1- System calls provide the means for a user program to ask the operating system to perform tasks reserved for the operating system on the user program's behalf. System calls provide an interface to the services made available by an operating system. These calls are generally available as functions written in C and C++, although certain low-level tasks (for example, tasks where hardware must be accessed directly) may have to be written using assembly-language instructions. System calls are in 6 types:

• Process control
◦ create process, terminate process
◦ load, execute
◦ get process attributes, set process attributes
◦ wait event, signal event
◦ allocate and free memory

• File management
◦ create file, delete file
◦ open, close
◦ read, write, reposition
◦ get file attributes, set file attributes

• Device management
◦ request device, release device
◦ read, write, reposition
◦ get device attributes, set device attributes
◦ logically attach or detach devices

• Information maintenance
◦ get time or date, set time or date
◦ get system data, set system data
◦ get process, file, or device attributes
◦ set process, file, or device attributes

• Communications
◦ create, delete communication connection
◦ send, receive messages
◦ transfer status information
◦ attach or detach remote devices

• Protection
◦ get file permissions
◦ set file permissions

2- با توجه به دو ویژگی سرعت و امنیت اشاره شده، میتوان گفت که الف نیاز به مکانیزم shared memory و ب نیاز به مکانیزم message passing دارد چرا که:

Message passing is useful for exchanging smaller amounts of data, because no conflicts need be avoided. It is also easier to implement than is shared memory for inter computer communication. Shared memory **allows maximum speed** and convenience of communication, since it can be done at

memory transfer speeds when it takes place within a computer. Problems exist, however, in **the areas of protection** and synchronization between the processes sharing memory.

Shared memory: In the shared-memory model, processes use shared memory create() and shared memory attach() system calls to create and gain access to regions of memory owned by other processes. Recall that, normally, the operating system tries to prevent one process from accessing another process's memory. Shared memory requires that two or more processes agree to remove this restriction. They can then exchange information by reading and writing data in the shared areas. The form of the data is determined by the processes and is not under the operating system's control. The processes are also responsible for ensuring that they are not writing to the same location simultaneously

Message passing: In the message-passing model, the communicating processes exchange messages with one another to transfer information. Messages can be exchanged between the processes either directly or indirectly through a common mailbox. Before communication can take place, a connection must be opened. The name of the other communicator must be known, be it another process on the same system or a process on another computer connected by a communications network. Each computer in a network has a host name by which it is commonly known. A host also has a network identifier, such as an IP address. Similarly, each process has a process name, and this name is translated into an identifier by which the operating system can refer to the process. The get hostid() and get processid() system calls do this translation. The identifiers are then passed to the general purpose open() and close() calls provided by the file system or to specific open connection() and close connection() system calls, depending on the system's model of communication. The recipient process usually must give its permission for communication to take place with an accept connection() call. Most processes that will be receiving connections are special-purpose daemons, which are system programs provided for that purpose. They execute a wait for connection() call and are awakened when a connection is made. The source of the communication, known as the client, and the receiving daemon, known as a server, then exchange messages by using read message() and write message() system calls. The close connection() call terminates the communication.

-3

الف) سیاست: تعریف اولویت و نمایش نخ با بالاترین اولویت در هر لحظه
مکانیزم: پیاده سازی صفی از نخ ها به ترتیب اولویت(برای مثال FIFO برای شانس اجرا نخی که زودتر آمده)

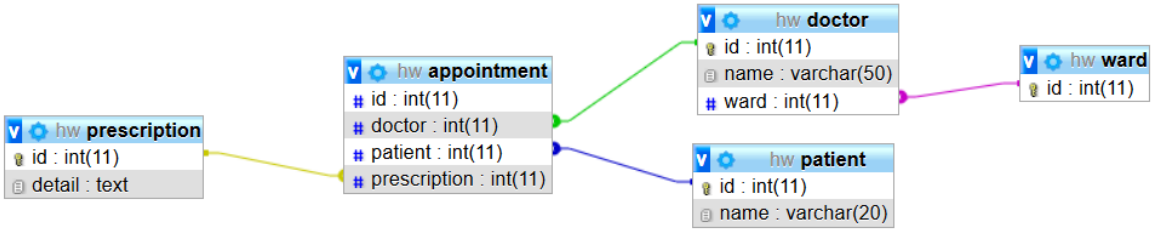
ب) سیاست: پیاده سازی سیستم ورود کاربران سهل، کاربردی و امن
مکانیزم: ذخیره سازی نام کاربری، ایمیل و پسورد(ترجیح بالا بر encrypt کردن آن در هنگام ذخیره) کاربران در هنگام ثبت نام و ارائه پنل مناسب برای ورود نام کاربری/ایمیل و پسورد و چک کردن محل ذخیره کاربران با اطلاعات وارد شده

4-

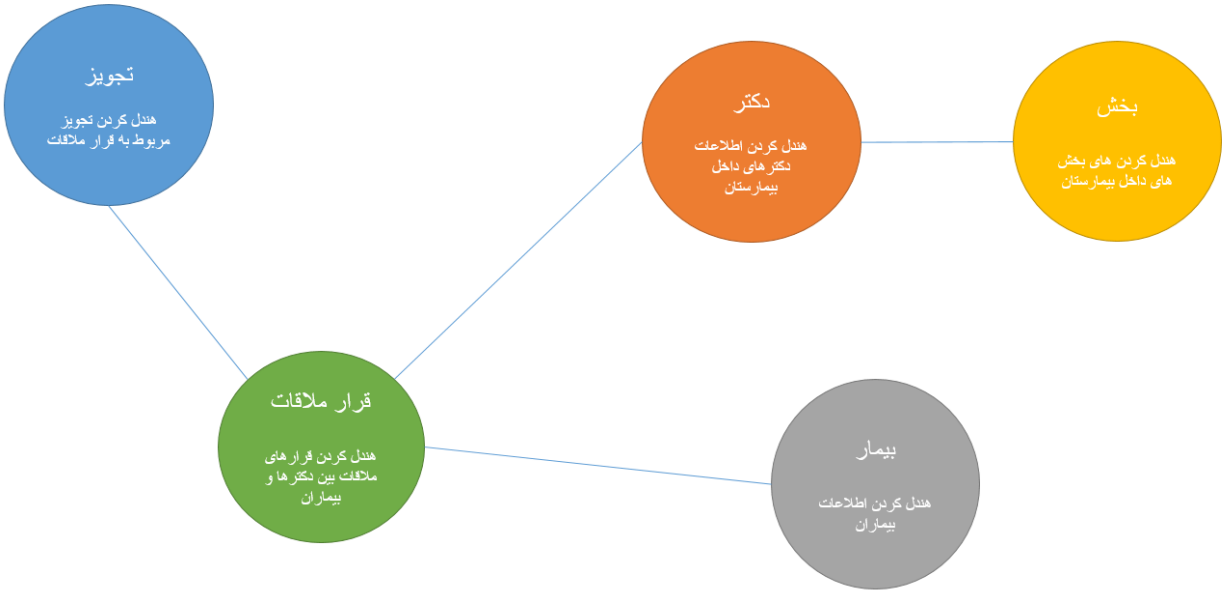
benefit: One benefit of the microkernel approach is that it makes extending the operating system easier. All new services are added to user space and consequently do not require modification of the kernel. When the kernel does have to be modified, the changes tend to be fewer, because the microkernel is a smaller kernel. The resulting operating system is easier to port from one hardware design to another. The microkernel also provides more security and reliability, since most services are running as user—rather than kernel—processes. If a service fails, the rest of the operating system remains untouched.

Downside: the performance of microkernels can suffer due to increased system-function overhead. When two user-level services must communicate, messages must be copied between the services, which reside in separate address spaces. In addition, the operating system may have to switch from one process to the next to exchange the messages. The overhead involved in copying messages and switching between processes has been the largest impediment to the growth of microkernel-based operating systems.

5- ساختار ساده شده پایگاه داده بیمارستان بصورت زیر است:



ساختارهای سرویس های کار با این دیتابیس هم بصورت زیر تعریف میشود:



ساختار بصورت monolithic پیاده میشود و داخل این ساختار همانطور که از شکل پیداست از سرویس های مخصوص و مجزا استفاده میشود. دلیل monolithic بودن آن، بالا بردن سرعت و دسترسی سریعتر سرویس ها به هم بوده و دلیل سرویس های مجزا، قابلیت اعتماد و در دسترس بودن بیشتر میباشد تا در صورت down شدن هر سرویس ضربه کمتری به ساختار کل وارد شود