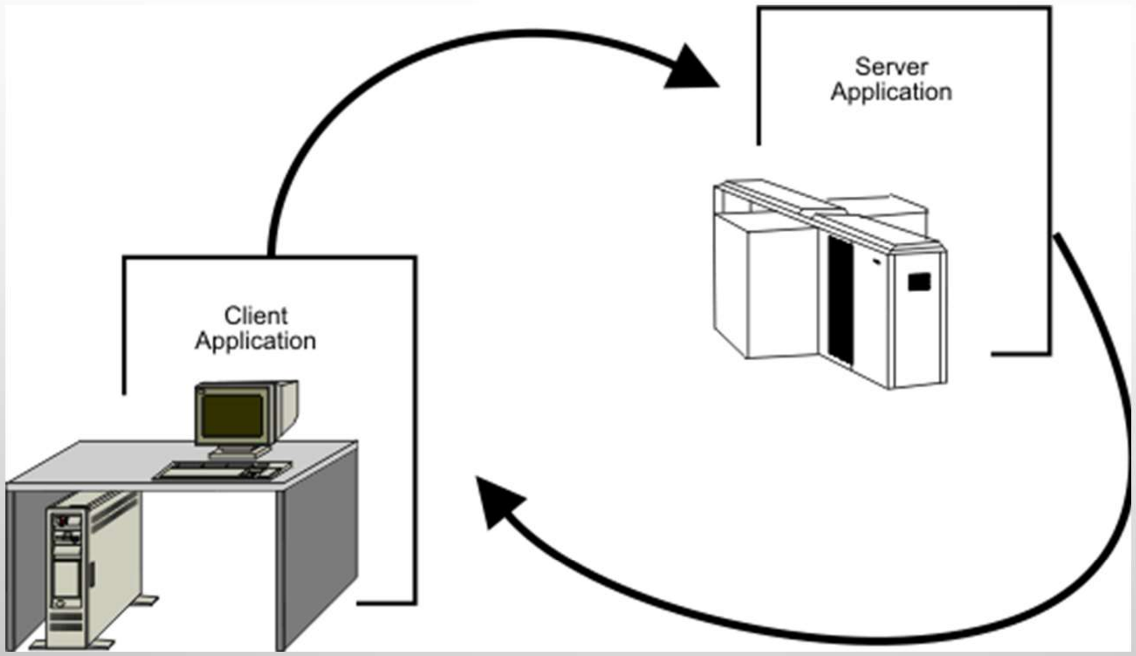
- Computing environments refer to the **technology infrastructure and software platforms** that are used to develop, test, deploy, and run software applications.
- There are several types of computing environments, including: main frame, client-server, mobile computing, grid computing, cloud computing.
- when a problem is solved by the computer, during that **computer uses many devices, arranged in different ways and which work together to solve problems**.
- This constitutes a computing environment where various number of computer devices arranged in different ways to solve different types of problems in different ways.
- In different computing environments computer devices are arranged in different ways and they exchange information in between them to process and solve problem.

- One computing environment consists of many computers other computational devices, software and networks that to support processing and sharing information and solving task.
- Based on the organization of different computer devices and communication processes there exists multiple types of **computing environments**.


COMPUTING ENVIRONMENTS
Personal Computing Environment
Time-Sharing Computing Environment
Client Server Computing Environment
Distributed Computing Environment
Grid Computing Environment
Cloud Computing Environment
Cluster Computing Environment

CLIENT SERVER COMPUTING ENVIRONMENT

- In client server computing environment **two machines** are involved i.e., **Client machine and server machine**, sometime same machine also serve as client and server.
- In this computing environment **client requests resource/service and server provides** that respective resource/service.
- A server can provide service to multiple clients at a time and here mainly communication happens through computer network.
- The client is usually the part of the application that is "seen" by the end-user.
- Therefore, the client half of a client/server application most often resides on a workstation, where the end-user can interact with the application through the workstation operating system's graphical user interface.


Server Application
Client Application

- Servers, on the other hand, are usually transparent to the end-user.
- That is, the person who sits at the workstation only perceives the client half of the application, the part that displays the information (though it was actually retrieved by a remote server).
- Because there is only one server for a given set of clients, **servers provide an ideal way of managing shared access to system resources, such as data sets.**
- For this reason, servers are likely to reside on larger machines such as z/OS mainframe computers.

THIN CLIENT COMPUTING

- A thin client is a network-dependent terminal capable of displaying remote applications that run entirely on an attached server.
- A thin-client device can be a PC, network computer (NC), or terminal.
- The key to thin-client computing is that applications run on the server—not on the client.
- An NC or PC that runs all or part of an application is not a thin client.
- A thin-client device uses one of three protocols to communicate with the server: **Independent Computing Architecture (ICA), Remote Desktop Protocol (RDP), or x**.
- These protocols transfer display information from the server to the client, and keyboard and mouse input from the client to the server.

- ICA can adapt its performance characteristics by running compressed for remote devices and uncompressed for locally attached devices.
- ICA also supports **shadowing,** which lets an **administrator take control** of a thin-client device. Shadowing is useful for end-user support and training.
- RDP will support windows-based terminals and ICA will also support windows-based terminals—and all other thin-client devices.
- RDP does not currently support shadowing.
- X is an **open-standard protocol that x terminals use**.
- X is optimized for locally attached devices and does not perform well in remote-computing situations.

- Thin clients have a number of benefits, including:
    - Reduced cost
    - Increased security
    - More efficient manageability
    - Scalability

- Thin client deployment is more **cost effective** than deploying regular pcs. Because so much is centralized at the server-side, **thin client computing can reduce IT support and licensing costs.**
- **Security** can be improved through employing thin clients because the thin client itself is restricted by the server.
- Thin clients **cannot run unauthorized software**, and **data can't be copied or saved anywhere except for the server**.
- System monitoring and management is easier based on **the centralized server location.**
- Thin clients can also be simpler to manage, since **upgrades, security policies, and more can be managed in the data center instead of on the endpoint machines**. This leads to **less downtime, increasing productivity** among IT staff as well as endpoint machine users.

- There are three ways a thin client can be used: shared services, desktop virtualization, or browser based.
- With **shared terminal services**, all users at thin client stations **share a server-based operating system and applications.** Users of a shared services thin client are **limited to simple tasks** on their machine like creating folders, as well as running it-approved applications.
- **Desktop virtualization**, or UI processing, means that each desktop lives in a virtual machine, which is partitioned off from other virtual machines in the server. The **operating system and applications are not shared resources**, but they still physically live on a remote server. These virtualized resources can be accessed from any device that is able to connect to the server.
- A **browser-based approach** to using thin clients means that an ordinary device connected to the internet carries out its application functions within a web browser instead of on a remote server. **Data processing is done on the thin client machine, but software and data are retrieved from the network**.

TRANSFORMATIONAL SYSTEM

- The working environment of many of the organizations has been greatly affected by applications of client/server technologies. Following are the examples of technologies that have changed the trade processes.

  (I) ELECTRONIC MAIL.

  (II) CLIENT/SERVER AND USER SECURITY.

  (III) EMERGENCY PUBLIC SAFETY.

  (IV) ELECTRONIC DATA INTERCHANGE

  (V) FINANCIAL ANALYSIS

ELECTRONIC MAIL

- Electronic mail is already the most heavily used network application in the corporate world.
- It is a facility that allows users at workstations and terminals to compose and exchange messages.
- Intranet mail products provide standards, simple methods for attaching documents, sound, images, and other multimedia to mail messages.
- The simplest form of electronic mail is the single system facility allowing the users of a shared computer system to exchange messages , the electronic mail facility is an application program available to any user logged onto the system.
- A user may invoke the electronic mail facility, prepare a message, and “send” it to any other user on the system.
- The act of sending simply involves putting the message in the **recipient’s mailbox.**
- Mailbox is actually **an entity maintained by the file management system** and is in nature of a file directory.

- Any incoming mail **is simply stored as a file under that user's mailbox directory**. One mailbox is associated with each user
- With a single system electronic mail facility, messages can only be exchanged among users of that particular system. For a distributed mail system, a number of mail handlers (mail servers) connect over a network facility (e.g., WAN or internet) and exchange mail.
- Intranet mail **system creates and manages an electronic mailing list** that is an alias to multiple destinations.
- **Mailing list is usually created to discuss specific topics**. Any one interested in that topic may join that list, once a client has been added to the list.
- A user can ask question or respond to some one else's question by sending a message to the list address

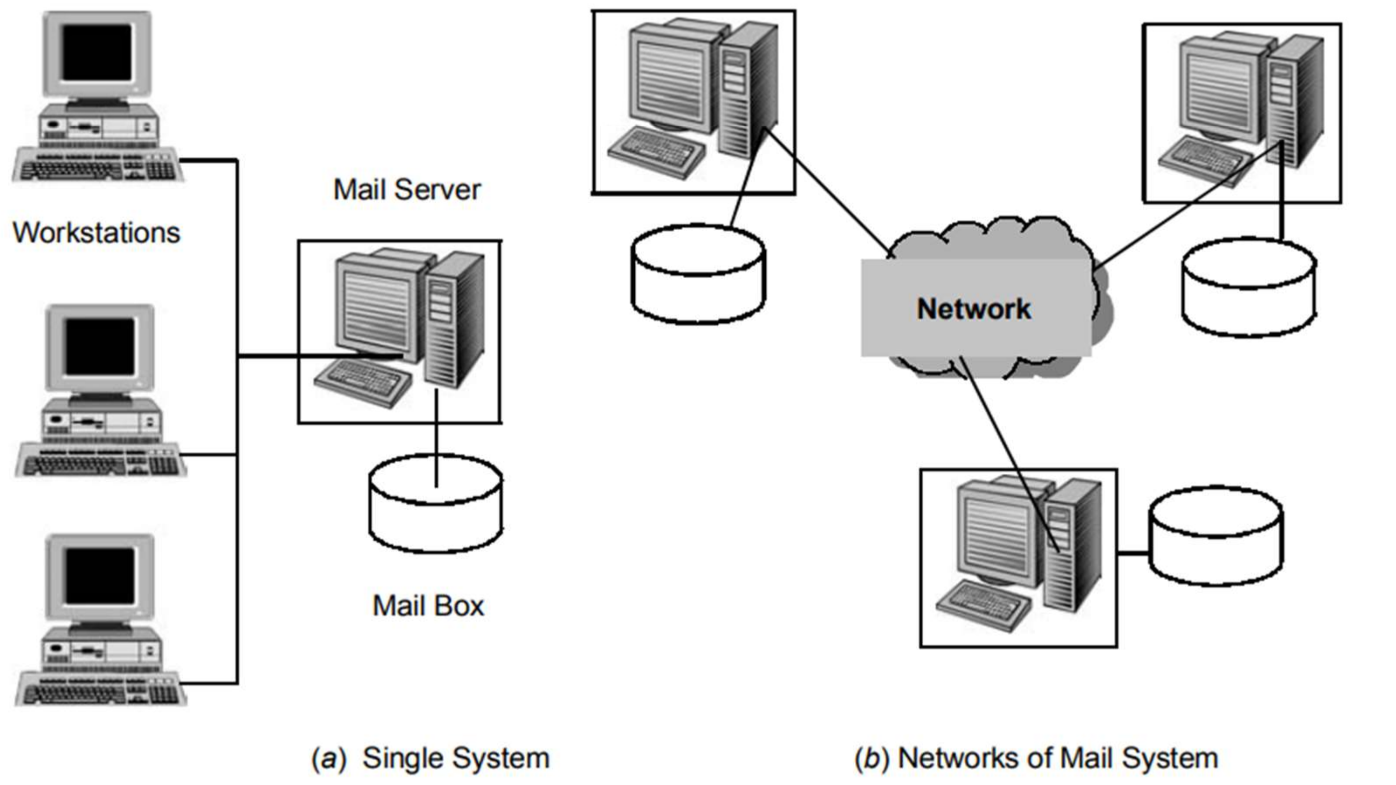


Fig. 9.4: Electronic Mail Configurations

ELECTRONIC DATA INTERCHANGE

- Electronic data interchanged **uses direct link between computers**, even computers on different sites, to transmit data to eliminate data sent in printed form.
- It is a controlled transfer of data between business and organizations via established security standards.
- One of the examples of EDI is generally thought of as replacing standardized documents such as order forms, purchase orders, delivery notes, receiving advices and invoices in a standardized electronic message format.
- EDI documents **are electronically exchanged over communication networks** which connect trading partners to one another.

- These documents **are stored in user mailboxes on the networks' EDI server** from where they can be downloaded/uploaded at the user is convenience from any one of the workstations.
- But it differs from electronic mail in that it transmits an **actual structured transaction** (with field such as the transition date, transaction amount, senders name and recipient name) as opposed to unstructured text message such as a letter.
- The main purpose of EDI is **cost reduction by eliminating paper document handling.**
- Faster electronic document transmission **further saves time and man power by avoiding the need to re-key data.** And the data arrival rate is much faster that mail.

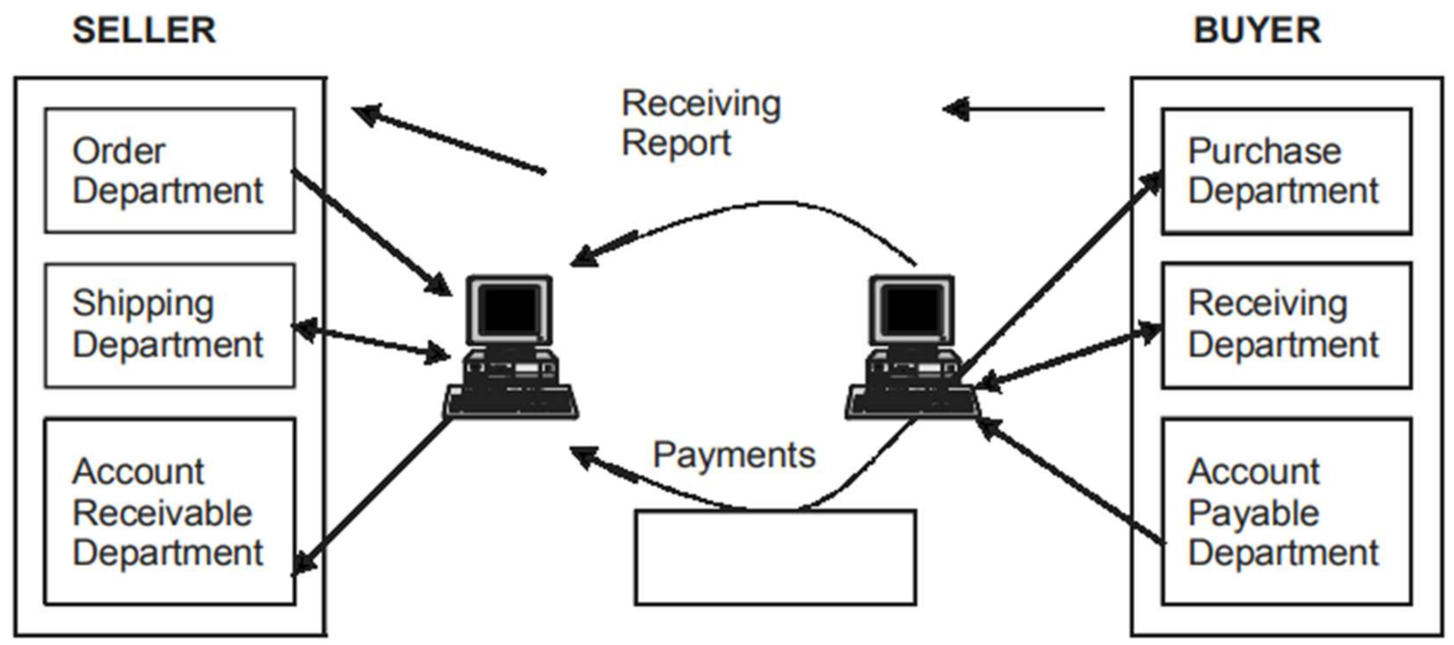


Fig. 9.9: EDI as an Example

EMERGENCY PUBLIC SAFETY

- Emergency dispatch operators are responsible for sending right emergency response vehicles to an incident as quickly as possible.
- Through the use of client/server computing it is possible to duplicate all the functionality with additional advantage of better performance, a GUI, a single point of contact, higher reliability and lower cost.
- With a client server based system , the dispatch operator is empowered to oversee how staff and equipment are allocated to each incident.

FINANCIAL ANALYSIS

- Financial analysis are overloaded with data.
- Powerful workstation technology enables these analysts to specify personal filters to be applied against the data in order to present only information of likely interest and present in the order of most likely interest.
- Improvement in technology enable the data to be scanned in real-time.
- Meaningful and useful data is available to support the analysts decision.

FUTURE OF CLIENT/ SERVER COMPUTING ENABLING TECHNOLOGIES

- (I) EXPERT SYSTEMS
- (II) FULL TEXT RETRIEVAL.
- (III) GEOGRAPHIC INFORMATION SYSTEM.
- (IV) POINT OF SERVICE TECHNOLOGY (POS).
- (V) ELECTRONIC DOCUMENT MANAGEMENT-MULTIMEDIA
- (VI) IMAGING

ENABLING TECHNOLOGIES

- CLIENT/SERVER COMPUTING DESCRIBES A MODEL FOR BUILDING APPLICATION SYSTEMS, ALONG WITH THE CORE HARDWARE AND SOFTWARE TECHNOLOGY THAT HELPS IN BUILDING THESE SYSTEMS.

EXPERT SYSTEM

- Many organizations are severly shorts of experts.
- Expert system application is well suited for client / server model.
- Expert system is a branch of artificial intelligence that makes extensive use of specialized knowledge to solve problem at the level of human expert.
- The term expert system is often applied today to any system that uses expert system technology that includes special expert system languages, programs and hardware design to aid in the development and execution of expert systems.

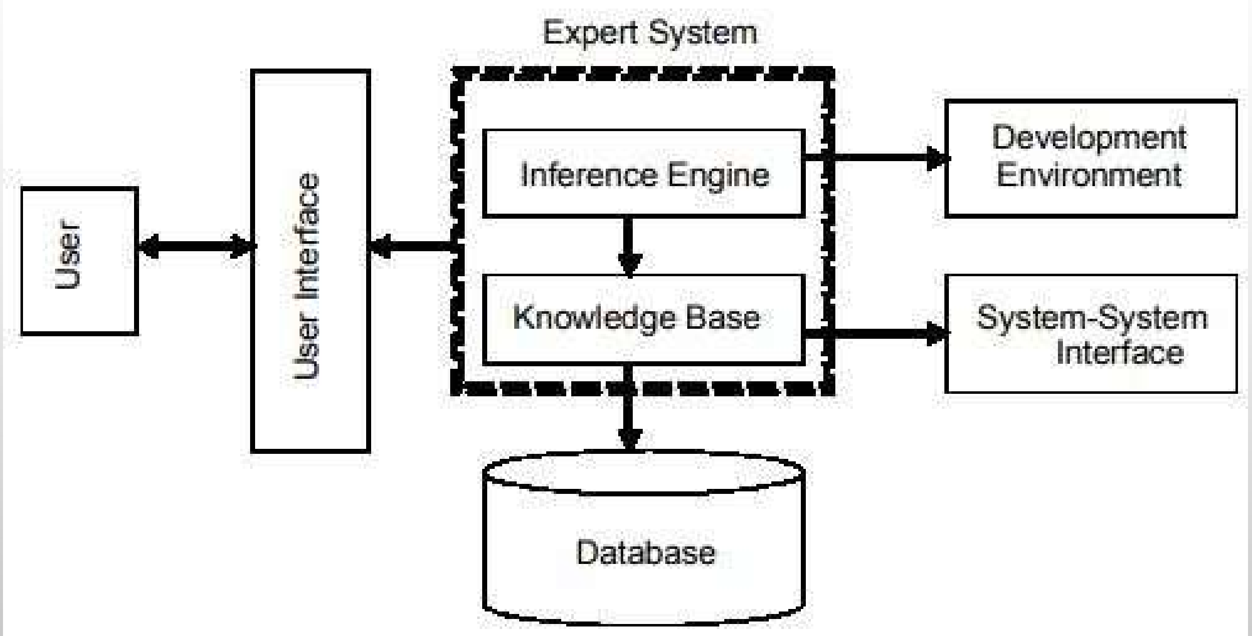


Fig.9.1: Basic Concept of Expert System

- Applications of expert system are widely accepted and well-suited for the client/server models.
- The user interface provides some rule based advantages relat ed with the processing power and some of the benefits at the workstations.
- The inference engine, a cpu-intensive process that takes advantage of low-cost processing and ram available with client/server technology, enforces the rules.
- Most applications will be implemented using existing databases on host-based

ELECTRONIC DOCUMENT MANAGEMENT.

- The meaning of a document management system (or often mistakenly called DMS system) is defined as the database-supported management of paper or electronic based documents. It also adapts classic tasks of an enterprise content management system (ECM).
- The aim is to archive records and make them available to every user who has permission within the organization. The creation, revision, control, and distribution of documents are all organized and coordinated using DMS software.
- Client-server networks, interfaces to other software, and critical business processes are commonly used to integrate these services. The key aim of document management is to improve efficiency in the work environment by reducing processing times and making necessary information faster accessible.

GEOGRAPHIC INFORMATION SYSTEMS (GIS)

- Geographic information systems (GIS) represent a combination of hardware, software and data, which allows for the capture, management, analysis, and display of geospatial (geographically referenced) information.
- In a GIS, layers of spatially explicit data are linked to tabular (attribute) data in relational databases, allowing the user to analyze and visualize patterns and trends.

- From an information management standpoint, the architecture (structure) of a GIS can be split into three main components, each contributing to its core functionality:
- A user-interface / client: allows the user to interact and use GIS tools through a graphical user interface (GUI).
- An application engine / server: the collection of tools available for the user to manipulate and analyze gis data
- A database: the data stored as files or web services and the associated database management software
- This three-tier architecture of GIS systems allows for a number of operational implementations, differentiated by the location of the main components and services (longley et al., 2010):

- Desktop: all software components are installed and run from a single machine (usually a desktop computer).
- Client-server: functionality is split between the client (computer running the user-interface), with some functionality/services and the data typically being hosted on a remote computer (server).
- Centralized desktop: a GIS desktop application with limited processing capability is installed on the client side; tools and data are hosted on a server; common with departmental implementations.
- Centralized server: enterprise GIS and web-based GIS belong to this category; simple interfaces (e.G. Web browser) allow control of the user interface (typically hosted on a remote server), while tools and databases are hosted on dedicated servers, often linked within large networks.

IMAGING

- Imaging is a conversion of documents from a physical medium to a digital form where they are manipulated by computers.
- Optical character recognition (OCR) is sometimes referred to as text recognition. An OCR program extracts and repurposes data from scanned documents, camera images and image-only pdfs. OCR systems use a combination of hardware and software to convert physical, printed documents into machine-readable text.
- **Intelligent character recognition** (ICR) is **used to extract handwritten text from image images** using icr, also referred to as intelligent OCR. It is a more sophisticated type of OCR technology that recognizes different handwriting styles and fonts to intelligently interpret data on forms and physical documents.
- Document images are stored and accessed through standard data access request.

- The movement towards standards for the creation, distribution , indexing, printing, displaying and revision of images has enabled large no of vendors to enter the market and dramatically reduce in the price of these components.
- Imaging is the method of getting digital documents from physical ones.
- An imaging system passes a paper document through a scanner that renders it digital and then stores the digital data as a bit mapped image of the document.
- The keyword for each document that helps in indexing and retrieval are entered during scanning.
- The index usually is stored in a relational database on a high-speed magnetic disk these image documents can be accessed