Software Engineering I

Dr. Elham Mahmoudzadeh

Isfahan University of Technology

mahmoudzadeh@iut.ac.ir

2021

Chapter 11
Physical Architecture Layer Design

Steps(I)

1. Preparing proposal
2. Requirements determination
   - User story
3. Abstract Business Process Modelling
4. Analysis
   - Functional Modelling
   - Structural Modelling
   - Behavioral Modelling

Steps(II)

5. Design
   - Optimization
   - Database Management
   - User Interface
   - Physical Architecture

Introduction

- An important component of the design of an information system is the design of the physical architecture layer.
- It describes the system’s hardware, software, and network environment.
- The physical architecture layer design flows primarily from the nonfunctional requirements, such as operational, performance, security, cultural, and political requirements.

Introduction

- In today's environment, most information systems are spread across multiple computers. A Web-based system, for example, runs in the browser on a desktop computer but interacts with the Web server (and possibly other computers) over the Internet. A system that operates completely inside a company's network may have a Visual Basic program installed on one computer but interact with a database server elsewhere on the network.
- Therefore, an important step of design is the creation of the physical architecture layer design, the plan for how the system will be distributed across the computers, and what hardware and software will be used for each computer.
- In many cases, systems are built to use the existing hardware and software in the organization. Therefore, the current architecture restricts the choice. Other factors such as corporate standards, existing site-licensing agreements, and product–vendor relationships also can mandate what architecture, hardware, and software the project team must use.

این طراحی مرتبطه با اون نان فانکشنال ریکوارمنت است

Introduction

- Designing the physical architecture layer can be quite difficult; therefore, many organizations hire expert consultants or assign very experienced analysts to the task.
- The nonfunctional requirements play a key role in physical architecture layer design. These requirements are reexamined and refined into more-detailed requirements that influence the system’s architecture.

Elements of the Physical Architecture Layer

- The objective of designing the physical architecture layer is to determine what parts of the application software will be assigned to what hardware.
- Although there are numerous ways the software components can be placed on the hardware components, there are three principal application architectures in use today: *server-based architectures*, *client-based architectures*, and *client–server architectures*.
- The major *architectural components* of any system are the software and the hardware.

Architectural Software Components

- The major software components of the system being developed have to be identified and then allocated to the various hardware components on which the system will operate. All software systems can be divided into four the basic building blocks.
  - *Data storage* (associated with the object persistence located on the data management layer. Most application programs require data to be stored and retrieved, whether the information is a small file such as a memo produced by a word processor or a large database that stores an organization's accounting records. These are the data documented in the structural model.
  - *Data access logic* (associated with the data access and manipulation classes located on the data management layer, the processing required to access data, which often means database queries in *SQL (structured query language).*
  - A*pplication logic*, which can be simple or complex, depending on the application. This is the logic documented in the functional and behavioral models.
  - *Presentation logic*, the presentation of information to the user, and the acceptance of the user's commands (the user interface).

---
data storage: لایه استورج دیتا است --> بخش نرم افزاریه که داده ذخیره میشه توش و بازیابی انجام میشه
2: نرم افزاریه که نقش access به اون لایه دیتا رو به عهده داره
1 , 2 نظیر میشه به لایه DM
3: نظیرمیشه به همون مدل های فانکشنال و رفتار --> نظیر میشه به لایه HCI
4: لایه یوزر اینترفیس و نمایش میشه --> ارتباط سیستم با دنیای خارج --> نظیر میشه به لایه PD

Architectural Hardware Components

- The three primary hardware components of a system are *client computers*, *servers*, and the *network* that connects them.
- Client computers are the input/output devices employed by the user and are usually desktop or laptop computers, but they can also be handheld devices, cell phones, special-purpose terminals, and so on.
- Servers are typically larger computers that are used to store software and hardware that can be accessed by anyone who has permission.
- The network that connects the computers can vary in speed from a slow cell phone, to medium-speed always-on frame relay networks, to fast always-on broadband connections such as cable modem, DSL, or T1 circuits, to high-speed always-on Ethernet, T3, or ATM circuits.

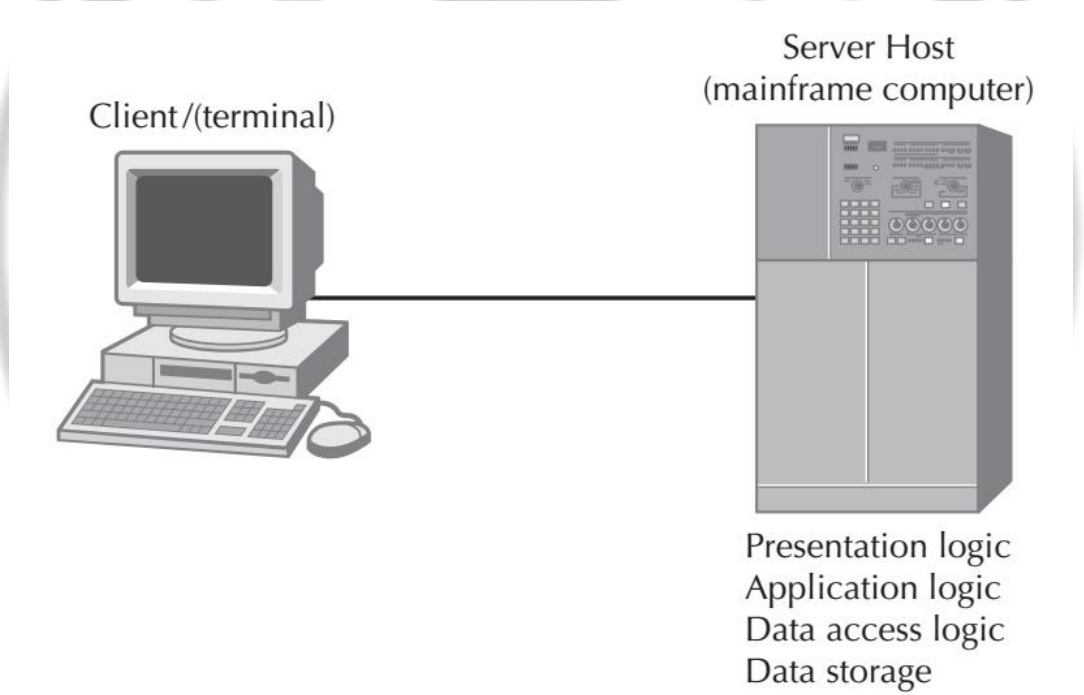
مولفه های سخت افزار که میان روی معماری میشن که کامپیوترهای ما هستن که نقش کلاینت یا سرور یا شبکه ای که اینارو بهم وصل میکنه

Server-Based Architecture

- The server performing all four functions.
- The clients enabled users to send and receive messages to and from the server. The clients merely captured keystrokes and sent them to the server for processing and accepted instructions from the server on what to display.
- Application software is developed and stored on one computer, and all data are on the same computer. There is one point of control, because all messages flow through the one central server.
- The fundamental problem with server-based networks is that the server must process all messages. As the demands for more and more applications grow, many server computers become overloaded and unable to quickly process all the users' demands. Response time becomes slower, and network managers are required to spend increasingly more money to upgrade the server computer. Unfortunately, upgrades come in large increments and are expensive; it is difficult to upgrade a little.

انواع معماری:
معماری با تمرکز سرور است
خود سرور قسمت عمده ی اون لاجیک رو به عهده می گیره و کلاینت فقط ورود و خروج است
سرور همه لاجیک هارو به عهده میگیره
کللاینت یک ترمینال است
کلاینت نقش خیلی عمده ای نداره
مشکل:
سرور باید همه کارها رو انجام بده و همه پیام ها بایید پردازش بشه و هرچی تعداد درخواست ها زیاد میشه از یه جایی به بعد اون سرور جوابگو نیست و زمان پاسخگوی سرور کم میشه و مدیریت کارها رو نمی تونه کارامد جواب بده
و یک سرور دیگه گداشتن کنارش هم سنگین در میاد

Server-Based Architecture

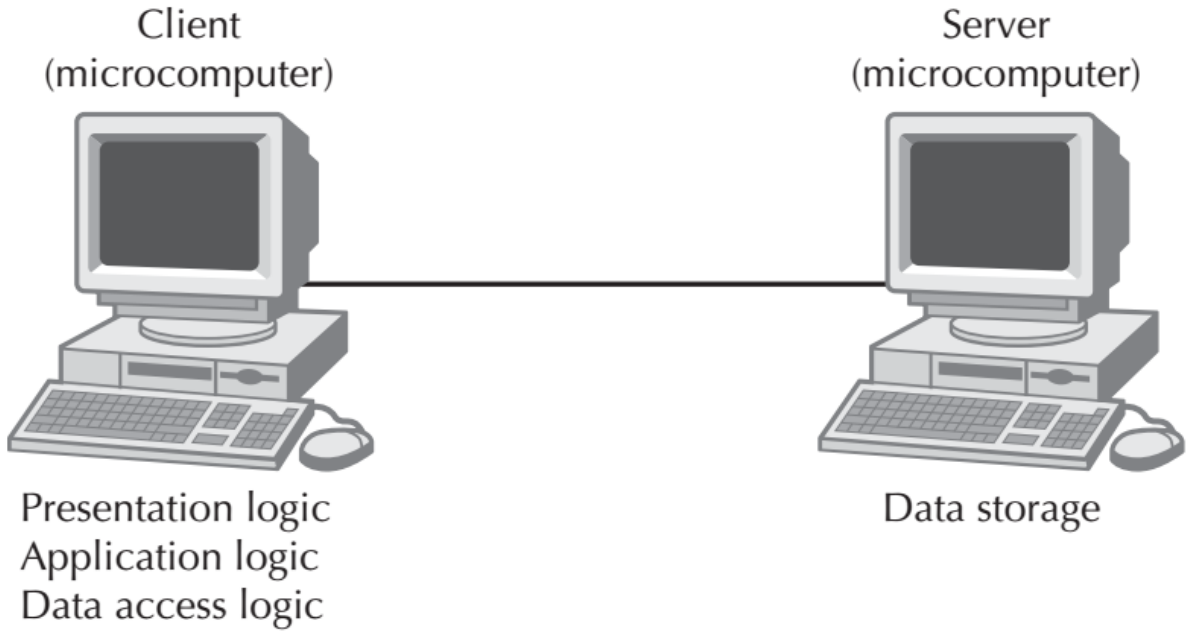


Client-Based Architecture

- Clients are personal computers on a local area network (LAN), and the server computer is a server on the same network.
- The application software on the client computers is responsible for the presentation logic, the application logic, and the data access logic; the server simply stores the data. As the demands for more and more network applications grow, the network circuits can become overloaded.
- The fundamental problem in client-based networks is that all data on the server must travel to the client for processing. This can overload both the network and the power of the client computers.

2
قسمت عمده کار به عهده کلاینت است و سرور هم فقط دیتارو ذخیره میکنه
هرچی تقاضا بیشتر میشه همه دیتا باید بیاد سمت کلاینت و بعد کلاینت برمبنای اون اپلیکیشن تصمیم بگیره چه کاری باید روی اون دیتا انجام بده پس حجم ترافیک شبکه بیش از حد اشغال میشه
مشکل:
داده باید بیاد سمت کلاینت و overload میشه شبکه و قدرت اون کلاینته
اصل خود دیتا روی سرور است ولی access به دیتا رو کلاینت انجام میده

Client-Based Architectures

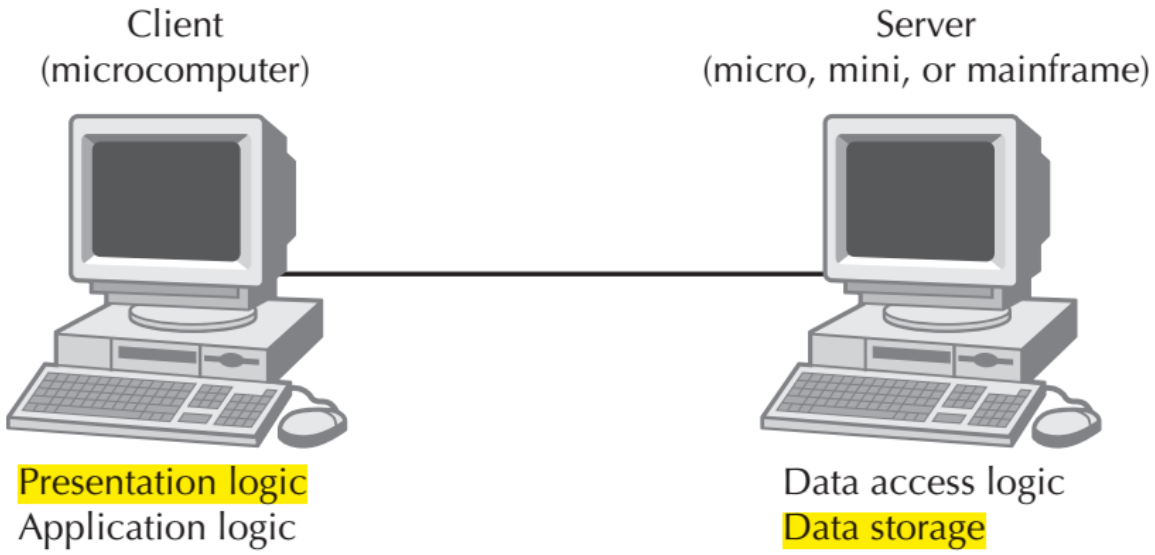


Client–Server Architecture

- Balance the processing between the client and the server by having both do some of the application functions. In these architectures, the client is responsible for the presentation logic, whereas the server is responsible for the data access logic and data storage. The application logic may reside on either the client or the server or be split between both. The client can be referred to as a *thick*, or *fat*, *client* if it contains the bulk of application logic.
- For example, many Web-based systems are designed with the Web browser performing presentation, with only minimal application logic using programming languages like Java and the Web server having the application logic, data access logic, and data storage.

بین دوتا توزیع بشه
اپلیکیشن لاجیک می تونه توزیع بشه روی هرکدوم از اونها و برمبنای این که چقدر از این اپلیکیشن لاجیک روی سرور باشه یا کلاینت چاق بودن و لاغر بودن اونارو نشون میده
مثلا برای سیستم های وب بیس که طراحی میکنیم مرورگر نقش کلاینت رو داره و یک مینیممی از کارهای اپلیکیشن رو هم انجام میده
وب سرور هم میشه سرور و قسمت های دیگه از اپلیکیشن لاجیک هم داره

Client–Server Architecture



Client–Server Architecture benefits

- They are *scalable.* That means it is easy to increase or decrease the storage and processing capabilities of the servers.
- If one server becomes overloaded, you simply add another server so that many servers are used to perform the application logic, data access logic, or data storage.
- The cost to upgrade is much more gradual, and you can upgrade in smaller steps rather than spending hundreds of thousands to upgrade a mainframe server.
- Client–server architectures can support many different types of clients and servers. It is possible to connect computers that use different operating systems so that users can choose which type of computer they prefer.
- Finally, because no single server computer supports all the applications, the network is generally more reliable. There is no central point of failure that will halt the entire network if it fails, as there is in server-based computing. If any one server fails in a client–server environment, the network can continue to function using all the other servers.

مزیت معماری کلاینت و سرور:
scalable است ینی تعداد تجهیزات یا .. رو کم یا زیاد بکنیم
هزینه ارتقا سیستم اینجا خیلی گرون نیست و اینجا می تونیم کم کم ارتقا رو انجام بدیم مثل قبلی ها نیست پس گام سنگینی برای ارتقا نداریم
این معماری انواع مختلفی از کلاینت ها و سرور هارو می تونه ساپورت بکنه
نت ورک اینجا قابل قبول تر است
اینجا مثل سرور بیس نیست که اگه سرور شکست خورد همش از بین بره اینجا این اتفاق نمی افته می تونیم بک اپ یا سرور های دیگه اون کار انجام بشه

Client–Server Architecture limitation

- The most important of which is its complexity. All applications in client–server computing have two parts, the software on the client and the software on the server. Writing this software is more complicated than writing the traditional all-in-one software used in server-based architectures.
- Updating the network with a new version of the software is more complicated, too. In server-based architectures, there is one place where application software is stored; to update the software, we simply replace it there. With client–server architectures, we must update all clients and all servers.

محدودیت هایی:
پیچیدگی معماری کلاینت و سرور است ینی نرم افزار روی کلاینت و روی سرور از ماژول های مختلفی هستن
اپدیت کردن اون شبکه با ورژن های جدیدی از اون نرم افزار چون همه کلاینت ها و سرور ها باید اپدیت بشن

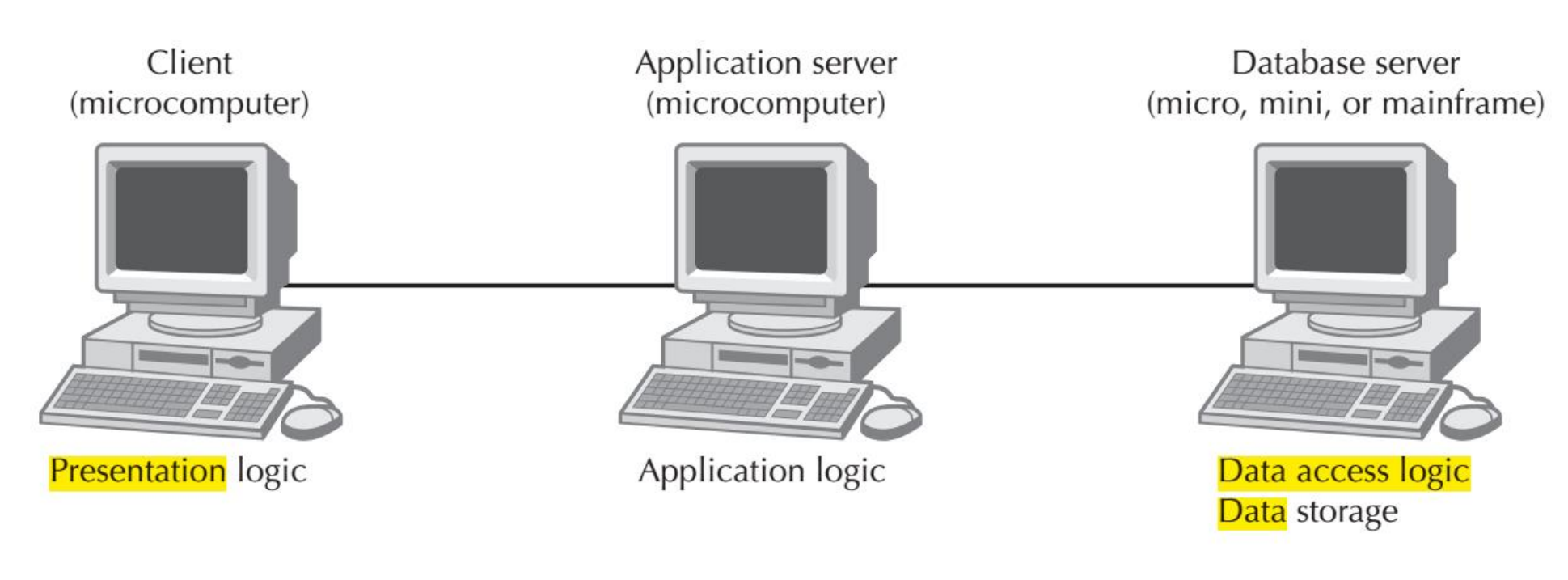
که مهمترین آن پیچیدگی آن است. تمام برنامه های کاربردی در محاسبات کلاینت-سرور دارای دو بخش نرم افزار روی کلاینت و نرم افزار روی سرور هستند. نوشتن این نرمافزار پیچیدهتر از نوشتن نرمافزار همهکاره سنتی مورد استفاده در معماریهای مبتنی بر سرور است.
به روز رسانی شبکه با نسخه جدید نرم افزار نیز پیچیده تر است. در معماری های مبتنی بر سرور، یک مکان وجود دارد که نرم افزار کاربردی در آن ذخیره می شود. برای به روز رسانی نرم افزار، به سادگی آن را در آنجا جایگزین می کنیم. با معماری های کلاینت-سرور، ما باید همه کلاینت ها و همه سرورها را به روز کنیم.

Client–Server Tiers

- There are many ways the application logic can be partitioned between the client and the server.
- Client-Server architecture is called a *two-tiered architecture* because it uses only two sets of computers, clients, and servers.
- A *three-tiered architecture* uses three sets of computers. In this case, the software on the client computer is responsible for presentation logic, an application server (or servers) is responsible for the application logic, and a separate database server (or servers) is responsible for the data access logic and data storage.
- An *n-tiered architecture* uses more than three sets of computers. In this case, the client is responsible for presentation, database servers are responsible for the data access logic and data storage, and the application logic is spread across two or more different sets of servers.

قبلی به دو دسته تقسیم میشد ینی به دو دسته کلاینت و سرور تقسیم می شن ینی دو مجموعه از کامپیوترها ولی خب می تونیم بیشتر هم داشته باشیم مثلا معماری 3 یا 4 یا n
اگر 3 باشه ینی سه تا کامپیوتر داریم

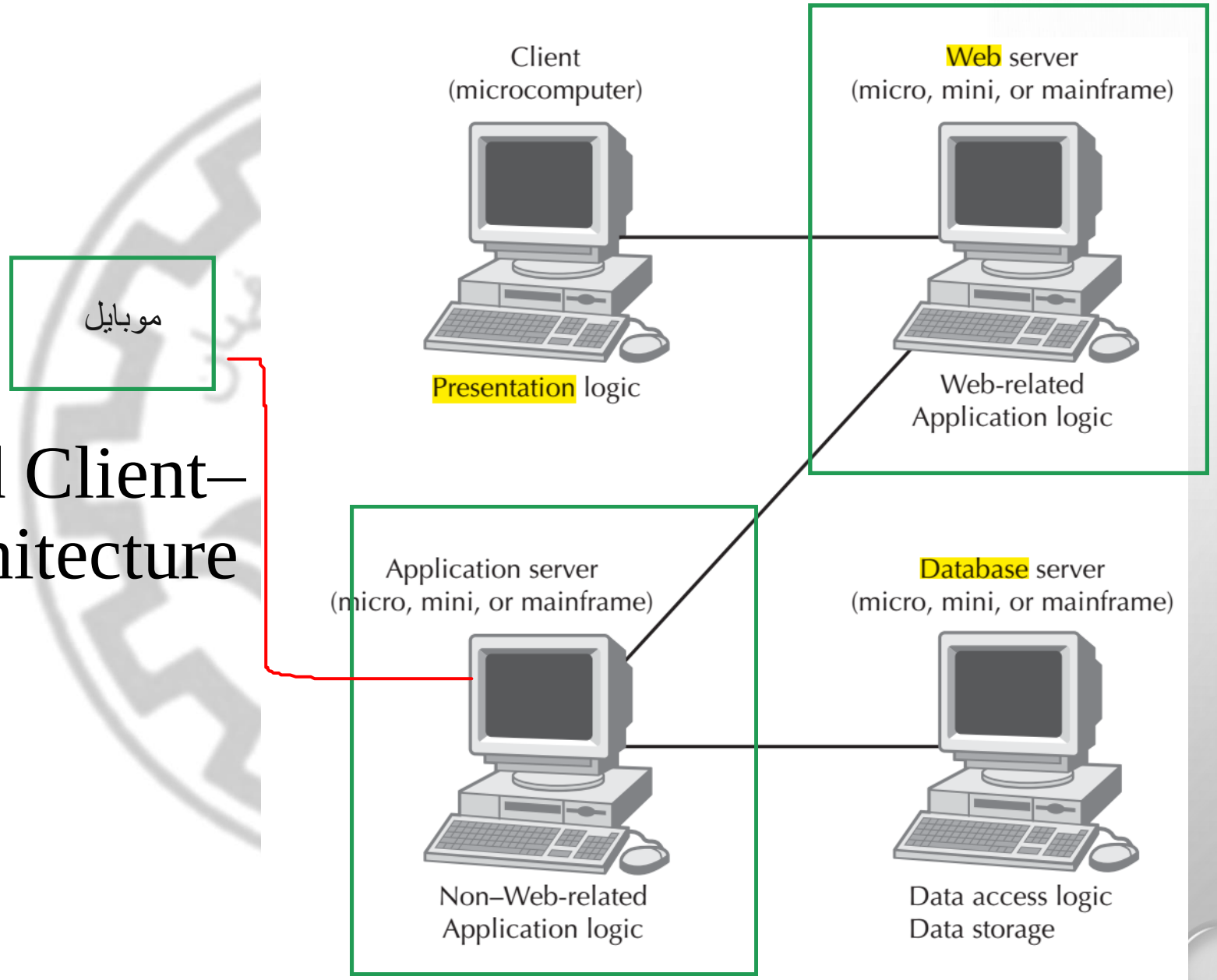
Three-Tiered Client–Server Architecture



Four-Tiered Client–Server Architecture

- This type of architecture is common in today's e-commerce systems. The first component is the Web browser on the client computer employed by a user to access the system and enter commands (presentation logic). The second is a Web server that responds to the user's requests, either by providing (HTML) pages and graphics (application logic) or by sending the request to the third component on another application server that performs various functions (application logic). The fourth component is a database server that stores all the data (data access logic and data storage). Each of these four components is separate, making it easy to spread the different components on different servers and to partition the application logic on two different servers.

Four-Tiered Client–Server Architecture



N-Tiered Client–Server Architecture

- The primary advantage of an *n*-tiered client–server architecture compared with a two tiered architecture (or a three-tiered architecture with a two-tiered architecture) is that it separates the processing that occurs to better balance the load on the different servers; it is more scalable. If we discover that the application server is too heavily loaded, we can simply replace it with a more powerful server or just put in several more application servers. Conversely, if we discover the database server is underused, we could store data from another application on it.
- There are two primary disadvantages to an *n*-tiered architecture compared with a two-tiered architecture (or a three-tiered architecture with a two-tiered architecture). First, the configuration puts a greater load on the network. N-tiered model requires more communication among the servers; it generates more network traffic, so you need a higher-capacity network. It is also much more difficult to program and test software in *n*-tiered architectures than in two-tiered architectures because more devices have to communicate to complete a user's transaction.

مهمترین مزیتش اینه که توزیع میکنیم پردازش هارو بین تایرها و یک بالانسی بین لود کلاینت ها و لود سرور ها داره برقرار میشه
توزیع حجم پردازش و لود بین کامپیوترها انجام میشه و این باعث میشه حجم پردازش ها یک جا نباشه

عیب:
پیچیدگی است که زیاد است
چون داره توزیع می کنیم پیکربندی کارها هزینه بیشتری داره
ارتباط بین سرور ها و لودی که روی شبکه میاد زیاد میشه
تست و دولوپ کردنش نسبت به قبلی ها بیشتر است
برای اینکه یک فرایند انجام بشه و به نتیجه برسه دیوایس های زیادی باید باهم در ارتباط باشن تا به اون نتیجه از اون فرایند برسیم

Selecting a Physical Architecture

- Most systems are built to use the existing infrastructure in the organization, so often the current infrastructure restricts the choice of architecture. For example, if the new system will be built for a mainframe-centric organization, a server-based architecture may be the best option. Other factors such as corporate standards, existing licensing agreements, and product/vendor relationships can also mandate what architecture the project team needs to design.
- However, many organizations now have a variety of infrastructures available or are openly looking for pilot projects to test new architectures and infrastructures, enabling a project team to select an architecture based on other important factors.

کدومشون رو انتخاب بکنیم؟
زیرساخت هایی که داریم توی سازمان ها ممکنه انتخاب مارو محدود بکنه
پارامتر دیگه: توافق یا استاندارد ها ...

همه اینا میرسه به معماری که میشه داشته باشیم

Characteristics of Computing Architectures

Characteristic	Server-based	Client-based	Client–Server
Cost of infrastructure	Very high	Medium	Low
Cost of development	Medium	Low	High
Ease of development	Low	High	Low to medium
Interface capabilities	Low	High	High
Control and security	High	Low	Medium
Scalability	Low	Medium	High

1: هزینه زیر ساخت--> زیاد توی سرور بیس چون کامپیوترهای بزرگی داریم با حجم ذخیره بزرگ داده - چون کلاینت هم باید اینجا تصمیم بگیره پس میانه میشه - توزیع میشه کار پس حجم بار کم میشه
2: هزینه دولوپ کردن سیستم --> کلاینت: فقط میخوایم ریکوست بزنیم به دیتابیس - کلاینت وسرور: خیلی بالاست بخاطر پیچیدکی هایی که وجود داره و تست و دیبا کردن کل سیستم چون داره توزیع میشه روی دیوایس ها اینها هزینه های زیادیه - سرور: بخاطر ایکه یک ماژول نرم افزاری و فقط می خوایم اون دیتا رواز کلاینت بگیریم و بعد تحویل بدیم خیلی کار خاصی نیست ولی بخاطر اینکه مین فریم است و زبان ها متنی است و گرافیکالی نیستن هزینه یکم بیشتری داریم
3: راحت دولوپ کردن کار --> سرور: چون زبان ها متنی هستن و گرفیکالی نیستن - کلاینت چون گرافیکالی هستن و ماژول هاشون هم توزیع نمیشن - راحتی دولوپ نسبتا کمه بخاطر پیچیدگی توزیع ماژول ها بین تایرها
5 : سرور: کاملا تحت کنترل باشن چون داده رو نمیاریم روی کامپیوترهای مختلف بلکه یک کامپیوتر و می تونیم ورودی و خروجیشو رو کنترل بکنیم و سازمان های نظامی از این معماری می رن بخاطر کنترل کردن ورودی و خروجی - کلاینت کم داره چون کاملا دیتا رو میاریم روی شبکه و بعد کلاینت تصمیم میگیره که چی کار بکنه - کلاینت و سرور : سرور می تونه کنترل بشه ولی چون داره توزیع میشه هم عملکرد و هم دیتا نمیشه هم گفت خیلی تحت کنترل است

Cost of Infrastructure

- One of the strongest driving forces to client–server architectures is cost of infrastructure (the hardware, software, and networks that will support the application system).
- Simply put, personal computers are more than 1,000 times cheaper than mainframes for the same amount of computing power. The personal computers on our desks today have more processing power, memory, and hard disk space than the typical mainframe of the past, and the cost of the personal computers is a fraction of the cost of the mainframe. Therefore, the cost of client–server architectures is low compared to server-based architectures that rely on mainframes.
- Client–server architectures also tend to be cheaper than client-based architectures because they place less of a load on networks and thus require less network capacity.

Cost of Development

- The cost of developing systems is an important factor when considering the financial benefits of client–server architectures.
- Developing application software for client–server computing is extremely complex, and most experts believe that it costs four to five times more to develop and maintain application software for client–server computing than it does for server-based computing.
- Developing application software for client-based architectures is usually cheaper still, because there are many GUI development tools for simple stand-alone computers that communicate with database servers.
- The cost differential might change as more companies gain experience with client–server applications, new client–server products are developed and refined, and client–server standards mature. However, given the inherent complexity of client–server software and the need to coordinate the interactions of software on different computers, there is likely to remain a cost difference.

Ease of Development

- In most organizations today, there is a huge backlog of mainframe applications, systems that have been approved but that lack the staff to implement them. This backlog signals the difficulty in developing server-based systems. The tools for mainframe-based systems often are not user friendly and require highly specialized skills—skills that new graduates often don't have and aren't interested in acquiring.
- Contrast, client-based and client–server architectures can rely on *graphical user interface* (*GUI*) development tools that can be intuitive and easy to use. The development of applications for these architectures can be fast and painless. Unfortunately, the applications for client–server systems can be very complex because they must be built for several layers of hardware (e.g., database servers, Web servers, client workstations) that need to communicate effectively with one another. Project teams often underestimate the effort involved in creating secure, efficient client–server applications.

Interface Capabilities

- Typically, server-based applications contain plain, character-based interfaces.
- For example, think about airline reservation systems, which can be quite difficult to use unless the operator is well trained on the commands and hundreds of codes that are used to navigate through the system.
- Today, most users of systems expect a GUI or a Web-based interface that they can operate using a mouse and graphical objects. GUI and Web development tools typically are created to support client-based or client–server applications; rarely can server-based environments support these types of applications.

Control and Security

- The server-based architecture was originally developed to control and secure data, and it is much easier to administer because all the data are stored in a single location.
- Client–server computing requires a high degree of coordination among many components, and the chance for security holes or control problems is much more likely. Also, the hardware and software used in client–server architecture are still maturing in terms of security.

Scalability

- Refers to the ability to increase or decrease the capacity of the computing infrastructure in response to changing capacity needs.
- The most scalable architecture is client–server computing because servers can be added to (or removed from) the architecture when processing needs change. Also, the types of hardware that are used in client-server situations typically can be upgraded at a pace that most closely matches the growth of the application.
- In contrast, server-based architectures rely primarily on mainframe hardware that needs to be scaled up in large, expensive increments, and client-based architectures have ceilings above which the application cannot grow because increases in use and data can result in increased network traffic to the extent that performance is unacceptable.

Nonfunctional Requirements and Physical Architecture Layer Design(I)

- The design of the physical architecture layer specifies the overall architecture and the placement of software and hardware that will be used.
- Most organizations use client–server architectures for cost reasons, so in the event that there is no compelling reason to choose one architecture over another, cost usually suggests client–server.
- Creating a physical architecture layer design begins with the nonfunctional requirements. The first step is to refine the nonfunctional requirements into more-detailed requirements that are then used to help select the architecture to be used (server-based, client-based, or client–server) and what software components will be placed on each device.

طراحی لایه فیزیکی کاملا مرتبط با نان فانکشنال
مهمممم است
گام اول این نان فانکشنال رو بیایم جزئی تر بکنیم و دقیق تر بشه و بعد برمبنای اون اطلاعات رو
که داریم بیایم معماریشو مشخص بکنیم

Nonfunctional Requirements and Physical Architecture Layer Design(II)

- In a client–server architecture, one also has to decide whether to use a two-tier, three-tier, or *n*-tier architecture.
- Then the nonfunctional requirements and the architecture design are used to develop the hardware and software specification.
- Four primary types of nonfunctional requirements can be important in designing the architecture: operational requirements, performance requirements, security requirements, and cultural/political requirements. Furthermore, each of these requirements must be fully verified and validated.

پس اگر کلاینت وسرور n اش باید معلوم باشه که چنده
به 4 دسته تقسیم میشه

Operational Requirements

- Specify the operating environment(s) in which the system must perform and how those might change over time.
- Refers to operating systems, system software, and information systems with which the system must interact, but on occasion it also includes the physical environment if the environment is important to the application.

مشخص میکنه که محیط عملیاتی که سیستم می خواد کار بکنه چی باشه و چگونه در طول زمان تغییر بکنه این شامل سیستم عامل میشه ، سیستم نرم افزاری میشه و..

Operation Requirements

Type of Requirement	Definition	Examples
Technical Environment Requirements	Special hardware, software, and network requirements imposed by business requirements	• The system will work over the Web environment with Internet Explorer. • All office locations will have an always-on network connection to enable real-time database updates. • A version of the system will be provided for customers connecting over the Internet via a tablet or smartphone.
System Integration Requirements	The extent to which the system will operate with other systems	• The system must be able to import and exportExcel spreadsheets. • The system will read and write to the main inventory database in the inventory system.
Portability Requirements	The extent to which the system will need to operate in other environments	• The system must be able to work with differentoperating systems (e.g., Linux, Mac OS, and Windows). • The system might need to operate with handheld devices, such as Android and Apple iOS devices.
Maintainability Requirements	Expected business changes to which the system should be able to adapt	• The system will be able to support more than one manufacturing plant with six months' advance notice. • New versions of the system will be released every six months.

1: ریکوارمنت هایی که برمیگرده به محیط فنی
2: ینی اگر واقعا نیازه که این سیستم با سیستم های دیگه در ارتباط باشه مشخص میشه --> نحوه ارتباط برقرار کردن سیستم ها با هم دیگه و ارتباطه در حد ضرورت باشه
3: سیستم بتونه توی دیوایس های مختلف یا روی محیط های مختلف عملیات رو به نحو درستی انجام بده --> هرچی این بیشتر باشه هزینه های دیگه می تونه بیشتر باشه پس این هم باید در حد نیاز تعریف بشه
4: قدرت یا قابلیت نگهداری سیستم و قابلیت رشد سیستم ینی چقدر سیستم رو ساپورتش بکنیم و با چه مدتی و با چه پلنی

Technical Environment Requirements

- Specify the type of hardware and software system on which the system will work. These requirements usually focus on the operating system software, database system software, and other system software.
- It also includes all of the different types of hardware from mainframe computers to smartphones.
- Depending on the applications being deployed over the physical architecture, specialized hardware could be required.

--

System Integration Requirements and Portability Requirements

- *System integration requirements* are those that require the system to operate with other information systems, either inside or outside the company. These typically specify interfaces through which data will be exchanged with other systems.
- *Portability Requirements* Information systems never remain constant. Business needs change and operating technologies change, so the information systems that support them and run on them must change, too. *Portability requirements* define how the technical operating environments might change over time and how the system must respond. *Portability requirements* also refer to potential changes in business requirements that drive technical environment changes.

Maintainability Requirements

- *Maintainability requirements* specify the business requirement changes that can be anticipated. Not all changes are predictable, but some are.
- All information systems must be written to make it easy to track each plant separately, whether for personnel, budgeting, or inventory systems. The maintainability requirements attempt to anticipate future requirements so that the systems designed today will be easy to maintain if and when those future requirements appear.
- Maintainability requirements can also define the update cycle for the system, such as the frequency with which new versions will be released.

Performance Requirements

- Focus on performance issues, such as response time, capacity, and reliability.

Type of Requirement	Definition	Examples
Speed Requirements	The time within which the system must perform its functions	• Response time must be less than 7 seconds for any transaction over the network. • The inventory database must be updated in real time. • Orders will be transmitted to the factory floor every 30 minutes.
Capacity Requirements	The total and peak number of users and the volume of data expected	• There will be a maximum of 100–200 simultaneous users at peak use times. • A typical transaction will require the transmission of 10K of data.
Availability and Reliability Requirements	The extent to which the system will be available to the users and the permissible failure rate due to errors	• The system will store data on approximately 5,000 customers for a total of about 2 MB of data. • Scheduled maintenance shall not exceed one 6-hour period each month. • The system shall have 99% uptime performance.

1: سرعت پاسخگویی سیستم / سرعت عملکرد سیستم --> اگر محدودیت خاصی داریم روش باید اینجا معلوم بشه --> بخش عمده است توی پرفرمنس
2: پیک یوزر ها و دیتایی که برای ذخیره سازی داده ها نیازه --> ینی پیک یوزرها که دارن همزمان وصل میشن حداکثر چقدر است
3: چقدر سیستم در دسترس است و پاسخگو است --> اگر نیازه سیستم توی زمان های خاصی پاسخگو باشه باید مشخص بشه
پس چقدر پاسخگو است و چقدر به اون پاسخ های سیستم می تونیم اطمینان بکنیم و چقدر قابل قبوله که سیستم خطا بده

روی مسائل مربوط به عملکرد، مانند زمان پاسخگویی، ظرفیت و قابلیت اطمینان تمرکز کنید.

Speed Requirements

- Are exactly what they say: How fast should the system operate?
- First is the *response time* of the system: How long it takes the system to respond to a user request. Although everyone would prefer low response times, with the system responding immediately to each user request, this is not practical. We could design such a system, but it would be expensive. Most users understand that certain parts of a system will respond quickly, whereas others are slower. Actions that are performed locally on the user's computer must be almost immediate (e.g., typing, dragging, and dropping), whereas others that require communicating across a network can have longer response times (e.g., a Web request).
- The second aspect of speed requirements is how long it takes transactions in one part of the system to be reflected in other parts. For example, how soon after an order is placed will the items it contained be shown as no longer available for sale to someone else? If the inventory is not updated immediately, then someone else could place an order for the same item, only to find out later it is out of stock. This is especially true when one considers NoSQL database that does not update all copies of the data immediately.

Capacity Requirements

- Attempt to predict how many users the system will have to support, both in total and simultaneously. Capacity requirements are important in understanding the size of the databases, the processing power needed, and so on.
- The most important requirement is usually the peak number of simultaneous users because this has a direct impact on the processing power of the computer(s) needed to support the system. It is often easier to predict the number of users for internal systems designed to support an organization’s own employees than it is to predict the number of users for customer-facing systems, especially those on the Web.

Availability and Reliability Requirements

- Availability focus on the extent to which users can assume that the system will be available for them to use. Although some systems are intended to be used only during the forty-hour workweek, some systems are designed to be used by people around the world. For such systems, project team members need to consider how the application can be operated, supported, and maintained 24/7 (i.e., 24 hours a day, 7 days a week). This 24/7 requirement means that users might need help or have questions at any time, and a support desk that is available eight hours a day will not be sufficient support.
- It is also important to consider what reliability is needed in the system. A system that requires high reliability (e.g., a medical device or telephone switch) needs far greater planning and testing than one that does not have such high-reliability needs.

Security Requirements

- Security is the ability to protect the information system from disruption and data loss, whether caused by an intentional act (e.g., a hacker, a terrorist attack) or a random event (e.g., disk failure, tornado).
- Security is primarily the responsibility of the operations group—the staff responsible for installing and operating security controls, such as firewalls, intrusion-detection systems, and routine backup and recovery operations. Nonetheless, developers of new systems must ensure that the system's *security requirements* produce reasonable precautions to prevent problems; system developers are responsible for ensuring security within the information systems themselves.
- Developing security requirements usually starts with some assessment of the value of the system and its data. This helps pinpoint extremely important systems so that the operations staff is aware of the risks. Security within systems usually focuses on specifying who can access what data, identifying the need for encryption and authentication, and ensuring the application prevents the spread of viruses.

توانایی سیستم در برابر ایمن شدن از حمله ها و در برابر تخریب ها ... اتفاق می افته چقدر سیستم رو پا است یا چقدر سیستم تخریب میشه
این یکسریش برمیگرده به نصب یکسری از نرم افزارها مثلا فایروال نصب بکنیم
یکسریش هم برمیگرده به خود سیستم ینی معماری سیستم- کلاسهایی که تعریف میشه و دیتا Access ها ...
یکی از مهمترین کارها توی بحث این امنیت مشخص کردن ارزش اون سیستم و ارزش مولفه های اون سیستم --> وقتی این ارزشه مشخص باشه در میاد که به ازاش چقدر باید هزینه بکنیم

Security Requirements

Type of Requirement	Definition	Examples
System Value Estimates	Estimated business value of the system and its data	• The system is not mission critical, but a system outage is estimated to cost $50,000 per hour in lost revenue. • A complete loss of all system data is estimated to cost $20 million.
Access Control Requirements	Limitations on who can access what data	• Only department managers will be able to change inventory items within their own department. • Telephone operators will be able to read and create items in the customer file but cannot change or delete items.
Encryption and the Authentication Requirements	Defines what data will be encrypted Where and whether authentication will be needed for user access	• Data will be encrypted from the user's computer to website to provide secure ordering. • Users logging in from outside the office will be required to authenticate.
Virus Control Requirements	Requirements to control the spread of viruses	• All uploaded files will be checked for viruses before being saved in the system.

1: ارزش ها یا aset یا گنج های اون سیستم رو مشخص بکنیم ارزشش چقدر است --> پس اول باید valueاش رو تخمین بزنیم ینی چقدر هزینه بازیابی اون ارزش رو باید بدیم --> هرچی این هزینه بیشتر باشه می ارزه تدابیر امنیت بیشتری رو توی نرم افزار بدیم --> پس مشخص کردن ارزش اون aset و value است
2: این که کی چه کاری رو باید انجام بده در قالب access کنترل میاد --> ؟
3: احراز هویت باید انجام بشه و access به یوزر رو کنترل و تا حد مورد نیاز انجام بدیم
4: جلوگیری از ورود ویروس--> انتی ویروس گذاشت و از توسعه اون ویروس جلوگیری کرد

System Value(I)

- The most important computer asset in any organization is not the equipment; it is the organization's data.
- For example, suppose someone destroyed a mainframe computer worth $10 million. The mainframe could be replaced, simply by buying a new one. It would be expensive, but the problem would be solved in a few weeks. Now suppose someone destroyed all the student records at your university so that no one knew what courses anyone had taken or their grades. The cost would far exceed the cost of replacing a $10 million computer. The lawsuits alone would easily exceed $10 million, and the cost of staff to find paper records and reenter the data from them would be enormous and certainly would take more than a few weeks.

System Value(II)

- In some cases, the information system itself has value that far exceeds the cost of the equipment as well. For example, for an Internet bank that has no brick and mortar branches, the website is a *mission-critical system.* If the website crashes, the bank cannot conduct business with its customers. A mission-critical application is an information system that is literally critical to the survival of the organization. It is an application that cannot be permitted to fail, and if it does fail, the network staff drops everything else to fix it. Mission-critical applications are usually clearly identified so that their importance is not overlooked.
- Even temporary disruptions in service can have significant costs. The costs of disruptions to a company's primary website or the LANs and backbones that support telephone sales operations are often measured in the millions of dollars. Amazon.com, for example, has revenues of more than $10 million per hour, so if its website were unavailable for an hour or even part of an hour, it would lose millions of dollars in revenue. Companies that do less e-business or do telephone sales have lower costs, but recent surveys suggest losses of $100,000 to $200,000 per hour are not uncommon for major customer-facing information systems.

Access Control Requirements

- Some of the data stored in the system need to be kept confidential; some data need special controls on who is allowed to change or delete it. Personnel records, for example, should be able to be read only by the personnel department and the employee's supervisor; changes should be permitted to be made only by the personnel department.
- State who can access what data and what type of access is permitted: whether the individual can create, read, update and/or delete the data. The requirements reduce the chance that an authorized user of the system can perform unauthorized actions. One approach to address these requirements is through the use of *access control lists*, which can be implemented via the operating system or database management system.

Encryption and Authentication Requirements

- One of the best ways to prevent unauthorized access to data is *encryption*, which is a means of disguising information by the use of mathematical algorithms (or formulas).
- Encryption can be used to protect data stored in databases or data hat are in transit over a network from a database to a computer.
- The encryption and authentication requirements state what encryption and authentication requirements are needed for what data. For example, will sensitive data such as customer credit-card numbers be stored in the database in encrypted form, or will encryption be used to take orders over the Internet from the company's website? Will users be required to use a digital certificate in addition to a standard password?

Virus Control Requirements

- Address the single most common security problem: *viruses*. Studies have shown that almost 90 percent of organizations suffer a virus infection each year. Viruses cause unwanted events—some harmless (such as nuisance messages), some serious (such as the destruction of data).
- Any time a system permits data to be imported or uploaded from a user's computer, there is the potential for a virus infection.
- Many systems require that all information systems that permit the import or upload of user files to check those files for viruses before they are stored in the system.

Cultural and Political Requirements

- Are those specific to the countries in which the system will be used. In today's global business environment, organizations are expanding their systems to reach users around the world.

Type of Requirement	Definition	Examples
Customization Requirements	Specification of what aspects of the system can be changed by local users	• Country managers will be able to define new fields in the product database to capture country-specific information. • Country managers will be able to change the format of the telephone number field in the customer database.
Legal Requirements	The laws and regulations that impose requirements on the system	• Personal information about customers cannot be transferred out of European Union countries into the United States. • It is against U.S. federal law to divulge information on who rented what videotape, so access to a customer's rental history is permitted only to regional managers.

بستگی به اون قانون های که توی اون سیستم و مملکت وجود داره
1: کاملا جنبه های خاصی هستن که باید برای یوزرهای لوکال در نظر گرفته بشن مثلا کد ملی
2: قوانین و مقرراتی دولتی است که باید پیروی بکنیم از اونا و اونا حاکم بشن به سیستم

آیا این موارد مختص کشورهایی هستند که این سیستم در آنها استفاده خواهد شد. در محیط کسب و کار جهانی امروز، سازمان ها در حال گسترش سیستم های خود برای دسترسی به کاربران در سراسر جهان هستند.

Customization Requirements

- For global applications, the project team needs to give some thought to *customization requirements:* How much of the application will be controlled by a central group, and how much of the application will be managed locally?
- For example, some companies allow subsidiaries in some countries to customize the application by omitting or adding certain features. This decision has trade-off s between flexibility and control because customization often makes it more difficult for the project team to create and maintain the application. It also means that training can differ among different parts of the organization, and customization can create problems when staff moves from one location to another.
- Owing to the use of different languages, in some cases, specialized hardware that has been customized to the local culture is required. For example, having specialized keyboards makes sense for any language that does not use the typical Roman alphabet, e.g., Arabic, Hebrew, Greek, Japanese, Korean, Mandarin, or Russian. There are also emulators available for many different languages.

Legal Requirements

- Requirements imposed by laws and government regulations. System developers sometimes forget to think about legal regulations;
- unfortunately, forgetting comes at some risk because ignorance of the law is no defense. By formally considering legal regulations, you are less likely to overlook them.

Reference

- **Dennis, Wixon, Tegarden**, "**System Analysis and Design, An Object Oriented Approach with UML", 5th Edition, 2015.**