

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

" ای خدا ، من باید از نظر علم از همه برتر باشم، تا مبادا که دشمنان مرا از این راه طعنه زنند. باید به آن سنگدلانی که علم را بهانه کرده و به دیگران فخر می‌فروشند، ثابت کنم که خاک پای من هم نخواهند شد. باید همه آن تیره دلان مغرور و متکبر را به زانو در آورم، آنگاه خود خاضع‌ترین و افتاده‌ترین مرد روی زمین باشم. ای خدای بزرگ آنها که از تو می‌خواهم ، چیزهایی است که فقط می‌خواهم در راه تو به کار اندازم و تو خوب میدانی که استعداد آنها داشته‌ام. از تو می‌خواهم مرا توفیق دهی که کارهایم ثمر بخش شود و در مقابل خسان سر افکنده نشوم . "

دکتر مصطفی چمران



دانشکده فنی مهندسی - گروه کامپیوتر

پایان نامه کارشناسی

عنوان پایان نامه

طراحی و پیاده سازی سامانه مدیریت تجهیزات الکترونیکی و کامپیوتری

دانشجو :

محمد سعید توانا

استاد راهنما :

دکتر مریم طایفه محمودی

سال:

تیر ماه ۱۳۹۵

ابتدا بر خود می دانم که از خداوند مهربان که همیشه در تمام مراحل زندگی همراه من بوده است و همیشه بهترین ها را برای من پیش آورده است تشکر نمایم ؛ این پایان نامه که حاصل تلاش چند ماهه بنده بوده است را تقدیم می کنم به تمامی دانشمندان و بزرگان کشور عزیزمان ایران ، به رهبر عزیز کشورمان حضرت آیت الله خامنه ای (مد ظله العالی) که در سایه ی رهبری با درایت ایشان مردمان کشورم در امنیت و آرامش زندگی می کنند . در آخر این پایان نامه را تقدیم می کنم به مادر عزیزم خانم فرحناز ملک کوهی و پدر عزیزم محمد توانا که در راه تحصیل من از هیچ کمکی کوتاهی نکردند و من در دامن پر مهرشان پرورش پیدا کردم.

محمد سعید توانا

تیر ماه ۱۳۹۵

همیشه از خداوند شاکر هستم که همیشه در کنارم بوده است و همیشه من را در تمام مراحل مختلف زندگی یاری نموده است ؛ از مادر عزیزم خانم فرحناز ملک کوهی و پدر عزیزم آقای محمد توانا که همیشه در کنارم بوده اند و بهترین شرایط را برای من در زندگی فراهم نموده اند کمال تشکر و قدردانی را دارم همچنین از مادر بزرگ عزیزم خانم زهرا سید هندی پور تشکر می کنم که بسیار برای من زحمت کشیدند ؛ از برادر عزیزم آقای محمد جواد توانا و از خواهر عزیزم خانم ریحانه توانا کمال تشکر را دارم ؛ از تمامی اساتیدی که در طول مدت تحصیل؛ من را در فریضه علم آموزی یاری کردند تشکر می نمایم ، در آخر بر خود می دانم که از سرکار خانم مریم محمودی ،استاد ارجمند تشکر و قدر دانی کنم که با راهنمایی های درست من را در این پروژه یاری نمودند .

محمد سعید توانا

تیر ماه ۱۳۹۵

## فهرست

### ۱- مقدمه

- ۱-۱ معرفی موضوع و اهمیت آن ----- ۱۰
- ۱-۲ پرسش های تحقیق ----- ۱۱
- ۳-۱ فرضیه های تحقیق ----- ۱۲
- ۴-۱ روش اجرای پژوهش ----- ۱۴
- ۵-۱ محدودیت های تحقیق ----- ۱۵
- ۶-۱ اهداف تحقیق ----- ۱۷
- ۷-۱ جنبه های جدید و نوآوری تحقیق ----- ۱۸
- ۸-۱ واژگان کلیدی تحقیق ----- ۱۹
- ۹-۱ سازماندهی پایان نامه ----- ۲۰

### ۲- مروری بر سامانه های مدیریتی موجود در زمینه سیستمهای الکترونیکی

- ۱-۲ سامانه های موجود در این زمینه ----- ۲۵
- ۲-۲ ارزیابی سامانه های موجود و استفاده ی آن برای پروژه ----- ۳۰

### ۳- چارچوب و معماری پیشنهادی برای سامانه مدیریت سیستمهای الکترونیکی

- ۱-۳ نمایش شماتیک از سیستم طراحی شده ----- ۳۳
- ۲-۳ زبان برنامه نویسی مورد استفاده و قابلیت های آن ----- ۳۶
- ۴-۳ پایگاه داده مورد استفاده و ویژگی های آن ----- ۴۵
- ۵-۳ معماری و فناوری های مورد استفاده ----- ۵۵
- ۶-۳ محیط برنامه نویسی مورد استفاده و ویژگی های آن ----- ۵۹
- ۷-۳ الگوها اصول مهندسی نرم افزاری ضروری برای انجام کار ----- ۶۰

## فهرست

### ۴- تحلیل و ارزیابی سامانه مدیریت سیستمهای الکترونیکی

- ۱-۴ تحلیل و ارزیابی کل سیستم ----- ۶۶
- ۲-۴ نمودارهای UML مربوط به طراحی سیستم ----- ۷۰
- ۳-۴ تحلیل مربوط به قابلیت توسعه دیتابیس ----- ۸۷

### ۵- جمع بندی و پیش بینی کارهای آتی

- ۱-۵ جمع بندی مطالب و نتیجه گیری ----- ۹۲
- ۲-۵ پیش بینی کارهای آتی ----- ۹۴

### ۶- مراجع

- ۱-۶ مراجع انگلیسی ----- ۹۷
- ۲-۶ مراجع فارسی ----- ۹۸
- ۳-۶ ادرس سایت ها ----- ۹۹

### پیوست

- مطالب و توضیحات اضافه ----- ۱۰۲
- فرم های مربوط به نرم افزار ----- ۱۶۰
- کد های پیاده سازی شده ----- ۱۶۳

## چکیده

در این پایان نامه اجزای ضروری جهت طراحی و پیاده سازی یک سامانه مدیریت تجهیزات ارائه می شود. بدین منظور، ابتدا به بررسی و مقایسه بین زبان های برنامه نویسی خواهیم پرداخت و نقاط قوت و نقاط ضعف هریک را بررسی خواهیم نمود، سپس با مروری بر قابلیت های انواع پایگاه های داده ، به انتخاب پایگاه داده مناسب برای اینکار می پردازیم. آنگاه ابزارها، معماری و فناوری های مورد لزوم بررسی می شوند. نهایتاً، نتایج طراحی ها در قالب دیاگرام های UML ارائه می شوند.



# فصل اول

## مقدمه

آنچه خواهید خواند :

۱-۱ معرفی موضوع و اهمیت آن

۲-۱ پرسش های تحقیق

۳-۱ فرضیه های تحقیق

۴-۱ روش اجرای پژوهش

۵-۱ محدودیت های تحقیق

۶-۱ اهداف تحقیق

۷-۱ جنبه های جدید و نوآوری تحقیق

۸-۱ واژگان کلیدی تحقیق

۹-۱ سازماندهی پایان نامه

## ۱-۱ معرفی موضوع و اهمیت آن

در این بخش از فصل اول به بررسی علت طراحی این سیستم خواهیم پرداخت و بررسی می کنیم که این گونه سیستم ها در کجا کاربرد دارند و چگونه باعث سرعت پیشرفت کار می شوند.

در ابتدای بحث باید بررسی کنیم که بدون سیستم های کامپیوتری وبدون کامپیوتر چقدر زندگی سخت می شود . نیاز کامپیوتر و سیستم های کامپیوتری را با ذکر یک مثال بیان می کنیم

فرض کنید که بانک های امروزی دارای سیستم الکترونیکی نبودند و شما برای انجام دادن هر تراکنش بانکی باید به بانک مراجعه می کردید. خوب ، برای انجام دادن کوچک ترین کاری باید ساعت ها در صف منتظر می ماندید و چقدر از وقتتان که با ارزشترین چیز است هدر می رود . همچنین برای جلوگیری از ازدحام بیش از حد باید تعداد شعبه های بانک بیشتر شود که این خود یعنی بالا بردن هزینه های مربوط به منابع انسانی و هزینه های جاری ؛ بدیهی است که این کار باعث افزایش هزینه ها می شود . گرچه در گذشته به دلیل نبود سیستم های کامپیوتری از روش سنتی در بانک ها استفاده می شد ولی به دلیل جمعیت اندک آن موقع نسبت این دوره این سیستم سنتی جواب گوی نیاز های آن زمانه بوده است . بدیهی است با توجه به مکانیزه شدن زندگی های این دوره زمانه پیاده کردن سیستم سنتی جواب گوی نیاز های امروزه نیست . بدیهی است با توجه به نیاز های امروزه نیاز به یک سیستم اتوماسیون و یک سیستم کامپیوتری احساس می شود. حالا اگر همین سیستم بانکداری را به صورت الکترونیکی پیاده سازی نماییم خواهیم دید که بسیاری از کار های بانکی را می توانیم با استفاده از بانکداری مجازی انجام دهیم لذا در هزینه ها بسیار صرفه جویی خواهد شد ، همچنین در زمان ، آلودگی هوا نیز تاثیر گذار خواهد بود . با توجه به علت های گفته شده دیدیم که سیستم های کامپیوتری چه تاثیر مستقیمی بر روی زندگی انسان ها دارد . البته لازم به ذکر است که طراحی این سیستم ها باید بسیار با دقت صورت گیرد و نیاز سنجی های آن باید به

طور کامل صورت گیرد . همچنین در اینجا باید ذکر کنم که نگهداری این سیستم های نیز هزینه هایی از قبیل هزینه طراحی و تولید ، هزینه ی پیاده سازی . هزینه ی نگهداری و سرور می باشد ؛ با تمام این هزینه های ذکر شده باز این هزینه ها از سیستم سنتی کمتر می باشد . با بیان علت های فوق در می یابیم که سیستم های کامپیوتری از جایگاه ویژه ای برخوردار است . در ادامه این بخش به بررسی نیاز سنجی مربوط به این سیستم خواهیم پرداخت و بررسی خواهیم کرد که چرا این سیستم طراحی شده است و نیاز سنجی مربوط به آن را بررسی خواهیم کرد .

در مراکزی که کامپیوتر های زیادی وجود دارد و سیستم های الکترونیکی زیادی وجود دارد مدیریت و نگهداری آن ها از اهمیت ویژه ای برخوردار است لذا با مدیریت درست آن ها و عیب یابی سیستم های معیوب می توانیم بسیاری از هزینه های جاری را کاهش دهیم . حال به بیان این موضوع خواهیم پرداخت که چرا این سیستم را طراحی نموده ایم . در مراکز خصوصی و در مراکز دولتی به دلیل نبود یک سیستم منظم در مدیریت سیستم های کامپیوتری ، ممکن است که عیب یابی یک سیستم به درستی صورت نگیرد و این خود باعث می شود که هزینه ها بالا برود زیرا فرد نمی تواند در عیب یابی یک سیستم موفق عمل کند و این باعث خطا در مدیریت می شود. همچنین ممکن است که که فرد غیر متخصص به سیستم ها دست بزند که این خود نیز باعث خطا و اشتباه می شود و هزینه های نگهداری را بالا می برد . لذا ما سیستمی طراحی نمودیم که در آن یک سابقه از کلیه سیستم های را نگه می دارد که شامل اطلاعاتی در مورد با آن سیستم می باشد و نشان می دهد که چه تغییراتی بر روی این سیستم چه از لحاظ سخت افزاری و چه از لحاظ نرم افزاری صورت گرفته است ؛ در واقع این سیستم به صورتی به مسئول فنی سیستم ها کمک می کند و یک پیشینه از سیستم را در اختیار کاربر و مسئول فنی قرار می دهد. و مسئول فنی با توجه به پیشینه موجود در سیستم نسبت به آن سیستم تصمیم خود را اتخاذ می کند. با استفاده از این سیستم خواهیم دید که به چه میزان در هزینه های نگهداری سازمان صرفه جویی خواهد شد . همچنین با استفاده از این

سیستم می توانیم سرعت عیب یابی و خطا را بالا ببریم و در واقع کیفیت خدمات بالا خواهد رفت که این خود باعث صرفه جویی در زمان و در هزینه خواهد شد. در بسیاری از مراکزی که سیستم های الکترونیکی و کامپیوتری دارند از اینگونه سیستم ها استفاده نمی شود و اینجانب با بررسی در تعدادی از مراکزی که دارای زیادی سیستم های کامپیوتری بودند دریافتیم که نیاز به همچنین سیستمی احساس می شود.

این سیستم در تمامی مراکز و شرکت هایی که دارای سیستم های کامپیوتری هستند قابل پیاده سازی و استفاده است و تعداد سرویس گیرنده های آن بسته به تعداد متخصصین موجود در آن سازمان و مرکز و متفاوت است و می تواند حداقل ۱ سرویس گیرنده باشد. با توجه به اینکه این سیستم مخصوص به مسئولین فناوری اطلاعات سازمان ها می باشد تعداد سرویس گیرنده های آن زیاد نخواهد بود، لذا سخت افزار بسیار قوی و گران برای بخش سرور نیاز نیست و در بیشتر مراکز که تعداد کامپیوتر های آن کمتر از ۱۰۰ عدد باشد یک کامپیوتر معمولی را می توان به عنوان سرور استفاده نمود. نکته دیگری که از اهمیت بالایی برخوردار است این است که سخت افزار قسمت سرور رابطه ی مستقیمی با تعداد سرویس گیرنده ها دارد و بسته به تعداد این سرویس گیرنده ها سخت افزار سرور قابل تغییر است.

این سیستم در تمامی مراکز دولتی و خصوصی جدا از حوزه کاری آن مرکز قابل استفاده می باشد و می توان از این سیستم در این مراکز استفاده نمود البته لازم به ذکر است که این سیستم باید در مراکزی استفاده شود که تعداد کامپیوتر های آن زیاد باشد چون بخش کوچکی از هزینه ها، مربوط به هزینه نگهداری خود این نرم افزار است و استفاده این سیستم در مراکز بزرگ که دارای سیستم های الکترونیکی زیادی هستند به صرفه خواهد بود و باعث کاهش هزینه ها نیز خواهد شد ولی در نقطه مقابل استفاده از این سیستم در مراکزی که تعداد سیستم های آنها اندک است (معمولا زیر ۱۰ عدد)؛ استفاده از این سیستم برای آن ها به صرفه نخواهد بود زیرا مدیریت چند عدد سیستم کامپیوتری

بسیار آسان تر از چند ده سیستم کامپیوتری می باشد. لذا پیشنهاد می شود که این سیستم را در مراکزی که بیشتر از ۱۰ عدد کامپیوتر دارند استفاده نمایید.

این سیستم نیز در بانک ها نیز کاربرد دارد چون بانک ها مراکزی هستند که از تعداد بیشتری از سیستم های کامپیوتری استفاده می کنند . همچنین این سیستم بسیار مناسب برای مکان ها و مراکزی است که مجهز به سیستم های اتوماسیون می باشند و این سیستم می تواند مکملی برای سیستم اتوماسیون باشد.

با مطالب بررسی شده در این فصل متوجه شدیم ، این سیستم در کجا کاربرد دارد و چرا این سیستم طراحی و پیاده سازی شده است . همچنین پی بردیم که سیستم های کامپیوتری چگونه می توانند به راحتی زندگی سرعت بخشند و از اتلاف زمان و هزینه جلوگیری نمایند.

در فصل های بعدی بیشتر در مورد با این سیستم صحبت خواهیم نمود و اجزای مختلف آن را بررسی خواهیم کرد و همچنین طراحی های مربوط به هر بخش را نشان خواهیم داد.

## ۱-۲ پرسش های تحقیق

در این بخش از فصل اول به مطرح کردن پرسش هایی خواهیم پرداخت که ما را در انجام این تحقیق و پیاده سازی پروژه کمک خواهد کرد.

برای جهت دهی به این پروژه و تحقیق باید سوالات زیر را مطرح کنیم و برای این سوالات پاسخ های منطقی و مناسب را پیدا کنیم.

- از چه زبان برنامه نویسی استفاده کنیم ؟
- از چه سیستم مدیریت پایگاه داده استفاده کنیم؟
- از چه ساختاری استفاده کنیم؟
- از چه معماری و فناوری هایی استفاده کنیم؟
- این سیستم در چه مکان هایی استفاده می شود؟
- آیا این سیستم باعث سهولت کار می شد؟
- آیا این سیستم قبلا تولید شده است؟
- آیا این سیستم را می توانیم بدون توجه به نوع مرکز استفاده کنیم ؟
- برای امنیت این سیستم چه کار هایی باید انجام دهیم؟
- این سیستم چه ویژگی های متمایزی از دیگر سیستم های موجود دارد؟
- آیا از سیستم های موجود می توانیم برای رفع نیاز استفاده کنیم ؟
- آیا واقعا این سیستم مورد نیاز است ؟
- استفاده از این سیستم به صرفه است؟

- آیا پیش بینی می شود که از این سیستم استقبال شود؟
- چه علت هایی باعث مخالف با این سیستم خواهد شد ؟
- پیش بینی می شود که این سیستم چگونه رشد خواهد کرد؟
- آینده این سیستم چگونه است ؟
- برای توسعه ی ای سیستم چه کار هایی باید صورت گیرد؟

سوالات مطرح شده در بخش فوق تعدادی سوال است که در فصل های مختلف این تحقیق به بررسی آنها خواهیم پرداخت.

### ۱-۳ فرضیه های تحقیق

در این بخش از فصل به فرضیه های موجود خواهیم پرداخت .

ابتدا برای انجام این تحقیق باید یک سری فرضیه داشته باشیم تا بتوانیم سیستم را طراحی کنیم و این تحقیق را انجام دهیم ، برای این کار ما ابتدا با نیاز سنجی هایی که از یک مرکز بزرگ انجام دادیم یک سری فرضیه ها را برای خود مشخص کردیم .

فرض می کنیم که این سیستم وجود ندارد و باید برای اولین بار این سیستم را تولید کنیم ؛ پس از ابتدا به بررسی نیاز های موجود خواهیم پرداخت . بخش عمده این نیاز ها را با استفاده از صحبت کردن با مسئولین مرکز به دست خواهیم آورد . همچنین برای درک عمیق از نیاز های موجود لازم است که مدتی را در این مرکز کار کنیم برای این کار بنده به مدت ۴ ماه نیز در این مرکز مشغول به کار شدم تا در جریان امور قرار گرفته و نیاز ها را دقیقتر بررسی کنم . ازجمله فرضیه هایی که برای این سیستم در نظر می گیریم این است ؛ این سیستم مورد نیاز است و حتما باید تولید شود تا باعث سهولت در کار ها شود. و این فرضیه باعث می شود که برای ساخت بهتر این سیستم و بررسی های خود دقت بیشتری داشته باشیم .

فرض می کنیم که مسئولان و مدیران مراکز نسبت به این سیستم نظر موافق خواهد داشت . این خود باعث می شود که کار با دقت بالاتری دنبال شود .

البته در اینجا لازم به ذکر است که برخی مدیران با فناوری مخالف هستند و دوست دارند که امورات به صورت سنتی انجام شود که ما فعلا از اینگونه مدیران چشم پوشی می کنیم و فرض می کنیم که تمامی مدیران با اجرای این سیستم در مرکز تحت مدیریت خود موافق هستند.

موارد ذکر شده فرضیاتی بود که در انجام این تحقیق موثر هستند.



## ۴-۱ روش اجرای پژوهش

در این بخش به بررسی روش اجرای پژوهش خواهیم پرداخت .

ابتدا برای بررسی و اجرای این پژوهش باید مدتی را در یک مرکز که دارای تعدادی سیستم الکترونیکی مشغول به کار شویم تا با شیوه انجام امور آشنا شویم تا بتوانیم تمامی جنبه های تحقیق را بررسی کنیم و یک نیازسنجی دقیق و مناسب داشته باشیم . در مرحله نیازسنجی باید بسیار دقت نمود ، زیرا اگر نیازسنجی دقیق و کامل باشد می توانیم سیستمی طراحی کنیم که جوابگوی تمامی نیاز های آن مرکز باشد . اگر نیازسنجی کامل و دقیق نباشد باعث می شود که در فاز های دیگر پیاده سازی و ساخت پروژه دچار اشکال شویم و شاید هم نتوانیم تغییر مورد نظر را اعمال کنیم لذا باید وقت بیشتری برای نیازسنجی مصرف نماییم . با استفاده از مطالعات میدانی می توانیم اطلاعاتی در مورد با تمایل به این سیستم را به دست بیاوریم . البته از اینترنت هم می توانیم استفاده کنیم تا به برخی از پرسش های خود پاسخ بدهیم .

همچنین می توانیم از کسانی که در حوزه تولید سیستم های کامپیوتری تجربه دارند استفاده کنیم و

بهترین روش برای انجام این پروژه را از آنها بیاموزیم و استفاده کنیم.

## ۵-۱ محدودیت های تحقیق

در این بخش به بررسی محدودیت های تحقیق خواهیم پرداخت .

برای این تحقیق محدودیت هایی نیز وجود دارد که بررسی تعدادی از آنها خواهیم پرداخت .

اولین محدودیتی که داریم محدودیت جمع آوری اطلاعات میدانی است ؛ زیرا معمولاً مراکز دولتی و یا خصوصی اجازه جمع آوری اطلاعات را به افراد غیر کارمند نخواهد داد . همچنین نمی توانیم در تمامی مراکز به جمع آوری اطلاعات بپردازیم چون این کار هزینه های بسیاری دارد و زمان بر خواهد بود. از جمله محدودتی هایی که وجود دارد این است که چون پروژه ها با استفاده از زبان برنامه نویسی جاوا است و دیتابیس ما اوراکل است نمی توانیم از مقالات و مطالب شرکت اوراکل استفاده کنیم زیرا این سایت ایران را تحریم کرده است و با استفاده از ip های ایران نمی توانیم به این سایت دسترسی پیدا کنیم.

## ۱-۶ اهداف تحقیق

در این بخش به بررسی اهداف تحقیق خواهیم پرداخت .

تحقیقی که انجام داده ایم شامل اهدافی است . هر تحقیقی برای اهدافی انجام می پذیرد که اهداف این تحقیق را در این بخش ذکر خواهیم نمود.

در این تحقیق سعی شده است که تعدادی از روش ها و ابزار های تولید نرم افزار بررسی شود و نقاط قوت و نقاط ضعف آن ها را بررسی کنیم تا بتوانیم بهترین روش و ابزار را برای تولید نرم افزار خود استفاده کنیم. همچنین این تحقیق هدف دیگری دارد که به بررسی ابعاد مختلف این سیستم می پردازد و یک بررسی بر سیستم های موجود نیز دارد.

همچنین این تحقیق را انجام می دهیم تا سیستم های مشابه موجود در این زمینه را بشناسیم و در رابط با آنها اطلاعات جمع آوری نماییم تا بتوانیم یک سیستمی تولید کنیم که دارای ویژگی های منحصر به فرد خود باشد .

دیگر هدف این تحقیق انتقال مفاهیم پایه برای تولید یک نرم افزار به خواننده می باشد . در این تحقیق سعی شده است که تمامی بخش های یک نرم افزار بررسی شود تا خواننده با خواندن این تحقیق بتواند یک دید برای تولید نرم افزار داشته باشد و متوجه روند تولید نرم افزار ها بشود.

## ۷-۱ جنبه های جدید و نوآوری تحقیق

در این بخش به بررسی جنبه های جدید و نوآوری های صورت گرفته خواهیم پرداخت .

در این تحقیق ما از بسیاری از منابع که عموماً بیشتر به صورت سایت بوده است استفاده کرده ایم و سعی کردیم که با نشان دادن نمودار ها و اشکال و تفسیر آنها مطالب را بهتر به خواننده انتقال بدهیم. همچنین سعی کرده ایم با مقایسه ی ابزار های مختلف و با نشان دادن موارد کاربر آنها بهترین ابراز های موجود را به خواننده معرفی کنیم . در این تحقیق سعی شده است که به خواننده دیدی متفاوت از نرم افزار را بدهیم تا علل و عوامل تولید و گسترش آنها را به راحتی درک کند .

در این تحقیق سعی براین شده است در کنار مطالب علمی و سنگین از دید خواننده ، مطالب ، با زبان و نوشتار راحت و قابل فهم به خواننده انتقال داده شود . با آوردن مثال های کاربری سعی شده که خواننده از خواندن این تحقیق خسته نشود و با آوردن نمودار ها و عکس ها سعی بر این شده است که خواننده به خواندن این تحقیق میل پیدا کند .

در بخش پیاده سازی سعی بر این شده است که فناوری ها و روش های جدید برای پیاده سازی استفاده شود و ساختار برنامه به طوری باشد که برای توسعه ی آن به مشکل برخوردیم و بتوانیم تغییرات را به راحتی در آن اعمال نماییم.

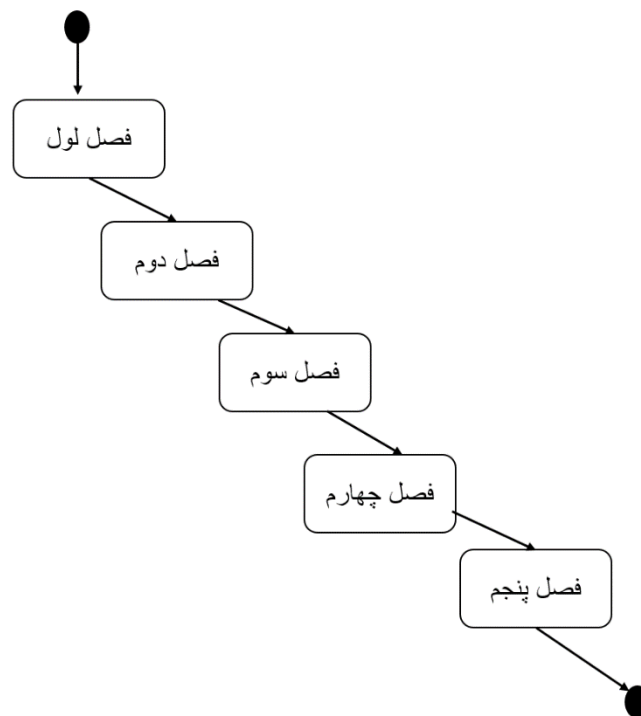
مطالب ذکر شده در قسمت فوق نوآوری های استفاده شده در این تحقیق و پروژه را نشان می دهد.

## ۸-۱ واژگان کلیدی تحقیق

در این قسمت به نام بردن واژگان کلیدی خواهیم پرداخت.

سیستم های توزیع یافته- معماری - معماری MVC - فناوری - نمودار ها توصیفی - زبان های برنامه نویسی - سیستم های مدیریت پایگاه داده - مهندسی نرم افزار - اصول مهندسی نرم افزار - محیط های برنامه نویسی - مقایسه سیستم های مدیریت پایگاه داده - مقایسه زبان های برنامه نویسی - برنامه نویسی شی گرا - الگوهای برنامه نویسی شی گرا - UML - نمودار های طراحی - سیستم های (Client /Server) - اصول طراحی پایگاه داده - مفاهیم پایگاه داده - مقایسه نرم افزار ها - مدیریت تجهیزات الکترونیکی .

## ۱-۹ سازماندهی پایان نامه



شکل ۱-۹-۱ فرایند مطالعه پایان نامه

در این بخش سازماندهی موجود در این پروژه را نام خواهیم برد و یک نگاه کلی به مباحثی که در این پروژه مطرح خواهد شد، خواهیم داشت.

در فصل اول بعد از خواندن مقدمه و آشنایی با موضوع و اهمیت اجرایی کردن این موضوع و اهمیت پیاده سازی آن و نیاز سنجی های مربوط به این سیستم با انواع سیستم های موجود در این زمینه که همانند سیستم طراحی شده است در فصل دوم آشنا خواهیم شد. در ادامه و در فصل سوم با تجزیه و تحلیل این سیستم آشنا خواهیم شد و این سیستم را از لحاظ کارایی بررسی خواهیم نمود، در این فصل به معرفی انواع زبان ها خواهیم پرداخت و علل انتخاب زبان برنامه نویسی پروژه را شرح خواهیم داد. در این فصل نیز ساختار و معماری مورد استفاده در این سیستم را شرح خواهیم داد و با انواع

فناوری های استفاده شده در این پروژه آشنا خواهید شد. در ادامه با اصول مهندسی نرم افزار استفاده شده در این پروژه آشنا خواهید شد.

در ادامه و در فصل چهارم نمودار های مربوط به طراحی این سیستم را نشان خواهید داد و به بررسی آنها خواهیم پرداخت و انواع نمودار های uml مربوط به پروژه را خواهیم دید . در فصل پنجم به جمع بندی مطالب خواهیم پرداخت و نتیجه گیری خواهیم نمود . در ادامه فصل کار های اتی این پروژه را نام خواهیم برد و آینده ی پیش روی سیستم را بررسی خواهیم نمود . شرح خواهیم داد که برای ارتقاء سطح کیفی سیستم چه کار هایی باید در آینده برای سیستم انجام گیرد.

با مطالعه مطالب فوق چارچوب کلی این پایان نامه را متوجه شدید و یک دید کلی از مطالبی که شرح داده می شود خواهید گرفت .

## فصل دوم

مروری بر سامانه های مدیریتی موجود در زمینه سیستمهای الکترونیکی

آنچه خواهید خواند :

۱-۲ سامانه های موجود در این زمینه

۲-۲ ارزیابی سامانه های موجود و استفاده ی آن برای پروژه



## ۲- ۱ سامانه های موجود در این زمینه

در این بخش از فصل دوم به معرفی تعدادی از سیستم های مدیریت تجهیزات الکترونیک خواهیم پرداخت .

همچنین سیستم های مدیریت وسایل مختلف را معرفی خواهیم نمود .

این سیستم ها برای سرویس دهی بهتر به مشتریان به وجود آمده اند و هرکدام سیاست های مربوط به

خود را دارند و هرکدام از سیستم هایی که معرفی می کنیم برای یک هدف واحد توسعه پیدا کرده اند ؛ اما

ذات آنها یکسان است و این است این سیستم ها همگی کار مدیریت بر روی وسایل را انجام می دهند . در

ادامه به بررسی و معرفی تعدادی از آنها خواهیم پرداخت و زمینه ی استفاده هر یک را بررسی خواهیم .

برای تولید این پروژه ابتدا سیستم های موجود را بررسی کردیم تا هم بیشتر با آنها آشنا شویم و بتوانیم از

آنها ایده نیز بگیریم . کاری که ما در این فصل انجام می دهیم سیستم های موجود را بررسی و معرفی می

کنیم و از آنها در پروژه خود استفاده می کنیم .

اولین سیستمی که می خواهیم بررسی کنیم سیستم مربوط به شرکت خودرو سازی بنز است .



شرکت بنز یکی از بزرگترین شرکت ها خودرو سازی آلمانی در جهان می باشد که خودروهایی

لوکس و با کیفیت تولید می کند . این شرکت کارخانه های متعددی در سطح جهان دارد و به تمامی کشور

ها ماشین صادر می کند و می توانیم ماشین های ساخت این شرکت را در تمامی نقاط دنیا ببینیم. این

شرکت برای کشور های مختلف با توجه به اقلیم آن کشور ماشین تولید می کند . برای مثال ماشین هایی

که این شرکت برای کشور های گرمسیری جنوب شرق اسیا تولید می کند و صادر می کند همگی فاقد

بخاری می باشند و در عوض سیستم سرمایش آن قوی تر از نمونه های دیگر است . و همچنین این شرکت

خودروهایی که برای مناطق سرد سیر می صادر می کند فاقد سیستم سرمایش است .

شرکت بنز پس از بررسی هایی که انجام داد و به دلیل تعدد محصولات این شرکت در دنیا سیستمی

یکپارچه را تولید نمود که اطلاعات تمامی نمونه های تولید شده از این شرکت را در خود نگه می دارد . این

سیستم حتی تعمیرات انجام گرفته را نیز در کامپیوتر خود ذخیره می کند و شما می توانید در هر جای دنیا

که باشید سابقه مربوط به خودرو های بنز را از طریق نمایندگی های این شرکت مشاهده فرمایید . این

سیستم را بنده از نزدیک مشاهده کرده ام ، سیستم جامعی است . این سیستم به شفافیت بازار خرید و

فروش این خودرو نیز کمک می کند. این سیستم انگیزه اصلی بنده برای انجام این تحقیق و پیاده سازی این

سیستم می باشد. و من از ایده ی این سیستم برای پیاده سازی این پروژه استفاده نموده ام.

شرکت دیگری که مورد بررسی قرار می دهیم شرکت خودروسازی بی ام و است.



شرکت بی ام و یکی دیگر از بزرگترین شرکت ها خودرو سازی آلمانی در جهان می باشد که خودروهایی لوکس و با کیفیت تولید می کند این خودرو ها بیشتر برای جوانان مناسب می باشد چون چهره و طراحی این ماشین ها جوان پسند می باشند. این شرکت کارخانه های متعددی در سطح جهان دارد و به تمامی کشور ها ماشین صادر می کند و می توانیم ماشین های ساخت این شرکت را در تمامی نقاط دنیا ببینیم. این شرکت برای کشور های مختلف با توجه به اقلیم آن کشور ماشین تولید می کند. برای مثال ماشین هایی که این شرکت برای کشور های گرمسیری جنوب شرق اسیا تولید می کند و صادر می کند همگی فاقد بخاری می باشند و در عوض سیستم سرمایش آن قوی تر از نمونه های دیگر است. و همچنین این شرکت خودروهایی که برای مناطق سرد سیر صادر می کند فاقد سیستم سرمایش است.

این شرکت خودرو سازی نیز همانند شرکت خودرو سازی بنز نیز دارای سیستم مدیریت خودرو ها می باشد و شناسنامه ای از مشخصات و ویژگی های خودرو را شامل می شود. همچنین این سیستم تعمیرات صورت گرفته بر روی خودرو را در خود نگه می دارد و این باعث شفافیت بازار خرید و فروش این خودرو می شود.

این سیستم نیز یک سیستم یکپارچه می باشد و شما در تمامی نقاط جهان هر منطقه ای نمایندگی این شرکت وجود داشته باشد می توانید شناسنامه خودروی خود را مشاهده کنید. همچنین در این سیستم گارانتی مربوط به خودرو ها نیز موجود می باشد.

شرکت دیگری که به بررسی آن خواهیم پرداخت شرکت تولید لوازم خانگی bosch است.

در ادامه ی این بخش به بررسی سیستم موجود در این شرکت خواهیم پرداخت .

این شرکت یک شرکت بزرگ تولید لوازم خارجی می باشد . این شرکت در تمامی کشور های دنیا شعبه و

نماینده گی دارد و به تمام کشور های جهان محصولات خود را صادر می کند . این شرکت برای جلوگیری از

تقلب و واردن شدن جنس تقلبی با نام شرکت خود سیستمی را طراحی نموده است که در این سیستم

شماره هایی به محصولات تولید شده این شرکت اختصاص داده شده است و با استفاده از این شماره می

توانیم اطلاعات مربوط به اصل بودن و یا تقلبی بودن محصولات را پیگیری کنیم . همچنین این سیستم

اطلاعات جامعی از محصولات را در خود نگه می دارد . این سیستم برای استفاده از گارانتی اجناس نیز

طراحی شده است و نشان می دهد که محصول مورد نظر تعمیر شده است یا خیر ؛ این سیستم یک

شناسنامه فنی از محصولات را در خود نگه می دارد و هر جا که لازم باشد می توان از آن استفاده نمود.

با مطالعه ی این سیستم و علل طراحی آن توانستیم سیستمی را طراحی کنیم از تقلب در مراکز جلوگیری

می کند به این صورت که در این سیستم تمامی مشخصات از جمله شماره قطعات نیز نگهداری می شود و

اگر کسی بخواهد قطعه ای را با قطعه کهنه تعویض نماید ؛ با یک بررسی ساده می توانیم جلوی این کار را

بگیریم.

بررسی این سیستم به ما در ایده های این پروژه بسیار کمک نمود .



شرکت دیگری که به بررسی آن خواهیم پرداخت شرکت apple می باشد .

این شرکت که یک شرکت آمریکایی می باشد و گوشی و کامپیوتر های هوشمند تولید می کند و تقریبا در تمامی کشور های جهان نمایندگی دارد. این شرکت برای فعال سازی تمامی محصولات خود از یک شناسه استفاده می کند که به آن apple id گفته می شود.. تمامی استفاده کنندگان محصولات این شرکت باید این شناسه را داشته باشند تا بتوانند از محصول خود استفاده کنند . البته شایان ذکر است که ساخت این شناسه کاربری توسط خود کاربر انجام می گیرد اما تمامی این شناسه ها در سرور های این شرکت ذخیره سازی می شوند و با استفاده از این شناسه می توان کلیه اطلاعات محصولات این شرکت را بازیابی نمود . البته در این جا باید به مسئله امنیت در این سیستم اشاره نمود که شرکت سازنده ی این محصولات می تواند به راحتی به تمامی اطلاعات حساب کاربران خود دسترسی پیدا کند. این سیستم یک سیستم هوشمند هست که تمامی فعالیت های کاربران خود را ذخیره نماید . همچنین این سیستم همانند سیستم هایی که در بخش های قبل توضیح داده شده برای گارانتی نیز کاربرد دارد . مدت و نوع گارانتی را نیز در خود نگه می دارد و از تقلب و خطا جلوگیری می کند. البته شرکت های دیگری نیز وجود دارند که سیستم های مشابه با این را دارند همانند شرکت Samsung که در زمینه تولید گوشی های هوشمند و لوازم خانگی فعالیت دارد .

شرکت هایی که مورد بررسی قرار گرفت شرکت هایی بودند که تقریبا همه ی افراد با آنها آشنایی دارند و شاید از محصولات آنها نیز استفاده کرده اند . هر کدام از این شرکت ها سیستم های مربوط به خود را دارند

و متناسب با نیاز های خود این سیستم را پیاده سازی کرده اند ؛ اما چیزی که بین همه ی این سیستم های مشترک است این است که همه ی این سیستم های مدیریتی هستند و باعث سهولت در مدیریت محصولات می شوند و روند خدمت رسانی شرکت ها به مشتریان خود را آسان می کنند.

در بخش بعدی از این فصل به بررسی تاثیرات این سیستم ها بر روی پروژه صحبت خواهیم نمود.

## ۲,۲ ارزیابی سامانه های موجود و استفاده ی آن برای پروژه

در این بخش از فصل دوم به بررسی ایده هایی که از شرکت های مطرح شده که در بخش قبل مورد بررسی قرار دادیم ، و بر روی پروژه تاثیر گذاشته است ؛ سخن خواهیم گفت.

با بررسی هر یک از این شرکت ها ایده هایی به دست آوردیم همانند : ایده ی جلوگیری از تقلب و یا ایده ی مربوط به گارانتی کردن محصولات . این سیستم ها همانطور که بررسی شد خود نیز در کنار هدف اصلی خود سیستم های دیگری را نیز شامل می شوند به طوری که می توانیم از این سیستم های چندین منظور استفاده نماییم. ما با ذخیره کردن اطلاعات مربوط به محصولات و تجهیزات می توانیم مدیریت بسیار خوبی بر روی این محصولات داشته باشیم و برای مثال می توانی سیستم گارانتی را در این سیستم استفاده نماییم و یا می توانیم با توسعه دادن این سیستم این سیستم را به سیستم مالی نیز تبدیل کنیم . همانطور که مطرح شد از این سیستم می توانیم به عنوان چندین سیستم استفاده نماییم.

البته این سیستمی که ما تولید نموده ایم یک سیستمی هست که فقط برای تجهیزات الکترونیکی شرکت ها و مراکز استفاده می شود و مشتریان این سیستم مردم نیستند و کاربران این سیستم ؛ مدیران و مسئولین فناوری اطلاعات آن مرکز می باشند.

همانطور که در دو بخش قبل بررسی شد؛ با بررسی سیستم های دیگر می توانیم ایده برداری کنیم برای تولید یک سیستم جدید . لذا بررسی های قبل از تولید یک پروژه از جایگاه ویژه ای برخوردار است و ما در تولید این پروژه ، سیستم های موجود را بررسی کردیم و از هر کدام از سیستم ها ایده هایی را برداشت کردیم سعی کردیم که تمامی این ایده ها را در کنار هم و در قابل یک سیستم با توجه به نیاز خود یکجا جمع کنیم.

در فصل های آینده به بررسی قسمت های مختلف این سیستم خواهیم پرداخت .

## فصل سوم

### چارچوب و معماری پیشنهادی برای سامانه مدیریت سیستمهای الکترونیکی

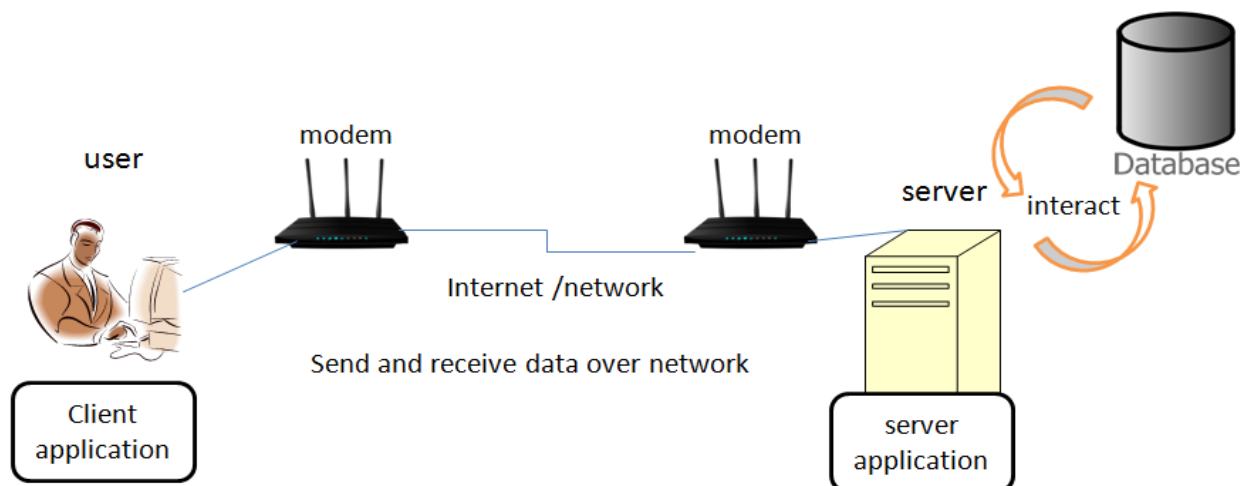
آنچه خواهید خواند :

- ۳-۱ نمایش شماتیک سیستم طراحی شده
- ۳-۲ زبان برنامه نویسی مورد استفاده و قابلیت های آن
- ۳-۳ پایگاه داده مورد استفاده و ویژگی های آن
- ۳-۴ معماری و فناوری های مورد استفاده
- ۳-۵ محیط برنامه نویسی مورد استفاده و ویژگی های آن
- ۳-۶ الگوها اصول مهندسی نرم افزاری ضروری برای انجام کار



### ۱-۳ نمایش شماتیک سیستم طراحی شده

در این بخش از فصل سوم به بررسی شماتیک سیستم خواهیم پرداخت و اجزاء مختلف آن را شرح خواهیم داد.



شکل ۱-۳-۱ نمایش شماتیک سیستم

در بخش اول به بررسی قسمت سرویس گیرنده می پردازیم. در این بخش که رابط کاربری قرار دارد ، کاربر می تواند از طریق این بخش با سیستم در ارتباط باشد . در واقع واسط کاربر پیچیدگی های سیستم را از کاربر پنهان می کند و اطلاعات را به کاربر نشان می دهد.

این سیستم چون یک سیستم توزیع شده است و بخش سرویس گیرنده از سرویس دهنده جدا می باشد ، برای ارتباط بین اجزا از بستر شبکه استفاده می شود و سرویس گیرنده ها باید IP سرور را داشته باشند تا بتوانند با آن ارتباط برقرار کنند .همچنین اجزای این سیستم که شامل سرویس دهنده و سرویس گیرنده ها است باید در یک شبکه باشند و برای این کار ما از modem استفاده می کنیم تا بتوانیم چند کامپیوتر را از طریق خط مخابرات به هم متصل نماییم . همانطور که در شکل فوق می بینید ۲ کامپیوتر به هم متصل

شده اند و می توانند با یکدیگر تعامل داشته باشند . البته لازم به ذکر است که اگر سرور در جایی دیگر و خارج از سازمان قرار می گیرد ، ip آن باید اعتبار داشته باشد تا بتوانیم از طریق اینترنت به سرور متصل شویم .

در بخش سرور ما نرم افزار سرور را داریم که به سرویس گیرنده ها و یا همان کاربر ها سرویس می دهد. سرور کامپیوتری است که دارای توان پردازشی بالاتری نسبت که سرویس گیرنده ها می باشد و قابلیت این را دارد که به صورت ۲۴ ساعته روشن بماند، البته این سیستم چون در ساعات اداری کار می کند ؛ می توانیم سرور را در ساعات غیر اداری خاموش نماییم ، اما سخت افزار سرور باید قوی باشد تا بتواند در زمان مناسب به سرویس گیرنده ها پاسخ بدهد.

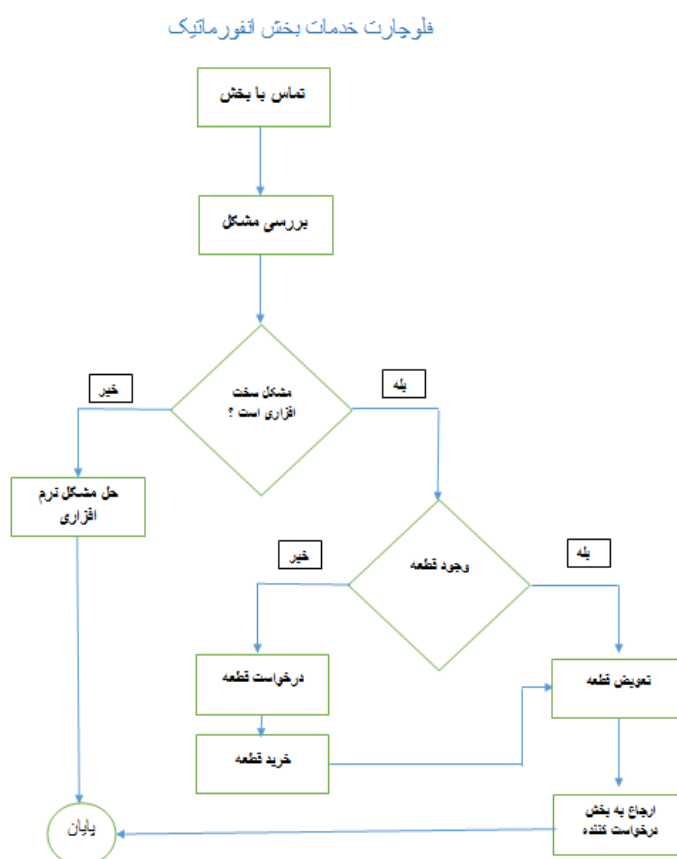
بخش آخری که به شرح آن می پردازیم بخش مربوط به پایگاه داده است . در این سیستم، پایگاه داده در سرور قرار می گیرد و نرم افزار سرور به آن متصل می شود و داده ها را ذخیره و یا می خواند . در این سیستم هنگامی که کاربر می خواهد اطلاعاتی را بنویسد و یا بخواند درخواست کاربر توسط نرم افزار کابر به نرم افزار سرور، تحت بستر شبکه ارسال می شود و نرم افزار تحت سرور با بررسی در خواست آن ، با اتصال به دیتابیس اطلاعات مورد نظر کاربر را ذخیره و یا می خواند و آن را به کاربر نشان می دهد.

در ادامه جدول مربوط به مشخصات نرم افزار را خواهیم دید.

جدول ۱-۳-۱ مشخصات مربوط به سیستم

مشخصات سیستم
قابلیت ارسال و دریافت تحت بستر شبکه
قابلیت سرویس دهی هم زمان به چند کاربر
قابلیت درج اطلاعات
قابلیت حذف اطلاعات
قابلیت مشاهده اطلاعات
قابلیت ویرایش اطلاعات
امکان بررسی اعتبارسنجی کاربران و مسئولان سیستم
امکان ذخیره ی تمامی اعمال انجام شده در پایگاه داده log server
امکان درج توضیحات برای تجهیزات
امکان مشاهده توضیحات برای هر کامپیوتر

در این پروژه ابتدا نرم افزار تحت سرور را با زبان برنامه نویسی جاوا نوشتیم و رابط های آن را با پایگاه داده برقرار نمودیم و سرویس های مورد نظر خود که شامل قابلیت های سیستم می شود را نوشتیم . بعد از اتمام بخش سرور به پیاده سازی و طراحی بخش پایگاه داده پرداختیم و جداول را در پایگاه داده با استفاده از دستورات مربوط به پایگاه داده ایجاد نمودیم . در بخش بعدی از پیاده سازی این پروژه به پیاده سازی بخش کاربر پرداختیم و توابع مورد استفاده برای ارسال درخواست به سمت سرور را پیاده سازی کردیم البته لازم به ذکر است که این پروژه در ابتدا به صورت آزمایشی مورد استفاده قرار می گیرد ؛ به همین علت بخش واسط کاربر آن متنی می باشد که بعد از انجام مراحل تست به محیط گرافیکی تبدیل خواهد شد.



شکل ۳-۱-۲ فرایند عیب یابی در مراکز

در این شکل ۳-۱-۲ فرایند عیب یابی به صورت شماتیک نشان داده شده است . همانطور که می بینید تمام حالات برای یک سیستم در آن در نظر گرفته شده است .

### ۲-۳ زبان برنامه نویسی مورد استفاده و قابلیت های آن

در این بخش از فصل سوم به علت انتخاب زبان برنامه نویسی در این پروژه خواهیم پرداخت . در این سیستم چون این سیستم چند کاربره ( multi user ) است و درواقع یک بخش نرم افزار به عنوان سرور است و بخش دیگر به عنوان سرویس گیرنده است باید از زبان برنامه نویسی استفاده نمود که قابلیت استفاده در سیستم های چند کاربره را داشته باشد. انتخاب یک زبان برنامه نویسی متناسب با نوع پروژه است و نمی توانیم بگوییم که یک زبان برنامه نویسی برای تمام استفاده ها متناسب است ؛ هر زبان برنامه نویسی یک سری نقاط قوت دارد و یک سری نقاط ضعف دارد . همچنین زبان های برنامه نویسی توسط یک سری از فاکتور ها مورد بررسی قرار می گیرند که توسط این فاکتور ها می توانیم متناسب با پروژه مورد نظر بهترین زبان را مشخص کنیم .

در ادامه تعدادی از این فاکتور ها را به همراه توضیح مختصر بیان خواهیم نمود. برای مطالعه ی بیشتر می توانید به کتاب هایی که در زمینه مقایسه زبان ها نوشته شده اند مراجعه فرمایید.

فاکتور های قابل بررسی :

#### قابلیت خوانایی ( readability )

قابلیت خوانایی به این معنا میباشد که خواندن کد توسط برنامه نویس آسان است و یا سخت و یا

اینکه می توانیم بگوییم که برنامه نویس با دیدن کد متوجه معنا و مفهوم کد می شود یا خیر؛ در

ادامه قطعه کدی که قابلیت خوانایی پایینی دارد و قطعه کدی که قابلیت بالایی دارد نشان داده

می شود.

## Objective – C

```
1. #import <Foundation/Foundation.h>
2.
3. int main (int argc, const char * argv[])
4. {
5.     NSAutoreleasePool *pool = [[NSAutoreleasePool alloc] init];
6.     NSLog (@"Hello, World!");
7.     [pool drain];
8.     return 0;
```

## Swift

```
Import uikit
Var s= “ hello world “
Print (s)
```

همانطور که مشاهده می کنید زبان برنامه نویسی Swift از خوانایی بهتری برخوردار است .

## قابلیت نوشتن ( write ability )

قابلیت نوشتن در زبان های برنامه نویسی به این معنا می باشد که برنامه نویسی بتواند منظور خود

را با تعدا کاراکتر های کمتر و یا با تعداد خط کد های کمتر بنویسد ، بدیهی است که سرعت کامپایل

برنامه بیشتر می شود. در ادامه مثالی از این قابلیت می زنیم .

### Case 1 :

```
Int counter =5;
Counter=counter +10;
```

### Case 2:

```
Int counter =5;
Counter+=10;
```

همانطور که می بینید در حالت ۲ از تعداد کاراکترهای کمتری استفاده شده است و این خود باعث می شود که برنامه نویس بتواند منظور خود را به راحتی بنویسد و همچنین سرعت کامپایل نیز بالا می رود.

### قابلیت اطمینان ( reliability )

قابلیت اطمینان به این معنا می باشد که برنامه ما در شرایط مختلط هنگ نکند و یا اصطلاحاً دچار Crash نشود و این قابلیت خود نیز دارای فاکتورهایی می باشد که چند تا از آن ها را در ادامه توضیح خواهیم داد.

### مدیریت خطا ( Exception handling )

زبان های برنامه نویسی باید قابلیتی داشته باشند که خطاهایی که در زمان اجرای برنامه رخ

می دهد را مدیریت و برنامه ریزی کنند و این امر با استفاده از `try _ throw _ catch`

در بیشتر زبان های برنامه نویسی صورت می گیرد.

### مدیریت حافظه و اشیا ( Memory management )

زبان های برنامه نویسی باید این قابلیت را داشته باشند که مدیریت حافظه را به صورت خودکار انجام

دهند . که معمولاً در بیشتر زبان ها این عمل به صورت خود کار صورت می گیرد.

### تعامد ( Orthogonality )

تعامد در زبان های برنامه نویسی به این صورت می باشد که برنامه نویس یک منظور را به

چندین روش از `syntax` انجام دهد ، زبان هایی که تعامد بالایی دارند یعنی یک مفهوم از برنامه

نویسی را فقط یا یک یا دو روش از `syntax` می توان انجام داد ؛ برای مثال در زبان برنامه نویسی

جاوا حلقه ی تکرار را فقط به دو روش می توان نوشت (for / for each) ؛ در حالی که در زبان

برنامه نویسی پایتون می توان حلقه تکرار را به چندین روش مختلف نوشت . این قابلیت به طور

نسبی است . زبان هایی که تعامل بالایی دارند عیب یابی و خطا در آن ها راحت تر می باشد.

### **قابلیت تعامل با دیتابیس های مختلف ( database connectivity )**

این قابلیت به آن معنا می باشد که زبان برنامه نویسی این قابلیت را داشته باشد با نرم افزار های

مدیریت دیتابیس های مختلف تعامل برقرار کند و بتواند کار کند.

### **امکان یادگیری آسان**

زبانی که یادگیری آن آسان باشد به سرعت گسترش می یابد و این خود باعث می شود که عیب های

آن برطرف شود و تکامل پیدا کند.

### **داشتن چارچوب (framework) های متعدد**

با استفاده از چارچوب های مختلف می توانیم کار های خود را به صورت خودکار و در یک قالب

انجام دهیم . زبان هایی که دارای چارچوب های مختلف می باشد برای هر منظور چارچوبی دارند

که این خود باعث افزایش سرعت ساخت پروژه و همچنین بالابردن سرعت عیب یابی و خطا می

شوند.

### **متن باز بودن ( open source )**

متن باز بودن ویژگی های زیادی دارد که در این جا یکی از ویژگی های آن را بیان خواهیم نمود

متن باز بودن به این معنا می باشد که شما تمامی قسمت های توابع و قسمت های زبان را می توانید

ببینید و همچنین می توانید در آن نیز تغییر ایجاد کنید ، بدیهی است که اولاً علم برنامه نویسی بالا می

رود و همچنین می توانیم قسمت های مختلف را براساس کار خود تغییر دهیم.

### **قابلیت برنامه نویسی با استفاده از پروتکل های شبکه (network programming)**

این قابلیت به ما اجازه می دهد که بتوانی برنامه های خود را در بستر شبکه توزیع کنیم و از

پروتکل های شبکه برای انتقال اطلاعات استفاده کنیم . زبانی که دارای این ویژگی می باشد،

می توان با آن برنامه های ( Client /Server ) تولید نمود.

### **زبان کامپایلری با قابلیت اتصال سریع (early binding )**

زبان های کامپایلری به زبان هایی گفته می شود که کامپایلر در آن کد نوشته شده را مستقیم به زبان

ماشین تبدیل می کند و بر روی سخت افزار اجرا می شود.

### **زبان مفسری با قابلیت اتصال کند (late binding)**

زبان های مفسری زبان هایی هستند که کامپایلرکد نوشته شده را به یک زبان میانی

( intermediate language ) تبدیل می کند و زبان میانی توسط یک مفسر به کد ماشین تبدیل می

شود. البته نمی توانیم بگوییم که یک زبان فقط مفسری است ویا بگوییم که فقط کامپایلری است. یک زبان

می تواند تا حدی رفتار کامپایلری و یا تا حدی نیز رفتار مفسری داشته باشد.

### **داشتن کتابخانه های غنی (full power library )**

زبان هایی که دارای کتابخانه هایی غنی هستند باعث می شود که کار برنامه نویس راحت شود

و بتواند به استفاده از کتابخانه ها کار های خود را به سرعت انجام دهد و در تعداد خط های کد نیز

صرفه جویی نیز می شود و از دیگر مزایای آن می توان به عیب یابی سریع برنامه اشاره نمود.



به توجه به ویژگی های بالا بعضی زبان ها برای طراحی صفحات وب مناسب هستند و بعضی دیگر نیز برای برنامه های سمت سرور مناسب هستند البته بعضی زبان های نیز در هر دور قسمت نیز فعالیت می کنند ولی در یکی از این فاز ها قوی تر هستند. با در نظر گرفتن ویژگی های بالا ، می توانیم زبان برنامه نویسی متناسب با پروژه را انتخاب کنیم. تعدادی از زبان ها برای نرم افزار های سیستمی (system software) مانند سیستم های عامل مناسب هستند و تعدادی برای سیستم های توزیع شده (distributed system) مناسب هستند و تعدادی برای طراحی صفحات وب مناسب هستند و تعدادی نیز برای نرم افزار های تحت وب ( web application ) مناسب هستند که در ادامه به معرفی مختصری از بعضی زبان ها خواهیم پرداخت.

## **java**

یک زبان برنامه نویسی مفسری و کامپایلری می باشد بسیار قدرتمند می باشد و قابلیت برنامه نویسی در تمامی زمینه ها را دارد ( java se /java ee/java me/java ce ) و قابلیت این را نیز دارد که بر روی تمامی پلتفرم ها ( windows /Mac/Linux ) اجرا شود. امروزه از این زبان برای تولید نرم افزار های بزرگ تحت وب استفاده می شود [5][3].

## **C#.**

یک زبان برنامه نویسی کامپایلری است که بیشتر برای تولید نرم افزار های تحت ویندوز استفاده می شود . همچنین همانند زبان جاوا نیز از برنامه نویسی شی گرای پشته‌بانی می کند.

## **c /c++**

این دو زبان که همانند هم می باشند با این تفاوت که در زبان C++ می توان برنامه نویسی شی گرای استفاده نمود .این زبان های کامپایلری می باشد و عموماً برای تولید نرم افزار های

سیستمی مانند سیستم عامل و کامپایلر ها و ... کاربرد دارند .

## **php**

یک زبان برنامه نویسی است که دارای چارچوب های کاری (frame work) متعددی است و در سمت سرور و کاربر استفاده می شود .

## **python**






















یک زبان برنامه نویسی مفسری می باشد که دارای قابلیت ها فراوانی است و امروزه بسیار گسترش پیدا کرده است. این زبان در تمامی زمینه ها مانند : وب ، نرم افزار های تحت ویندوز و تولید وب سایت ها کاربرد دارد.

## **html /css/java script**






















این زبان ها ، زبان هایی هستند برای ساخت واسط کاربری در نرم افزار های تحت وب استفاده می شود و این زبان های مفسری می باشد. مفسر این زبان ها مرورگر ها هستند.

با توجه به موارد ذکر شده در قسمت های قبل و با توجه به این که این نرم افزار تحت وب می باشد و دارای یک سرور و چندین سرویس گیرنده می باشند ، ما از زبان برنامه نویسی جاوا استفاده کرده ایم ، همچنین علت دیگری که ما این زبان را برای ساخت پروژه انتخاب نمودیم این است که جاوا در بیشتر زمینه ها فعالیت دارد و در آینده چون می خواهیم یک نرم افزار برای گوشی های هوشمند طراحی کنیم که با این سیستم تعامل داشته باشد ؛ و چون برنامه های اندروید نیز با جاوا نوشته می شود زبان برنامه نویسی جاوا گزینه ی مناسبی برای پیاده سازی این پروژه می باشد لذا ما از زبان برنامه نویسی جاوا استفاده کرده ایم.

در ادامه رتبه های این زبان برنامه نویسی را در سال گذشته خواهیم دید.

Language Rank	Types	Spectrum Ranking
1. Java	  	100.0
2. C	  	99.9
3. C++	  	99.4
4. Python	 	96.5
5. C#	  	91.3
6. R		84.8
7. PHP		84.5
8. JavaScript	 	83.0
9. Ruby	 	76.2
10. Matlab		72.4

شکل ۳-۲-۱ نمایش رتبه زبان های برنامه نویسی مختلف در سال ۲۰۱۵

Language Rank	Types	Spectrum Ranking	Spectrum Ranking
1. Java	  	100.0	100.0
2. C	  	99.9	99.3
3. C++	  	99.4	85.5
4. Python	 	96.5	93.5
5. C#	  	91.3	92.4
6. R		84.8	84.8
7. PHP		84.5	84.5
8. JavaScript	 	83.0	78.9
9. Ruby	 	76.2	74.3
10. Matlab		72.4	72.8

شکل ۳-۲-۲ نمایش رتبه زبان های برنامه نویسی مختلف در سال ۲۰۱۵



### ۳-۳ پایگاه داده مورد استفاده و ویژگی های آن

در پروژه ها، دیتابیس یا پایگاه داده ها از اهمیت بالایی برخوردار هستند چون داده ها در آن ذخیره می شوند لذا انتخاب نوع سیستم مدیریت پایگاه داده ها از اهمیت بالایی برخوردار می باشد ؛ برای این منظور باید شناخت دقیقی از ساختار و انواع دیتا بیس ها داشته باشیم . همچنین باید شناختی از انواع داده های خود نیز داشته باشیم تا بتوانیم بهترین دیتابیس را انتخاب کنیم .

برای مطالعه در مورد پایگاه داده می توانید به کتب پایگاه داده مراجعه فرمایید؛ در اینجا اسامی تعدادی از دیتا بیس ها ، و توضیح مختصری از هر یک را بیان خواهیم نمود.



یک نرم افزار مدیریت پایگاه داده می باشد که توسط شرکت ماکروسافت تولید شده است برای مدیریت پایگاه داده ها ؛ که قابلیت ذخیره سازی اطلاعاتی از نوع تصویر و صوت را نیز دارد. این نرم افزار تعامل بسیار خوبی با زبان های .net مانند C# دارد .



یک نرم افزار مدیریت پایگاه داده می باشد که توسط شرکت ماکروسافت تولید شده است برای مدیریت پایگاه داده ها ؛ این نرم افزار مناسب برای پایگاه داده های کوچک می باشد. این نرم افزار تعامل بسیار خوبی با زبان های .net مانند C# دارد .



یک نرم افزار مدیریت پایگاه داده می باشد که توسط شرکت اوراکل تولید شده است که این نرم افزار قابلیت پشتیبانی از داده های عظیم را دارد ، همچنین بیشتر در دیتابیس هایی استفاده می شود که دارای داده هایی از نوع ویدئو و صوت می باشد . این پایگاه داده تعامل بسیار خوبی با زبان برنامه نویسی جاوا دارد .



یک نرم افزار مدیریت پایگاه داده می باشد که توسط شرکت اوراکل تولید شده است ؛ و برای دیتا بیس های وب سایت ها و نرم افزار هایی که پایگاه داده های سبک دارند استفاده می شود . این نرم افزار نرم افزار سبکی است و تعامل بسیار خوبی با زبان برنامه نویسی جاوا دارد.



یک نرم افزار مدیریت پایگاه داده می باشد، با این تفاوت که مبنای آن همانند دیگر سیستم ها، رابطه ای نیست و بر مبنای no sql می باشد که برای داده های عظیم مناسب می باشد . برای مطالعه بیشتر به پیوست مراجعه فرمایید. .

در ادامه جدولی که شامل مقایسه میان پایگاه داده ها می باشد را خواهیم دید و علل استفاده از دیتابیس مورد استفاده در این پروژه را بیان خواهیم نمود.

جدول ۳-۳-۱ مقایسه ی میان سیستم های مدیریت پایگاه داده

نام سیستم	ویژگی	نقاط ضعف ❌	نقاط قوت ✅
<p>Sql server</p> 	<p>مدیریت پایگاه داده های متوسط که دارای متن بیشتر نسبت به فایل های حجیم</p> <p>مانند صوت و فیلم هستند</p> <p>RDBMS</p>	<p>تعامل پایین با زبان های غیر net.</p> <p>تعامل خوبی با زبان برنامه نویسی جاوا ندارد .</p>	<p>مناسب برای زبان های برنامه نویسی</p> <p><b>Net.</b></p> <p>مانند : <b>c#</b></p>
<p>Access</p> 	<p>مدیریت پایگاه داده های کوچک و مناسب برای پروژه های کوچک</p> <p>RDBMS</p>	<p>تعامل پایین با زبان های غیر net.</p> <p>نا توانی در مدیریت پایگاه داده های عظیم و بزرگ</p>	<p>مناسب برای زبان های برنامه نویسی</p> <p><b>Net.</b></p> <p>مانند : <b>c#</b> و مدیریت پایگاه داده های کوچک</p>
<p>Oracle</p> 	<p>مدیریت پایگاه داده های عظیم و توزیع شده که دارای داده های با حجم زیاد</p> <p>مانند صوت و فیلم هستند.</p> <p>RDBMS</p>	<p>تعامل پایین با زبان های Net.</p> <p>و بالا بودن هزینه های نگهداری و دشوار بودن یادگیری</p>	<p>تعامل بسیار بالا با زبان برنامه نویسی جاوا (از یک کمپانی)</p> <p>پایداری داده ها بسیار زیاد است</p>
<p>Mysql</p> 	<p>مدیریت پایگاه داده های سبک و کم حجم و بیشتر برای صفحات وب استفاده می شود</p> <p>RDBMS</p>	<p>دیتابیس های حجیم و بزرگ را به خوبی مدیریت نمی کند</p> <p>و تعامل پایین با زبان های Net.</p>	<p>رایگان بودن ، نرم افزار آن کم حجم است و یادگیری آن آسان است تعامل بالا با زبان برنامه نویسی جاوا</p>

با توجه به بررسی هایی که در بالا بر روی نرم افزار های مدیریت پایگاه داده های مختلف صورت گرفت می توانیم پایگاه داده مناسب با این پروژه را انتخاب نماییم . این سیستم به دلیل آن دارای کاربر های زیادی در

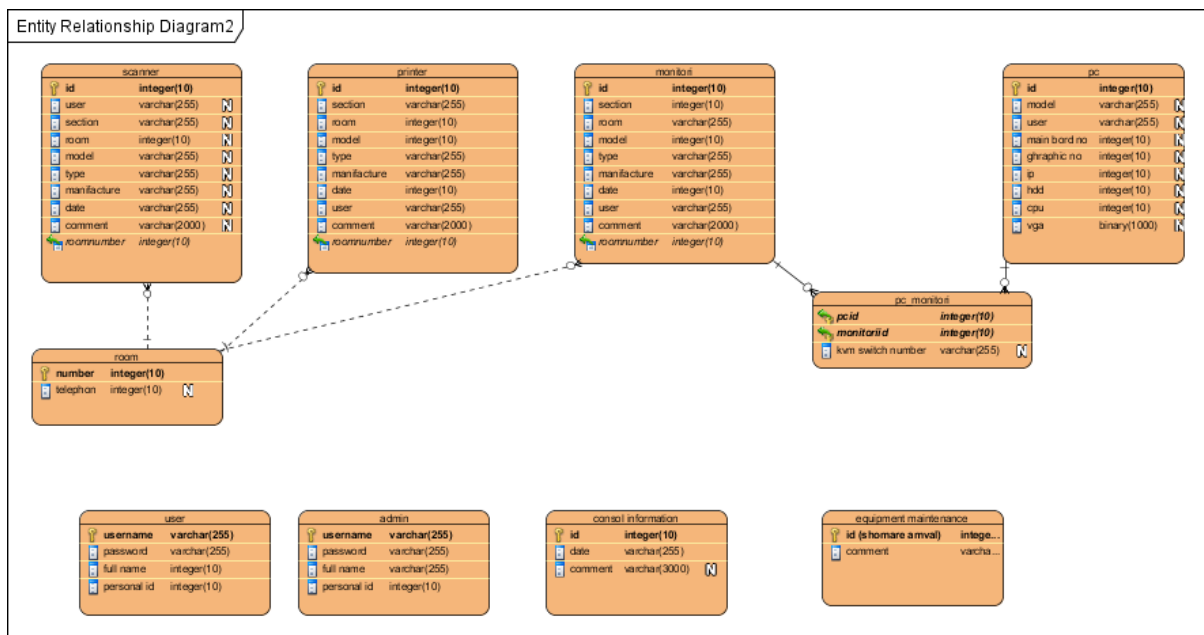
آینده خواهد بود و معمولاً در ادارات و مراکزی نصب می شود که دارای شعبه های متعددی هستند، ما از سیستم مدیریت پایگاه داده ی اوراکل استفاده نموده ایم . حتما این سوال برایمان پیش خواهد آمد که چگونه از این سیستم که یک سیستم مدیریت پایگاه داده است و برای استفاده از آن باید هزینه پرداخت شود استفاده می کنیم؟.

سیاست شرکت اوراکل بر این است که هرکسی می تواند از این سیستم مدیریت پایگاه داده استفاده کند و اگر هزینه ی آن را پرداخت نکند ، اگر در آینده برای دیتابیس مشکلی اعم از مشکلات امنیتی و مشکلات دیگری پیش بیاید ؛ این شرکت هیچگونه تعهدی در قبال مشکل به وجود آمده ندارد و مشکل را باید خودمان برطرف نماییم . در صورتی که هزینه استفاده از آن پرداخت شود وظیفه ی برطرف کردن مشکل بر عهده شرکت می باشد و شرکت اوراکل وظیفه دارد که فردی را برای برطرف ساختن مشکل به محل اعزام نماید . در ایران به دلیل آنکه قانون **Copy right** وجود ندارد بسیاری از مراکز دولتی و غیر دولتی از این سیستم به صورت رایگان استفاده می کنند.

دیگر علتی که ما از این سیستم مدیریت پایگاه داده برای پروژه خود انتخاب کردیم این است که این سیستم تعامل بسیار خوبی با زبان برنامه نویسی جاوا دارد .

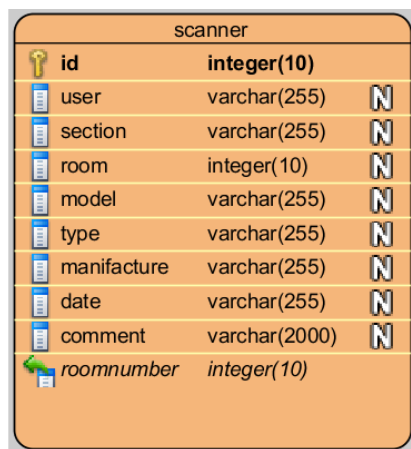
و علت آخری که ما از این سیستم استفاده نموده ایم این است که ، چون سیستمی که ما طراحی کردیم، رو به رشد و توسعه است و باتوجه به اینکه اوراکل مناسب برای پایگاه داده های عظیم است ؛ در آینده برای توسعه نرم افزار به مشکل بر نخواهیم خورد و می توانیم به راحتی پایگاه داده را توسعه دهیم . همچنین در آینده می خواهیم عکس هایی از هر یک از تجهیزات الکترونیکی را در پایگاه داده ذخیره نماییم و با توجه به اینکه اوراکل مناسب برای دیتابیس هایی است که دارای عکس و صوت می باشند ؛ لذا انتخاب این سیستم مدیریت پایگاه داده برای این پروژه بسیار مناسب می باشد

در ادامه این بخش به بررسی شکل پایگاه داده ی این پروژه خواهیم پرداخت .



شکل ۷-۳-۳ طراحی کلی مربوط به پایگاه داده سیستم

در شکل بالا [۷-۳-۳] طراحی کلی پایگاه داده را می بینیم که رابطه ی بین هر یک از جداول را نشان می دهد ، در ادامه این بخش هر یک از جداول را با شکل بررسی خواهیم نمود.













شکل ۸-۳-۳ جدول مربوط به اسکنر

همانطور که در این شکل [۸-۳-۳] مشاهده می کنید. جدول اسکنر که اطلاعات مربوط به اسکنر را در خود نگه می دارد دارای صفت هایی می باشد که اطلاعات در آنها ذخیره می شود. اولین صفتی که در این جدول می بینیم id می باشد که نوع آن از نوع عددی است و کلید اصلی این جدول می باشد و با استفاده از این کلید می توانیم اطلاعات را از جدول برداشت کنیم. همچنین roomnumber هم یک کلید













خارجی است که شماره اتاق را نشان می دهد. و این جدول دارای صفت های دیگری می باشد که می توانید در شکل مشاهده فرمایید.

printer	
 <b>id</b>	<b>integer(10)</b>
 section	varchar(255)
 room	integer(10)
 model	integer(10)
 type	varchar(255)
 manufacture	varchar(255)
 date	integer(10)
 user	varchar(255)
 comment	varchar(2000)
 roomnumber	integer(10)










شکل ۳-۳-۹ جدول مربوط به پرینتر

همانطور که در این شکل [ ۳-۳-۹ ] مشاهده می کنید. جدول پرینتر که اطلاعات مربوط به پرینتر را در خود نگه می دارد دارای صفت هایی می باشد که اطلاعات در آنها ذخیره می شود. اولین صفتی که در این جدول می بینیم id می باشد که نوع آن از نوع عددی است و کلید اصلی این جدول می باشد و با استفاده از این کلید می توانیم اطلاعات را از جدول برداشت کنیم. همچنین roomnumber هم یک کلید خارجی است شماره اتاق را نشان می دهد. و این جدول دارای صفت های دیگری می باشد که می توانید در شکل مشاهده فرمایید.

monitori	
 <b>id</b>	<b>integer(10)</b>
 section	integer(10)
 room	varchar(255)
 model	integer(10)
 type	varchar(255)
 manufacture	varchar(255)
 date	integer(10)
 user	varchar(255)
 comment	varchar(2000)
 roomnumber	integer(10)

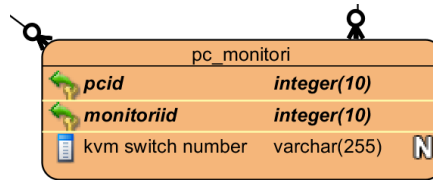
شکل ۳-۳-۱۰ جدول مربوط به مانیتور

همانطور که در این جدول شکل [ ۳-۳-۱۰ ] مشاهده می کنید. جدول مانیتور که اطلاعات مربوط به مانیتور ها را در خود نگه می دارد و دارای صفت هایی می باشد که اطلاعات در آنها ذخیره می شود. اولین صفتی که در این جدول می بینیم id می باشد که نوع آن از نوع عددی است و کلید اصلی این جدول می باشد و با استفاده از این کلید می توانیم اطلاعات را از جدول برداشت کنیم. همچنین roomnumber هم یک کلید خارجی است شماره اتاق را نشان می دهد. و این جدول دارای صفت های دیگری می باشد که می توانید در شکل مشاهده فرمایید.

pc		
	id	integer(10)
	model	varchar(255) N
	user	varchar(255) N
	main bord no	integer(10) N
	ghraphic no	integer(10) N
	ip	integer(10) N
	hdd	integer(10) N
	cpu	integer(10) N
	vga	binary(1000) N

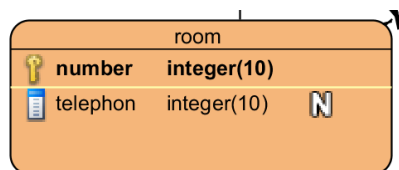
شکل ۳-۳-۱۱ جدول مربوط به کامپیوتر

همانطور که در این جدول شکل [ ۳-۳-۱۱ ] مشاهده می کنید. جدول کامپیوتر که اطلاعات مربوط به کامپیوتر ها را در خود نگه می دارد و دارای صفت هایی می باشد که اطلاعات در آنها ذخیره می شود. اولین صفتی که در این جدول می بینیم id می باشد که نوع آن از نوع عددی است و کلید اصلی این جدول می باشد و با استفاده از این کلید می توانیم اطلاعات را از جدول برداشت کنیم. بین جدول کامپیوتر و جدول مانیتور یک رابطه وجود دارد زیرا در یک اداره و مرکز یک کامپیوتر می تواند به چند مانیتور متصل شود و چند کامپیوتر نیز می تواند به یک مانیتور متصل شود لذا بین این دو جدول یک رابطه وجود دارد که در شکل زیر شکل [ ۳-۳-۱۲ ] خواهید دید.







شکل ۳-۳-۱۲ جدول مربوط به ارتباط میان جداول مانیتور و کامپیوتر

در این جدول شکل [ ۳-۳-۱۲ ] که شامل کلید های دو جدول دیگر است که این کلید ها کاید خارجی هستند و جمع این دو کلید کلید کاندید برای این جدول می باشد. این جدول نیز دارای صفت دیگری است با نام kvm switch با استفاده از کلید های موجود در این جدول می توانیم ببینیم که کدام کامپیوتر به کدام مانیتور یا مانیتور ها متصل است. در واقع این جدول بین ۲ جدول مانیتور و کامپیوتر قرار دارد که این نوع جدول را در پایگاه داده های رابطه ای داریم. از این رو به سیستم هایی که این جدول ها را پشتیبانی می کنند. سیستم های مدیریت پایگاه داده ی رابطه ای گفته می شود.







شکل ۳-۳-۱۳ جدول مربوط به اتاق ها

همانطور که در این شکل [ ۳-۳-۱۳ ] مشاهده می کنید. جدول اتاق که اطلاعات مربوط به اتاق ها را در خود نگه می دارد و دارای صفت هایی می باشد که اطلاعات در آنها ذخره می شود. در این جدول شماره اتاق، کلید اصلی جدول برای پیدا کردن اطلاعات می باشد. و این جدول با جدول های مانیتور ، اسکر و پرینتر رابطه n:۱ دارد . و با استفاده از کلید roomnumber که در جدول های ذکر شده کلید خارجی است می توانیم ببینیم که هر کدام از تجهیزات در کدام اتاق قرار دارند.

user	
 <b>username</b>	<b>varchar(255)</b>
 password	varchar(255)
 full name	integer(10)
 personal id	integer(10)





شکل ۳-۳-۱۴ جدول مربوط به حساب های کاربری

در این جدول شکل [ ۳-۳-۱۴ ] که اطلاعات مربوط به کاربران در آن نگهداری می شود ، **username** کلید اصلی است که می توانیم اطلاعات را توسط آن از جدول برداشت کنیم . این جدول هنگامی استفاده می شود که کاربران می خواهند وارد سیستم شوند. و نرم افزار سرور با بررسی صحت اطلاعات اجازه ورود به سیستم را به کاربر می دهد.

admin	
 <b>username</b>	<b>varchar(255)</b>
 password	varchar(255)
 full name	varchar(255)
 personal id	integer(10)

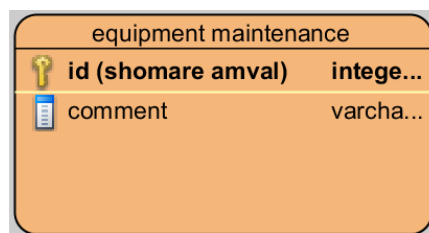
شکل ۳-۳-۱۵ جدول مربوط به حساب های مدیران

در این جدول شکل [ ۳-۳-۱۵ ] که اطلاعات مربوط به مدیران در آن نگهداری می شود ، **username** کلید اصلی است که می توانیم اطلاعات را توسط آن از جدول برداشت کنیم. این جدول هنگام ورود مدیران به سیستم استفاده می شود و اطلاعات وارد شده با اطلاعات موجود در این جدول بررسی می شود و در صورت صحیح بودن اجازه ورود به سیستم صادر می شود.

consol information	
 <b>id</b>	<b>integer(10)</b>
 date	varchar(255)
 comment	varchar(3000) 

شکل ۳-۳-۱۶ جدول مربوط log های سیستم

در این جدول شکل [ ۳-۳-۱۶ ] تمامی تراکنش هایی که توسط کاربران و مدیران انجام می شود نگهداری می شود و در مواقع نیاز می توانی بر اساس **id** تراکنش های ثبت شده در سیستم را بازبینی نمود.



شکل ۳-۳-۱۶ جدول مربوط تعمیرات انجام شده بر روی تجهیزات

در این جدول شکل [ ۳-۳-۱۶ ] که اطلاعات تعمیرات مربوط به هر یک از تجهیزات در آن نگهداری می شود ؛ با استفاده از ID که در این جدول به عنوان کلید اصلی است ، می توانیم تعمیرات صورت گرفته بر روی هر یک از تجهیزات را با استفاده از شماره ی هر یک از تجهیزات بازیابی کنیم .

### ۴-۳ معماری و فناوری های مورد استفاده HIBERNATE

در این بخش به معماری استفاده شده و فناوری های استفاده شده در این نرم افزار خواهیم پرداخت و بررسی می کنیم که در این نرم افزار از چه روشی استفاده شده است و آن روش ها را شرح خواهیم داد [2][1].

ابتدا به معماری مورد استفاده در این نرم افزار خواهیم پرداخت .

در تولید این نرم افزار از معماری mvc استفاده شده است که دارای سه بخش می باشد [7] :

**Model** : در این بخش کد های مربوط به دیتابیس قرار می گیرد و این لایه از این معماری وظیفه ی

دسترسی به داده های دیتابیس دارد. که خود دارای سه لایه دیگر است (TO /DA/BL). برای اطلاعات

بیشتر به پیوست مراجعه فرمایید .

**View** : بخشی است که در واقع رابط کاربر است و اطلاعات را به کاربر نشان می دهد .

در این پروژه تمامی کلاس هایی که وظیفه ی واسط گرافیکی را دارند را همگی در این پوشه قرار داده ایم و کار های مربوط به Listener ها را نیز همگی در این فایل قرار می دهیم . که ما در این پروژه ۳ کلاس داریم که هر یک وظیفه ی نمایش یک فرم را به صورت گرافیکی بر عهده دارند. برای طراحی واسط گرافیکی در جاوا روش های مختلفی وجود دارد مانند : swing, javafx و ... که ما در این پروژه از swing برای طراحی واسط گرافیکی خود استفاده کرده ایم و تمامی بخش های گرافیکی را با استفاده از کد از پایه طراحی کرده ایم.

**Controller** : بخشی است که رابط بین دو بخش دیگر است . در این پروژه این بخش است که برنامه را اجرا می کند و در واقع این بخش است که تابع main در آن قرار دارد و اولین فرم از برنامه را اجرا می کند

و اولین فرم از برنامه ما فرم ورود به سیستم است. همچنین این بخش اطلاعات را با استفاده از BL از دیتابیس می خواند و آن را به بخش view می فرستد تا به کاربر نمایش داده شود. این بخش وظیفه ای اساسی بر عهده دارد.

**Model** : بخشی است که خود نیز دارای سه بخش دیگر است و با دیتابیس در ارتباط است.

TO (Table object) : کلاس های ما در این قسمت قرار گرفته می شوند.

DA (Data access) : این لایه وظیفه ی ارتباط با دیتابیس را بر عهده داد.

BL (business logic) : این لایه وظیفه برگرداندن داده ها به قسمت controller دارد.

با استفاده از معماری MVC بخش های مختلف برنامه از هم جدا می شوند و این باعث می شود که عیب یابی به سرعت صورت بگیرد و امکان توسعه برنامه نیز وجود داشته باشد و توسعه به راحتی صورت گیرد.

لازم به ذکر است که در این پروژه از معماری mvc توزیع شده نیز استفاده شده است. کاری که ما در این پروژه با استفاده از معماری mvc انجام دادیم این است ؛ در قسمت TO کلاس های پروژه را قراردادیم که این کلاس ها دقیقاً در پایگاه داده به جدول تبدیل می شود و ویژگی های هر کلاس ، صفت های جدول ها می شود. در فایل دیگر که به نام DA است با استفاده از درایور jdbc به دیتابیس متصل می شویم و درخواست های خود را به دیتابیس می نویسیم و پاسخ را از دیتابیس دریافت می کنیم. همچنین در این فایل دو تابع مجزا وجود دارد که یکی ارتباط با دیتابیس را برقرار می کند و دیگری ارتباط با دیتابیس را قطع می کند. در فایل bl تابع ارتباط با پایگاه داده را فراخوانی می کند و ارتباط برقرار می شود ، سپس تابع مورد نظر را فراخوانی کرده و اجرا می شود و در آخر نیز با استفاده از تابع دیگر ، ارتباط با دیتابیس را قطع می کند. درواقع در این معماری BL ارتباط با بخش کنترلر دارد و دیگر بخش ها مستقیماً با بخش کنترلر درگیر نمی شوند.

## (Remote method invocation ) RMI

یکی از فناوری های مورد استفاده در این پروژه برای برقراری ارتباط میان بخش سرور و بخش سرویس گیرنده استفاده از روشی است که به آن (Rmi) گفته می شود [1] و چون توسط خود جاوا استفاده می شود دارای سرعتی بالا می باشد . برای ارتباط میان اجزا در یک پروژه از web service نیز می توان بهره برد که چون میان تمام زبان ها مشترک می باشد از سرعت پایین تری برخوردار است . در این پروژه ما از روش اول یعنی rmi استفاده نموده ایم که باعث افزایش سرعت می شود . مطلب دیگری که باید مورد بررسی قرار گیرد این است که هنگامی که می خواهیم یک کلاس را به عنوان یک داده تحت شبکه با استفاده از این روش انتقال دهیم حتما باید به کلاس با استفاده از interface یک برچسب serializable بدهیم تا بتوانیم آن شی از کلاس را، تحت شبکه انتقال بدهیم برای مطالعه بیشتر به پیوست مراجعه فرمایید.

در این پروژه یک سری interface تعریف کردیم که ریموت های برنامه به شمار می روند و داخل هر یک توابع مربوط به اضافه کردن و دیگر عملیات های کار بر را قرار دادیم و با استفاده از کلاس های دیگر این توابع را متناسب با نوع نیاز پیاده سازی کردیم . هنگامی که این شی را در بستر شبکه به اشتراک می گذاریم ، کاربر با فراخوانی تابع مورد نظر درخواست خود را به سمت سرور ارسال می کند . در پیاده سازی این کلاس ها در برنامه سرور ؛ داخل بدنه این توابع کلاس های موجود در bl را شی می سازیم و توابع مورد نظر را برای ارتباط با دیتابیس فراخوانی می کنیم . بدین صورت برنامه سمت کاربر با برنامه سمت سرور ارتباط برقرار می کند و به تعامل به یکدیگر می پردازند.

## (Java Database connectivity) JDBC

یک فناوری در جاوا می باشد که با استفاده از آن می توان با دیتابیس ارتباط بر قرار کرد [1] . در این پروژ برای ارتباط با دیتابیس oracle از این فناوری استفاده شده است . با دانلود کردن درایور مخصوص به هر



نرم افزار مدیریت دیتابیس و استفاده ی آن داخل پروژه می توان با زبان جاوا با دیتابیس مورد نظر ارتباط بر قرار کرد . کتابخانه jdbc خود دارای ۳ فایل است که باید آن را به پروژه اضافه نماییم و با استفاده از دستوراتی در زبان جاوا آن ها را فراخوانی کنیم . دستوراتی مانند :

```
Class.forName("oracle.jdbc.driver.OracleDriver");  
connection =  
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl","managment","1374");
```

برای ارتباط با دیتابیس و وارد کردن اطلاعات کاربری برای اتصال به دیتابیس.

برای مطالعه ی بیشتر به پیوست مراجعه فرمایید.

### ۵-۳ محیط برنامه نویسی مورد استفاده و ویژگی های آن

برای برنامه نویسی به زبان جاوا محیط (IDE) های مختلفی وجود داد. پایه ای ترین محیط برای برنامه نویسی به زبان جاوا محیط Notepad می باشد؛ ولی این محیط به دلیل نداشتن امکانات، امروزه زیاد مورد استفاده قرار نمی گیرد . محیط برنامه نویسی خوب به دلیل داشتن یک سری امکانات خوب مانند نشان دادن خطا ها و ... در فرایند تولید یک نرم افزار بسیار کمک می کند و باعث سرعت در تولید برنامه و خطایابی برنامه می شود همچنین روند تولید نرم افزار را سرعت می بخشد.



ما در این پروژه از محیط برنامه نویسی jet brain intellij استفاده کرده ایم که محیطی می باشد که تولید نرم افزار های در آن آسان می باشد و امکانات بهتری را در اختیار برنامه نویس قرار می دهد..

محیط های دیگری نیز وجود دارد که به معرفی مختصری از 4 نمونه دیگر خواهیم پرداخت.





### ۳-۶ الگوها اصول مهندسی نرم افزاری ضروری برای انجام کار

در این بخش از فصل سوم به بررسی الگوهای شی گرای ( Design pattern ) استفاده شده در این پروژه خواهیم پرداخت [2] و همچنین اصول مهندسی نرم افزار استفاده شده در فرایند تولید این پروژه را نیز به طور مختصر بیان خواهیم نمود برای بررسی بیشتر به قسمت پیوست مراجعه فرمایید.

برای درک بهتر این الگو ها باید با برنامه نویسی شی گرای نیز آشنایی داشته باشید لذا پیشنهاد می شود اصول برنامه نویسی شی گرای را نیز مطالعه فرمایید.[6]

اولین الگویی که بررسی می کنیم الگوی singleton می باشد.

بررسی الگوی singleton :

در برنامه نویسی شی گرای ما دو نوع کلاس داریم کلاس های حالت دار (state full) و کلاس های بی حالت (state less) ؛ کلاس های حالت دار به کلاس هایی گفته می شود که دارای متغیر باشند ، علت اینکه می گوییم این نوع کلاس ها حالت دار است این است که شی های تولید شده از این کلاس با یکدیگر متفاوت هستند لذا به این نوع کلاس ها حالت دار گفته می شود.

کلاس های بی حالت به کلاس های گفته می شود که دارای متغیر نیستند و اگر هم متغیر دارند متغیر های آنها از نوع static ، constant و یا final است . این کلاس ها را بی حالت می گوییم چون تمامی اشیا تولید شده از این کلاسی ها با هم یکسان هستند لذا الگوی فوق را برای این گونه کلاس ها استفاده می کنیم .

گاهی برای ما در برنامه نویسی پیش می آید که می خواهیم تنها یک شی از یک کلاس داشته باشیم و برنامه نویس نتواند که بیشتر از یک شی از کلاس بسازد لذا در این مواقع از این الگو استفاده می کنیم . این الگو همچنین در ارتباط برنامه با دیتابیس استفاده نیز می شود . این الگو یکی از پر استفاده ترین الگو ها است و به راحتی نیز پیاده سازی می شود. لازم به ذکر است که تعداد ۲۳ تا از این الگو ها نیز در دنیای شی

گرایی وجود دارد که روند تولید برنامه را سرعت می بخشد و مشکلات شی گرایی را نیز برطرف می کنند. برای مشاهده پیاده سازی های الگو های مطرح شده به قسمت پیوست مراجعه فرمایید.

در این پروژه تعدادی از کلاس ها که بی حالت هستند را از الگوی singleton در تعریف کلاس های آن ها استفاده کرده ایم . به طور مثال در کلاسی که سرویس rmi را فعال و غیر فعال می کند , کلاس مربوط به این کار را singleton کرده ایم چون این کلاس فقط شامل توابع است.

دیگر الگویی که بررسی می کنیم و در این پروژه از آن استفاده شده است الگوی factory می باشد.

#### بررسی الگوی factory :

این الگو پیچیدگی های ساخت یک شی را از کاربر برنامه نویس پنهان می کند . این الگو با گرفتن یک نوع رشته توسط کاربر یک شی را متناسب با آن رشته ساخته و به کاربر برنامه نویس می دهد . این الگو باعث می شود که برنامه نویس برای ساخت اشیا دچار اشکال و سردرگمی نشود و فقط با داشتن یک نوع رشته می توان شی را ساخت. در بعضی پروژه های بزرگ چون اشیا و کلاس متعدد هستند ؛ کاربر برنامه نویس ممکن است که همه کلاس ها را نداند و از ساختار همه کلاس ها اطلاع نداشته باشد ؛ چون ممکن است هر نوع کلاس به روش های مختلف شی ساخته شود لذا با استفاده از این الگو پیچیدگی ساخت اشیا از کلاس ها را از دید کاربر برنامه نویس مخفی می کنیم و این باعث می شود که سرعت تولید برنامه بالا برود و روند عیب یابی نیز سرعت بخشیده شود . در این ساختار یک کلاس داریم که یک متد دارد و این متد وظیفه این را دارد که متناسب با نوع رشته ورودی شی مورد نظر را ساخته و بر می گرداند و این روش در ساخت اشیا بسیار موثر می باشد. در این پروژه یک کلاس تعریف کرده ایم با نام factory که داخل آن یک تابع وجود دارد با نام makeobj که این تابع یک نوع ورودی از نوع رشته را دارد که متناسب با نوع رشته شی مورد نظر را ساخته و به کاربر برنامه نویس برمی گرداند. هر جا از برنامه که می خواهیم شی تولید کنیم، ابتدا تابع makeobj را فراخوانی کرده و نوع رشته ورودی را متناسب با شی که می خواهیم تولید شود را به آن می دهیم . بدین ترتیب شی ما ساخته می شود و به کاربر برنامه نویس برگردانده می شود.

```

public static device mackobj(String s )
{
    if (s.equals("computer"))
    {
        return new Computer();
    }
    else if (s.equals("consol"))
    {
        return new consol();
    }
    else if (s.equals("equipment"))
    {
        return new equipmentMainteneace();
    }
    else if (s.equals("monitor"))
    {
        return new Monitor();
    }
    else if (s.equals("network"))
    {
        return new NetworkUtility();
    }
    else if (s.equals("printer"))
    {
        return new Printer();
    }
    else if (s.equals("room"))
    {
        return new room();
    }
    else if (s.equals("scaner"))
    {
        return new Scaner();
    }
    else
    {
        return null;
    }
}

```

شکل ۳-۶-۱ تابع ساخت شی

برای مطالعه بیشتر به قسمت پیوست مراجعه فرمایید.

الگوی دیگری که مورد بررسی قرار می دهیم الگوی builder می باشد.

بررسی الگوی builder :

این الگو که سازنده نام دارد روند اجرای چند متد از یک کلاس را به ترتیب اجرا می کند. بدین منظور ما یک کلاس تعریف می کنیم که دارای یک متد است که یک شی ورودی را دریافت می کند و با توجه به ترتیب خواسته شده ؛ توابع داخلی هر شی را فراخوانی می کند لذا به این الگو سازنده گفته می شود. برای مطالعه بیشتر به پیوست مراجعه فرمایید.



### الگوها اصول مهندسی نرم افزاری ضروری برای انجام کار

در مهندسی نرم افزار مدل های مختلفی برای اجرا یک پروژه تدوین شده است که هر کدام ویژگی های منحصر به فرد خود را دارد لذا در ابتدا باید فاز های آن را توضیح دهیم [7].

**Communication :** در این بخش افراد با مشتری ، که پروژه را سفارش داده است صحبت می کنند؛

و نیاز های مشتری را مشخص می کنند که این عمل با صحبت کردن حضوری صورت می گیرد.

**Planning :** در این بخش که خود دارای سه بخش دیگر است . برنامه ریزی مربوط به پروژه صورت می گیرد همچنین تخمین زده می شود که منابع مورد نیاز اولیه برای شروع پروژه چقدر است ، همچنین در بخش آخر که بخش پیگیری است فردی را مسئول می کنند که کار ها را پیگیری کند.

**Modeling :** در این بخش که دارای ۲ فاز analysis و design می باشد ، طراحی های مربوطه انجام می شود . در بخش آنالیز کلاس های مورد نیاز مشخص می شود و در بخش طراحی رابطه بین هر کلاس مشخص می شود .

**Construction :** این بخش خود نیز دارای ۲ بخش دیگر است : ۱- code - ۲- test

در بخش کد ، شروع به پیاده سازی پروژه می کنیم و در بخش تست نیز به تست های نرم افزار تولید شده می پردازیم .

**Development :** این فاز نیز خود دارای ۳ قسمت می باشد :

**Deployment :** در این بخش نرم افزار را در جای مورد نظر نصب می کنیم .

**Operation** : در این بخش به نرم افزار ، داده ی حقیقی می دهیم .

**Maintenance** : در این بخش پشتیبانی و برطرف کردن عیب های نرم افزار صورت می گیرد.

مدل های فرایند تولید نرم افزار متشکل از موارد فوق است که به دلیل متعدد بودن آنها از توضیح آنها صرفه نظر می کنیم . در این فرایند تولید این پروژه ما از اصول Agile استفاده کرده ایم [7] . این اصول دارای تعدادی روش است . در این روش ها در واقع مدیریت تغییرات و ریسک صورت می گیرد و این روش ها به ما می آموزد که چگونه در مقابل تغییرات به وجود آمده در مراحل مختلف بهترین راه حل ممکن برای پاسخگویی به آن نیاز را انتخاب کنیم . در فرایند تولید این نرم افزار از این روش استفاده شده است.

با استفاده از این اصول که در تولید نرم افزار استفاده شده است در هر زمان می توانیم به نیاز های مشتری و یا نیاز های به وجود آمده پاسخ بدهیم و پاسخگویی به این نیاز ها مستلزم آن است که طراحی خوبی داشته باشیم و با توجه به موارد مطرح شده در قسمت های قبل این پروژه دارای طراحی خوب است و این باعث می شود که توسعه و تغییر آن به صحوالت انجام گیرد.

## فصل چهارم

تحلیل و ارزیابی سامانه مدیریت سیستمهای الکترونیکی

۱-۴ تحلیل و ارزیابی کل سیستم

۲-۴ نمودارهای UML مربوط به طراحی سیستم

۳-۴ تحلیل مربوط به دیتابیس





#### ۱-۴ تحلیل و ارزیابی کل سیستم

در این بخش از فصل چهارم به بررسی این سیستم خواهیم پرداخت و ارزیابی این سیستم را انجام خواهیم داد و در ادامه فصل چهارم به تحلیل کلی سیستم خواهیم پرداخت و نمودارهای مربوط به طراحی پایگاه داده و نمودارهای کل سیستم را نیز بررسی خواهیم نمود.

در این بخش از سیستم به بررسی و تست این نرم افزار خواهیم پرداخت و نمودار تست این نرم افزار را نشان خواهیم داد.

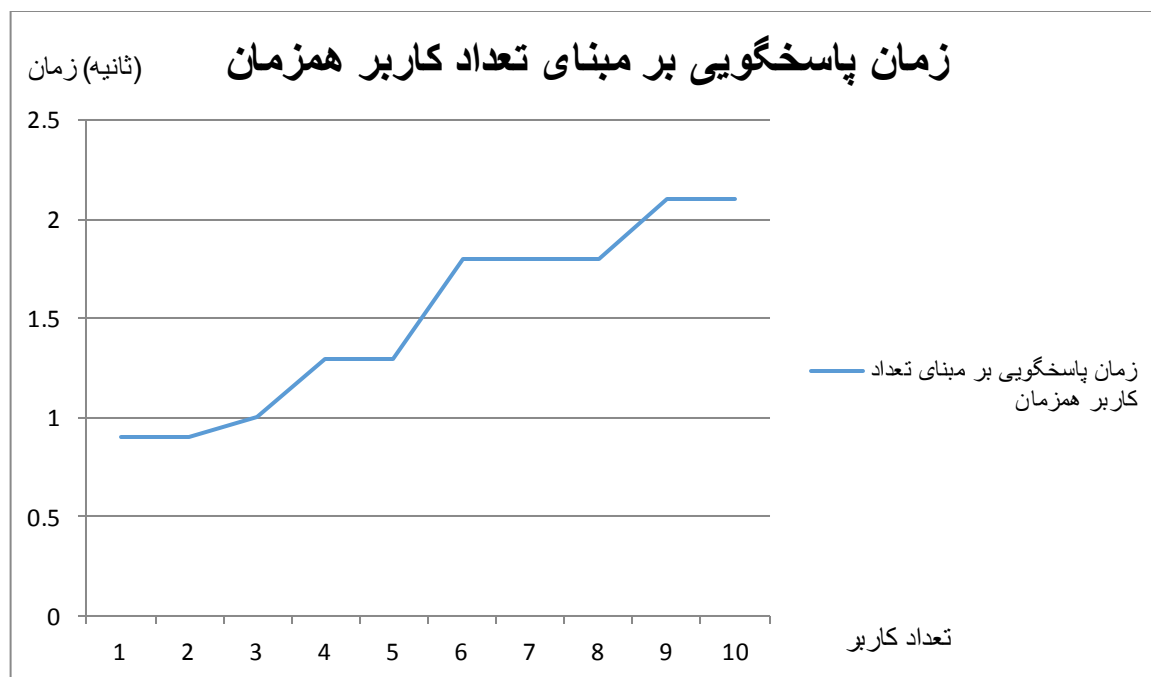
این نمودار ۱-۴-۱ زمان اجرای درخواست ها را با توجه به تعداد کاربران متصل به سیستم نشان میدهد . ما برای انجام این آزمایش ابتدا با استفاده از یک access point ، ۳ عدد کامپیوتر را با یکدیگر شبکه کردیم . این شبکه شامل ۲ لپ تاپ که بر روی یکی از این لپ تاپ ها نرم افزار سرور قرار می گیرد و یک کامپیوتر کوچک با نام raspberry pi است که همه این کامپیوتر ها از سیستم عامل ویندوز استفاده می کنند .

برای آنکه بتوانیم ۱۰ عدد کامپیوتر را بر روی ۳ عدد کامپیوتر شبه سازی کنیم بر روی کامپیوتر سرور علاوه بر نرم افزار سرور ۳ نرم افزار کلاینت باز می کنیم و بر روی ۲ کامپیوتر باقی مانده هم به همین روش تعداد ۳ عدد و ۴ عدد برنامه را باز می کنیم ؛ با استفاده از این روش می توانیم یک محیط شامل ۱۰ کاربر را شبیه سازی نماییم . البته برای برقراری ارتباط با برنامه سرور لازم است که ip کامپیوتر سرور را داشته باشیم تا بتوانی با آن ارتباط برقرار کنیم. برای انجام این آزمایش هم حداقل به ۲ نفر نیاز است که ما این آزمایش را انجام دادیم و مدت زمان را با استفاده از ثانیه شمار اندازه گرفتیم . البته این تست یک تست ابتدایی می باشد و تست اصلی باید در یک محیط واقعی صورت بگیرد و به مدت طولانی نیز انجام بگیرد. تستی که ما از

این سیستم گرفتیم در یک محیط آزمایشگاهی است و کامپیوتر ها را نیز شبیه سازی کرده ایم . حال به بررسی زمان ، سرعت و کارایی سیستم با توجه به این نمودار خواهیم پرداخت.

البته شایان ذکر است که چون با استفاده از یک ثانیه شمار و یه صورت دستی این آزمایش را انجام داده ایم مقداری خطا در آزمایش وجود دارد ولی سعی شده است که بهترین دقت را با استفاده از امکانات موجود در این آزمایش به دست آوریم .

همانطور که در این نمودار می بینید مدت زمان پاسخ گویی در تعداد ۳ کامپیوتر ثابت است و با افزایش تعداد کامپیوتر ها زمان آن افزایش پیدا می کند البته میزان افزایش زمان زیاد نیست . البته این تست بر روی یک سرور معمولی صورت گرفته است ؛ بدیهی است که هرچه سرور قوی تر باشد سرعت بیشتر می شود. و زمان پاسخگویی نیز کمتر می شود. البته کارایی این سیستم با ۱۰ کاربر خوب است و می توان آن را برای مراحل بعدی تست در یک مرکز نصب و راه اندازی نمود. همچنین دقت کار در این سیستم خوب است زیرا ما با استفاده از راه کار های موجود جلوی `sql injection` را گرفته ایم ؛ همچنین با استفاده از کد از وارد کردن اطلاعات اشتباه نیز جلوگیری کرده ایم که این خود دقت کار را بالا می برد و در زمان اجرا نیز صرفه جویی می کند. همچنین کاربر نمی تواند فیلد ها را خالی بگذارد و باید تمامی آن ها را تکمیل نماید. البته در مراحل بعدی تست با استفاده از `data mining` و با استفاده از نرم افزار `weka` می توانیم داد کاوی را بر روی داده ها انجام دهیم و یکی از علل هایی که باعث کاهش دقت در داده کاوی می شود وجود `missing values` است که ما با استفاده از طراحی خود جلوی این کار را گرفته ایم. علت دیگری که باعث کاهش دقت می شود وجود دادهایی است که مناسب نیستند ؛ مانند اینکه در این پروژه شماه اتاق منفی باشد که معنا ندارد . در این پروژه ما به کاربر اجازه نمی دهیم که داده های اشتباه را وارد سیستم کند .



۴-۱-۱ نمودار مربوط به تست نرم افزار با استفاده از ۱۰ کاربر

اعداد عمودی ثانیه را نشان می دهند و اعداد افقی تعداد کاربران را نشان می دهد.

این سیستم که تولید شده است دارای دو بخش است : بخش سرور و بخش سرویس گیرنده که بخش سرور به دیتابیس ما متصل می باشد و درخواست های کاربران سرویس گیرنده را پاسخ می دهد همچنین در سرور امکان این وجود دارد که بتوانیم فرایند سرویس دهی را متوقف کنیم.

تحلیل دیگری که می خواهیم انجام دهیم تحلیل مربوط به تعداد سرویس گیرنده هایی که می توانند به سرور متصل شوند . از لحاظ نرم افزاری که بر روی سخت افزار سرور اجرا می شود هیچ محدودیتی وجود ندارد . چون خود نرم افزار سرور به ازای هر یک اتصال و درخواست یک نخ اجرایی ( threads ) ایجاد می کند و تمامی درخواست ها می توانند موازی با هم اجرا شوند . تعداد سرویس گیرنده ها رابطه مستقیم با سخت افزار سرور دارد لذا تعداد سرویس گیرنده ها نیز به قابلیت و توان سرور بستگی دارد لذا هرچه سرور از لحاظ سخت افزاری قوی تر باشد تعداد سرویس گیرنده های بیشتری می توانند از سرور سرویس بگیرند .

همچنین بخشی از کارایی دیتابیس به سخت افزار سرور بستگی دارد . لذا معمولا بسته به نوع پروژه ه از کامپیوتر های قوی به عنوان سرور استفاده می شود. همچنین در سرور ها باید از سخت افزاری استفاده شود که قابلیت این را داشته باشد که بتواند به صورت ۲۴ ساعته روشن باشد و سرویس بدهد . همچنین سرور ها باید این قابلیت را داشته باشند که بعضی از قطعات را بدون خاموش کردن سرور تعویض نمود که به این قابلیت hot plug گفته می شود ؛ معمولا سرور های معمولی این قابلیت را دارند که بتوانند قسمت حافظه را بدون خاموش کردن سرور این قطعه را عوض نمود. البته سرور هایی نیز وجود دارد که این قابلیت را دارد دیگر قطعات را بتوان در حالت روشن تعویض نمود. اگر این سیستم را بر روی یک کامپیوتر که عادی باشد قرار دهیم می توانیم به ۴۰ تا ۵۰ تا کلاینت سرویس بدهیم ،البته این سیستم چون معمولا برای مدیران و مسئولین it در سازمان ها طراحی شده است لذا در بزرگترین مرکز حداکثر ۵۰ کامپیوتر کافی می باشد .



شکل ۴-۲-۱ اتاق های سرور در دیتا سنتر ها

همچنین سرعت اینترنت و شبکه بر روی کارایی سیستم نیز تاثیر می گذارد ، چون این سیستم یک سیستم توزیع شده می باشد و معماری آن سرویس دهنده و سرویس گیرنده می باشد ، لذا سرعت شبکه در کارایی سیستم بسیار تاثیر دارد. در مراکزی که شبکه با استفاده کابل ADSL می باشد که سرعت دانلود بالا می باشد و سرعت آپلود پایین می باشد و برای ارتباط بین بخش های نرم افزار باید سرعت آپلود بالا باشد لذا سرعت شبکه نیز بر افزایش و یا کاهش کارایی نیز موثر می باشد . البته لازم به ذکر است که نوع طراحی نیز در بهبود عملکرد سیستم نیز تاثیر دارد که در بخش های دیگر این فصل به طور مفصل به آن خواهیم پرداخت.

در این بخش از فصل چهارم به طراحی شماتیک پروژه خواهیم پرداخت و تعدادی از شکل ها را خواهیم کشید و مابقی را می توانید در بخش پیوست مشاهده فرمایید .

در ابتدا به تعریف Uml می پردازیم و در ادامه نمونه ای از هر یک از مدل ها را نشان خواهیم داد و مابقی را می توانید در بخش پیوست مشاهده فرمایید [7] .

### UML (Unified Modeling Language)

یک زبان است برای توصیف کردن و مدل کردن نرم افزار در زمینه ی مهندسی نرم افزار و با استفاده از این زبان می توانیم طراحی خود را به صورت شماتیک نشان دهیم . این زبان دارای تعدادی نمودار ها است که تعدادی از مهمترین و پر کاربرد ترین آن ها را بیان خواهیم نمود .

**Actor** : شی ویا چیزی است که می تواند انسان ، نرم افزار و سخت افزار باشد و سیستم برای این به وجود آمده است این اکتور ها به هدف خود برسند .

اکتور ها ۲ نوع هستند : ۱- اصلی ۲- فرعی

اصلی : سیستم برای این به وجود آمده است که این اکتور ها به هدف خود برسند.

فرعی : اکتوری است که در رسیدن اکتور اصلی به هدفش کمک می کند .

**Use case Diagram** : این نمودار رفتار کلی هر سیستم را نشان می دهد و کاری که این سیستم انجام می دهد و وظایف هر یک از اکتور ها را نشان می دهد. در این نمودار روابطی مانند include و extends داریم که به طور اختصار به هر یک خواهیم پرداخت .

**Include** : به این معنا می باشد که این use case شامل case use دیگری می باشد

به بیان ساده تر برای اجرا شدن این فعالیت باید ابتدا فعالیت دیگری اجرا شود و بعد این فعالیت اجرا شود.

**Extends** : این رابطه بیانگر آن است که این use case جزئی از use case دیگر است

و بخشی از یک فعالیت جامع تر است.

**Use case text** : رفتار کلی سیستم را به صورت متنی به ما نشان می دهد.

**Activity Diagram** : برای هر کدام از use case ها باید یک نمودار فعالیت رسم نماییم که این

نمودار به ما نشان می دهد که این فعالیت چگونه شروع و چگونه به پایان می رسد. و روند اجرای این فعالیت را حالت به حالت به ما نشان می دهد.

**Swim lane Diagram** : این نمودار همانند Activity diagram می باشد با این تفاوت که در

آن نقش هر یک از اکتور ها مشخص می شود.

## **(Data Flow Diagram ) DFD**

این نمودار که دارای سطوح مختلف می باشد ارتباط و تعامل سیستم با محیط اطراف خود نشان می دهد .

## **Class Diagram**

این نمودار کلاس های سیستم ما را نشان می دهد و روابط بین هر یک از کلاس ها را نیز نشان می دهد.

در این نمودار مفاهیمی مانند composition و aggregation داریم که هر یک را مختصر توضیح خواهیم داد .

**Composition** : به معنای ترکیب است یعنی یک موجودیت ترکیب موجودیت دیگر است

برای مثال : رابطه بین فصل های کتاب و کتاب رابطه ی ترکیب است . در این رابطه اگر جزء از بین برود کل نیز از بین می رود در این جا با از بین رفتن فصل های کتاب ، موجودیت کتاب دیگر معنا ندارد.

**Aggregation** : به معنای تجمع است و بیان گر این است که یک موجودیت جزئی از

موجودیت دیگر است . برای مثال چرخ و ماشین رابطه ی تجمع دارند به این معنا که چرخ بخشی از ماشین است . با از بین رفتن جزء موجودیت کل از بین نمی ورد. در این مثال با از بین رفتن چرخ موجودیت ماشین از بین نمی رود و به اینگونه از روابط aggregation گفته می شود.

**Sequence Diagram** : این نمودار رابطه بین هر یک از کلاس ها ی سیستم را به صورت نحوه

فراخوانی بین هر کلاس ها را نشان می دهد و نشان می دهد که بین هر کلای کدام یک از توابع فراخوانی می شود. علاوه بر این نمودار ها نمودار های دیگری نیز وجود دارد مانند : component Diagram , collaboration Diagram و ... که برای اطلاع بیشتر می توانید به کتب مربوط به مهندسی نرم افزار مراجعه فرمایید [7].

ابزار های موجود برای ترسیم نمودار های Uml :

برای ترسیم این نمودار ها به صورت دیجیتالی در کامپیوتر نرم افزار های متعددی وجود دارد مانند :

Microsoft visio  , visual paradigm  و ... که ما از نرم افزار

visual paradigm برای رسم این نمودار ها استفاده نموده ایم.

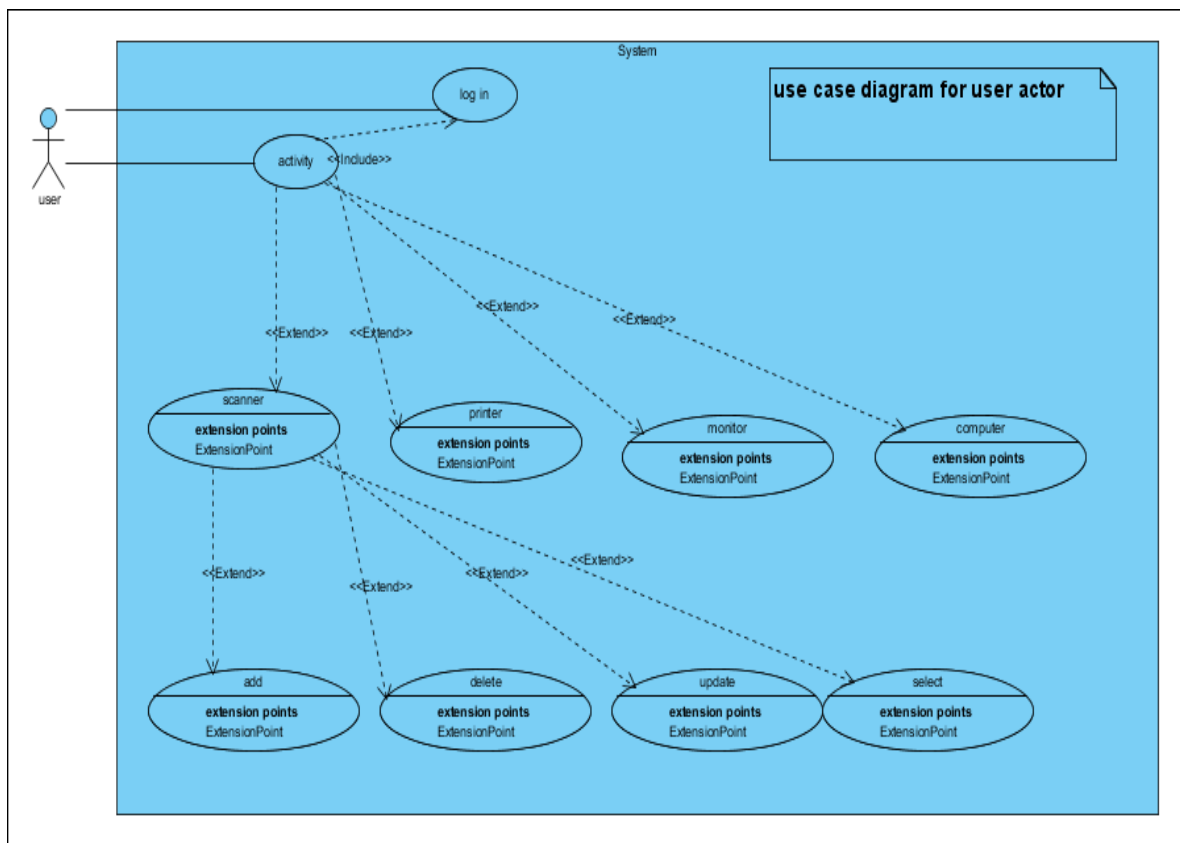
در ادامه یک نمونه از هر یک از این نمودار ها را نشان خواهیم داد و مابقی را می توانید به پیوست مراجعه

فرمایید.



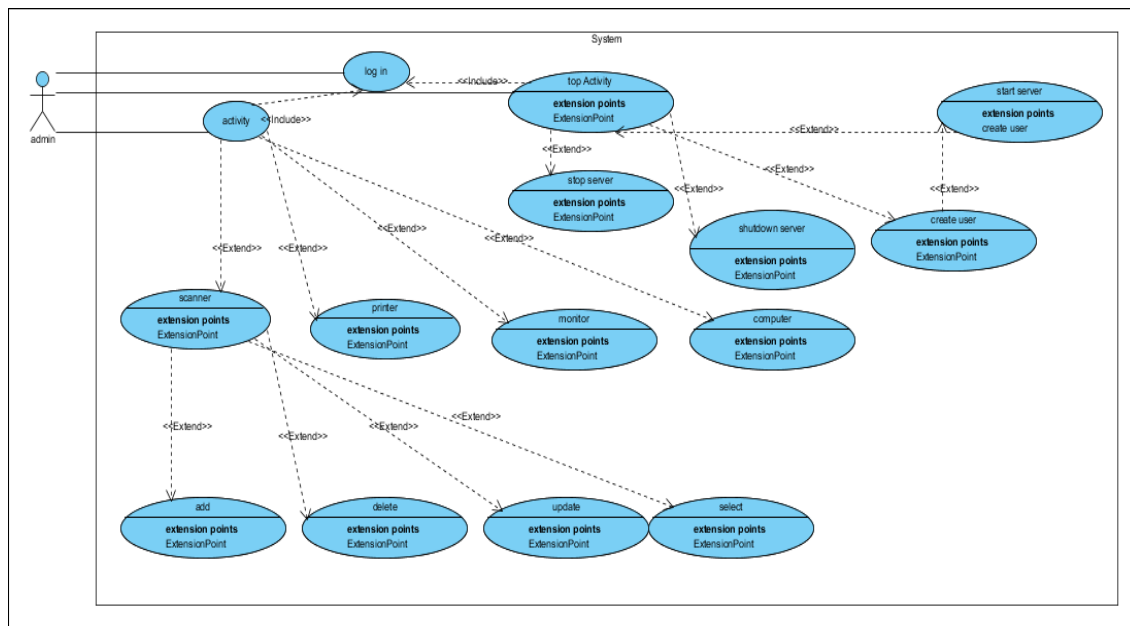
## Use case diagram

در این نمودار ۴-۲-۴ نشان می دهد که کاربر چه کار هایی می تواند در سیستم انجام دهد در این سیستم کاربر پس از وارد شدن به سیستم می تواند کار هایی مانند : درج اطلاعات ، حذف اطلاعات ، مشاهده ی اطلاعات ، تغییر در اطلاعات و اضافه کردن توضیحات را برای هر یک از تجهیزات را انجام دهد . با استفاده از این نمودار می توانیم رفتار سیستم را در یک نگاه مشاهده نماییم .



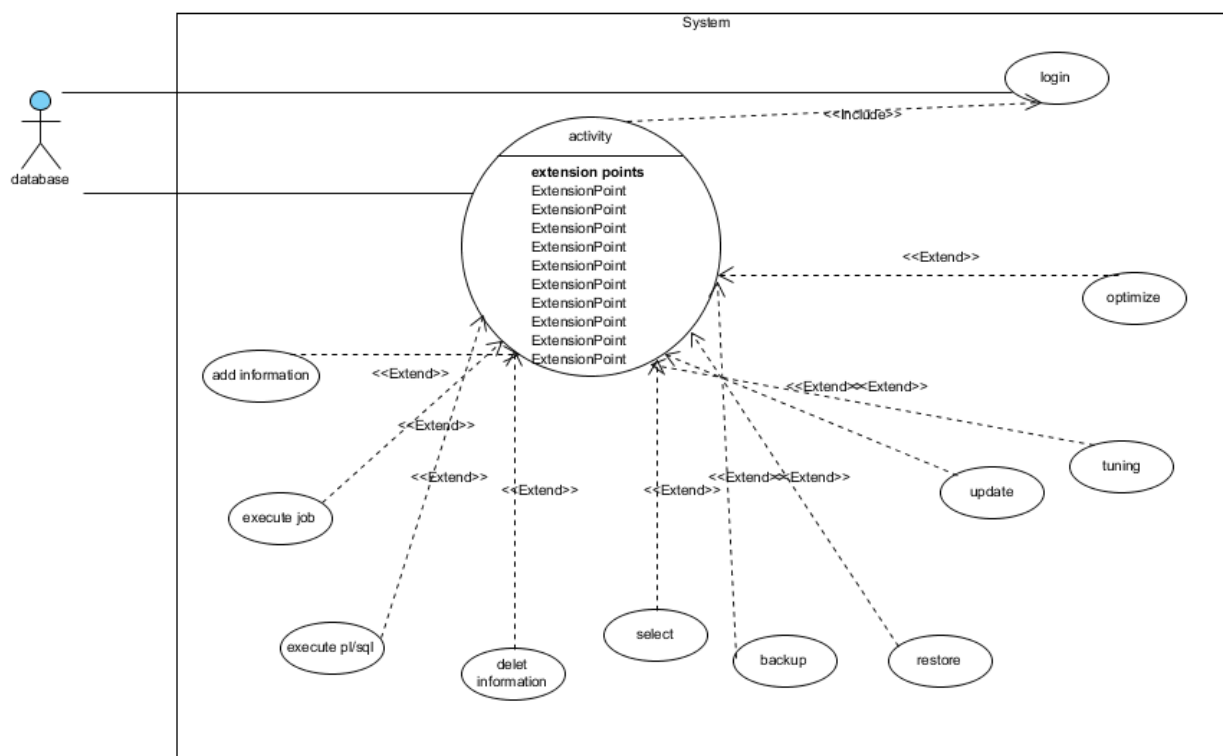
نمودار ۴-۲-۴ مربوط به فعالیت کاربر

این نمودار ۵-۲-۴ نشان می دهد که مدیر چه کار هایی می تواند در سیستم انجام دهد در این سیستم مدیر پس از وارد شدن به سیستم می تواند کار هایی مانند : درج اطلاعات ، حذف اطلاعات ، مشاهده ی اطلاعات ، تغییر در اطلاعات و اضافه کردن توضیحات را برای هر یک از تجهیزات را انجام دهد . با استفاده از این نمودار می توانیم رفتار سیستم را در یک نگاه مشاهده نماییم . همچنین در این سیستم مدیر می تواند دیتابیس را مدیریت نماید و می تواند سرویس را فعال و یا متوقف نماید.



۴-۲-۵ شکل نمودار مربوط به فعالیت مدیر

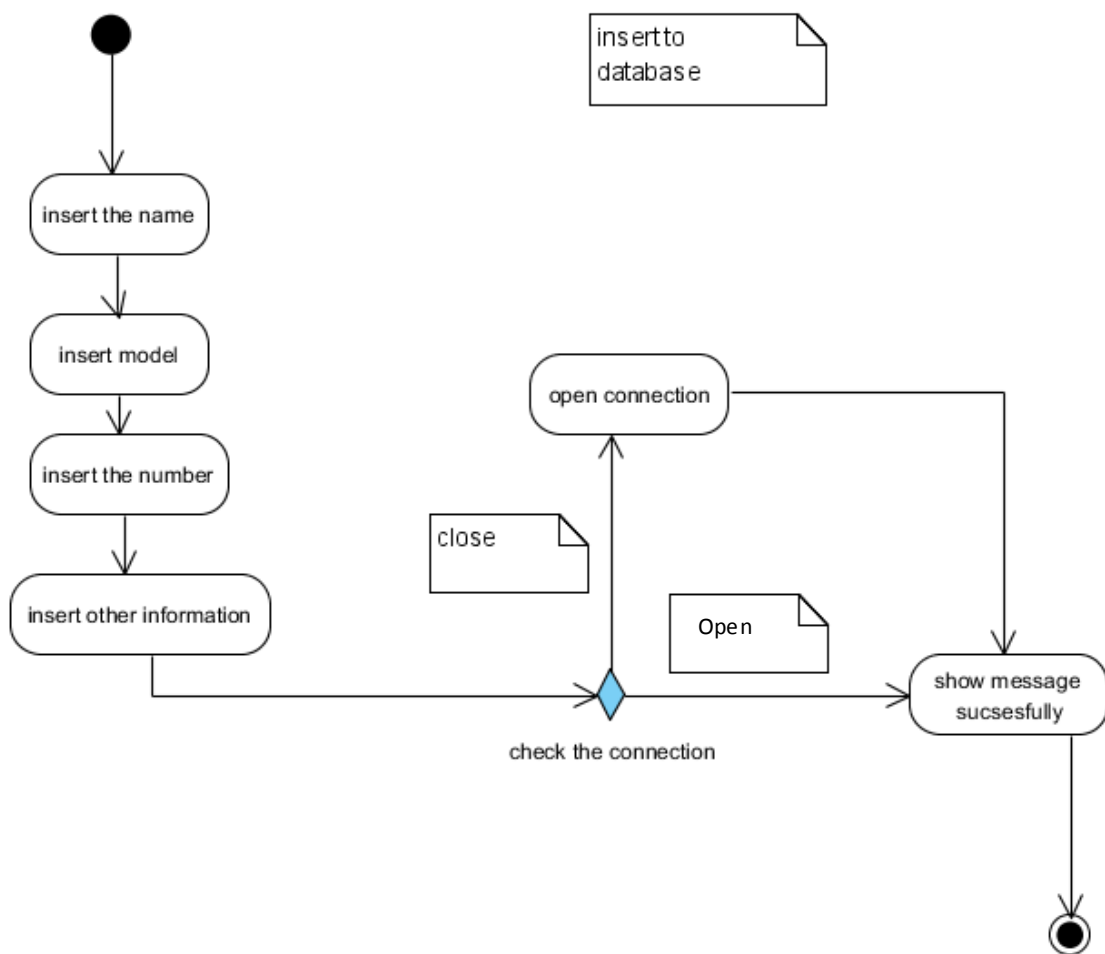
در این نمودار ۴-۲-۶ نشان می دهد که دیتابیس چه کار هایی می تواند در سیستم انجام دهد در این سیستم دیتابیس می تواند کار هایی مانند : درج اطلاعات ، حذف اطلاعات ، مشاهده ی اطلاعات ، تغییر در اطلاعات و اضافه کردن توضیحات را برای هر یک از تجهیزات را انجام دهد ؛ همچنین پایگاه داده عملیات مربوط به اجرای درخواست ها را انجام میدهد و بهترین روش را برای اجرای یک کوئری انتخاب می کند ؛ پایگاه داده در این سیستم برای اجرای بهتر یک درخواست از الگوریتم های مختلفی استفاده می کند.



شکل ۶-۲-۴ نمودار مربوط به کار های دیتابیس

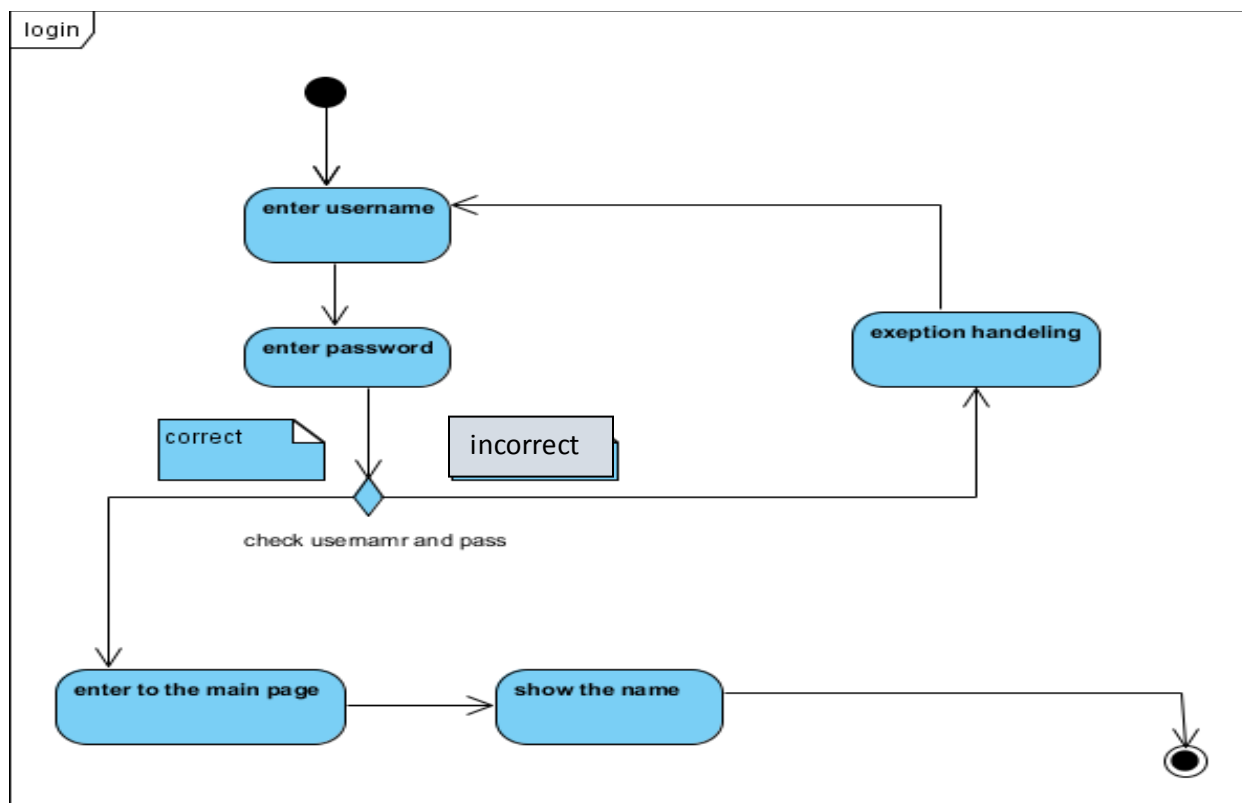
## Activity Diagram

این نمودار ۷-۲-۴ فعالیت هایی که برای عمل درج صورت می گیرد را نشان می دهد.. در این نمودار کاربر قبل از انجام عمل درج وارد سیستم می شود و عمل درج را انتخاب می کند ، سپس اطلاعات مربوط به نام ، مدل، شماره و الباقی اطلاعات را وارد می کند ، سپس نرم افزار ارتباط با پایگاه داده را بررسی می کند در صورتی که ارتباط برقرار بود اطلاعات وارد می شود و پیغام موفقیت آمیز برای کاربر نشان داده می شود و در صورتی که ارتباط بسته باشد ، ارتباط باز می شود و سپس عمل درج صورت می گیرد.



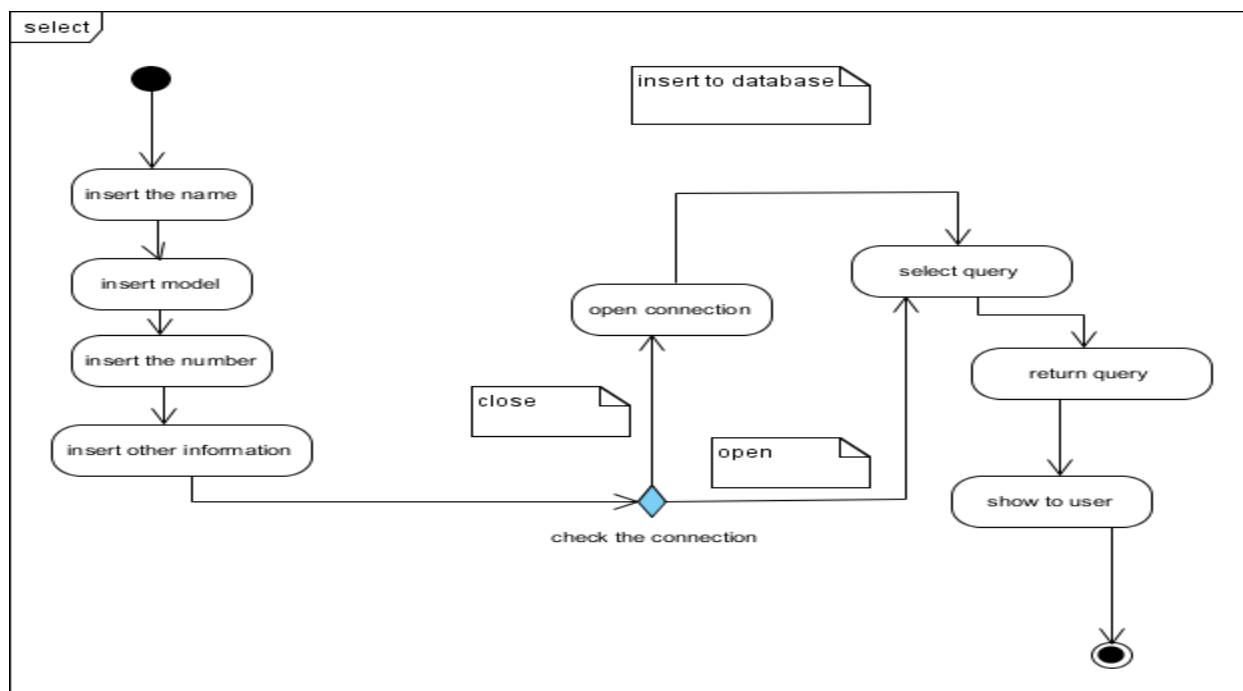
شکل ۷-۲-۴ نمودار مربوط به فعالیت درج

این نمودار ۸-۲-۴ مراحل مربوط به ورود به سیستم را نشان می دهد. در ابتدا کاربر نام کاربری و رمز خود را وارد می کند ، سپس سیستم نیز با اتصال به دیتابیس اطلاعات حساب کاربری را بررسی می کند در صورتی اطلاعات اشتباه بود از کاربر می خواهد که دوباره اطلاعات را وارد کند و اگر اطلاعات صحیح بود کاربر ویا مدیر وارد حساب کاربری می شود و اسم آن نمایش داده می شود.



شکل نمودار مربوط به فعالیت وارد شدن به سیستم ۸-۲-۴

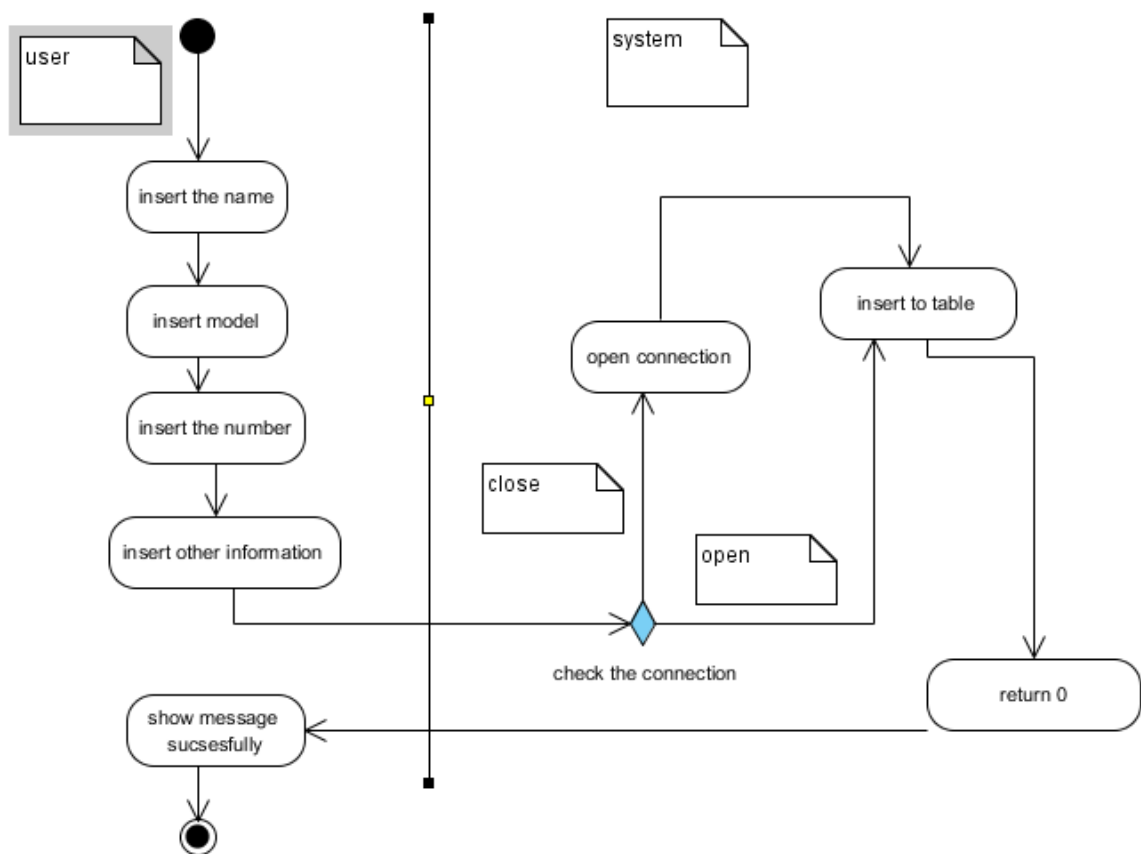
این نمودار ۹-۲-۴ مراحل خواندن اطلاعات از دیتابیس را نشان می دهد. ابتدا کاربر و یا مدیر با وارد کردن اطلاعات مربوط به دستگاهی که می خواهد اطلاعات آن را ذخیره کند کار را آغاز می کند. سپس ارتباط با دیتابیس بررسی می شود در صورتی که ارتباط برقرار بود جواب درخواست مورد نظر بر می گردد و اطلاعات در دیتابیس ذخیره می شود. و در صورتی که ارتباط برقرار نبود ارتباط برقرار خواهد شد و سپس درج اطلاعات انجام خواهد شد.



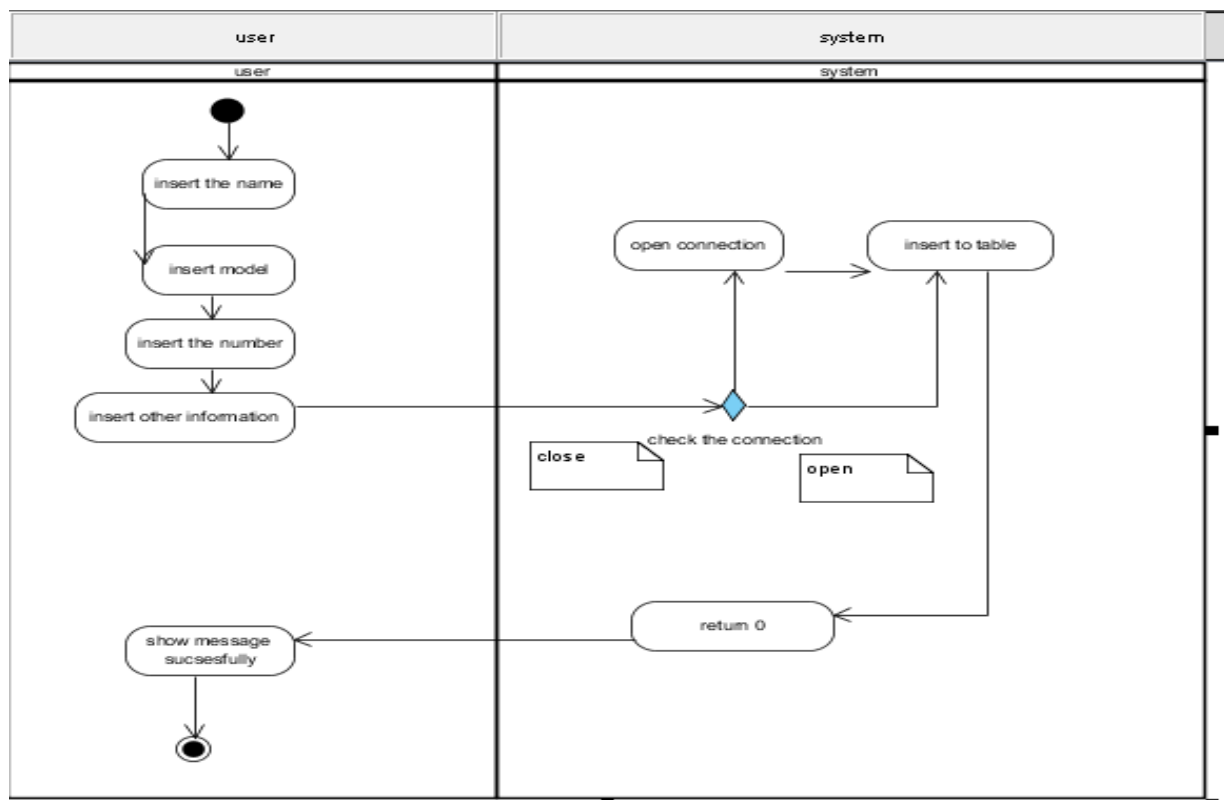
۹-۲-۴ شکل نمودار مربوط به فعالیت خواندن اطلاعات

## Swim lane Diagram

نمودارهای ۱۰-۲-۴ و ۱۱-۲-۴ همانند نمودار قسمت قبل می باشد با این تفاوت که نقش هر یک از قسمت ها در آن مشخص می باشد. ابتدا کاربر و یا مدیر با وارد کردن اطلاعات جهت ذخیره سازی کار را آغاز می کند و بعد ارتباط با دیتابیس بررسی می شود ؛ در صورتی که ارتباط برقرار بود اطلاعات وارد می شود . همانطور که در این نمودار مشاهده می کنید بخش های مربوط به ورودی اطلاعات در بخش نرم افزار سمت کاربر انجام می گیرد و مراحل مربوط به بررسی ارتباط و درج اطلاعات در نرم افزار سمت سرور صورت می گیرد و جواب حاصل از این تراکنش به کاربر نشان داده می شود.



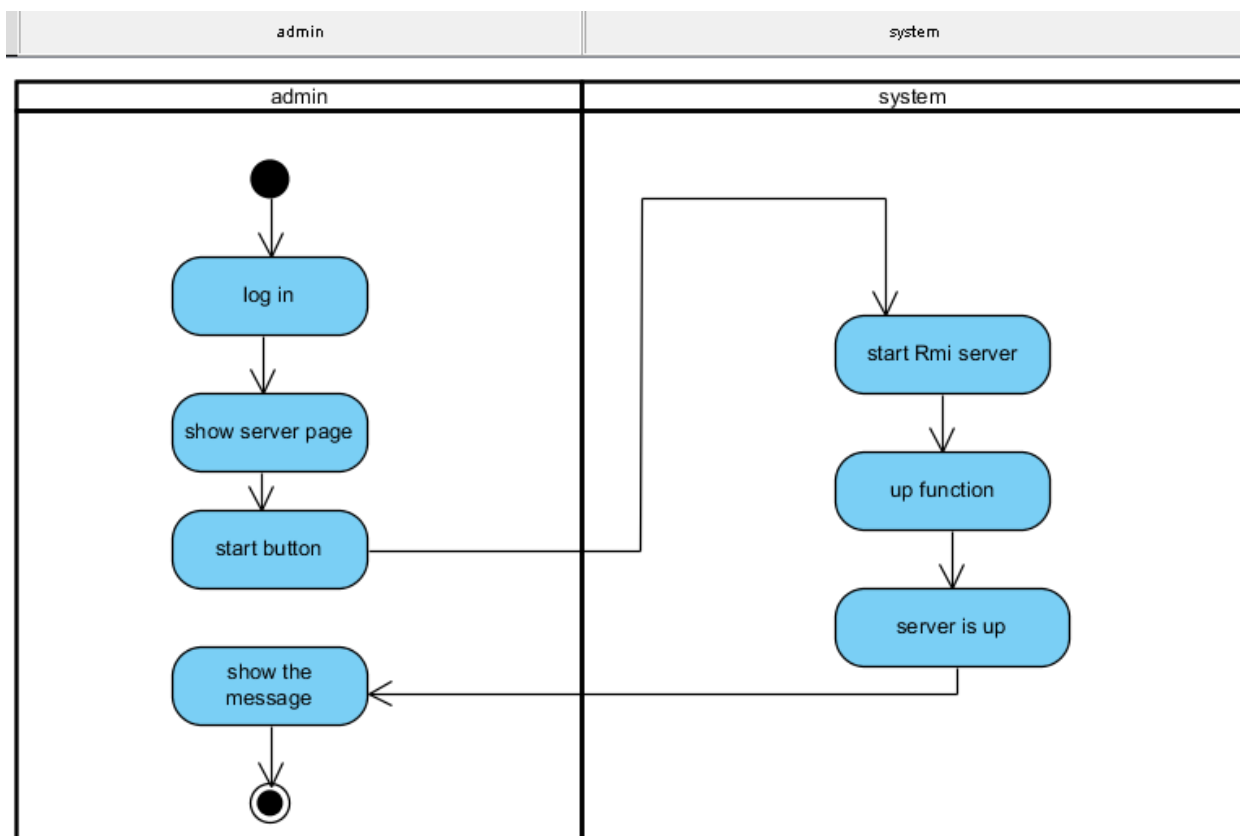
۱۰-۲-۴ شکل نمودار مربوط به فعالیت درج با نمایش وظایف هر بخش



۱۱-۲-۴ شکل نمودارمربوط به فعالیت درج با نمایش وظایف هر بخش

در نمودار ۱۱-۲-۴ مراحل مربوط به فعال کردن سرویس نشان داده شده است. در این نمودار مدیر ابتدا وارد سیستم می شود؛ سپس صفحه مربوط به مدیریت سرور نمایش داده می شود و مدیر دکمه مربوط به فعال کردن سرویس را می زند و بعد سرویس در قسمت برنامه ی سرور فعال می شود. و پیام فعال شدن سرویس به مدیر نمایش داده می شود.

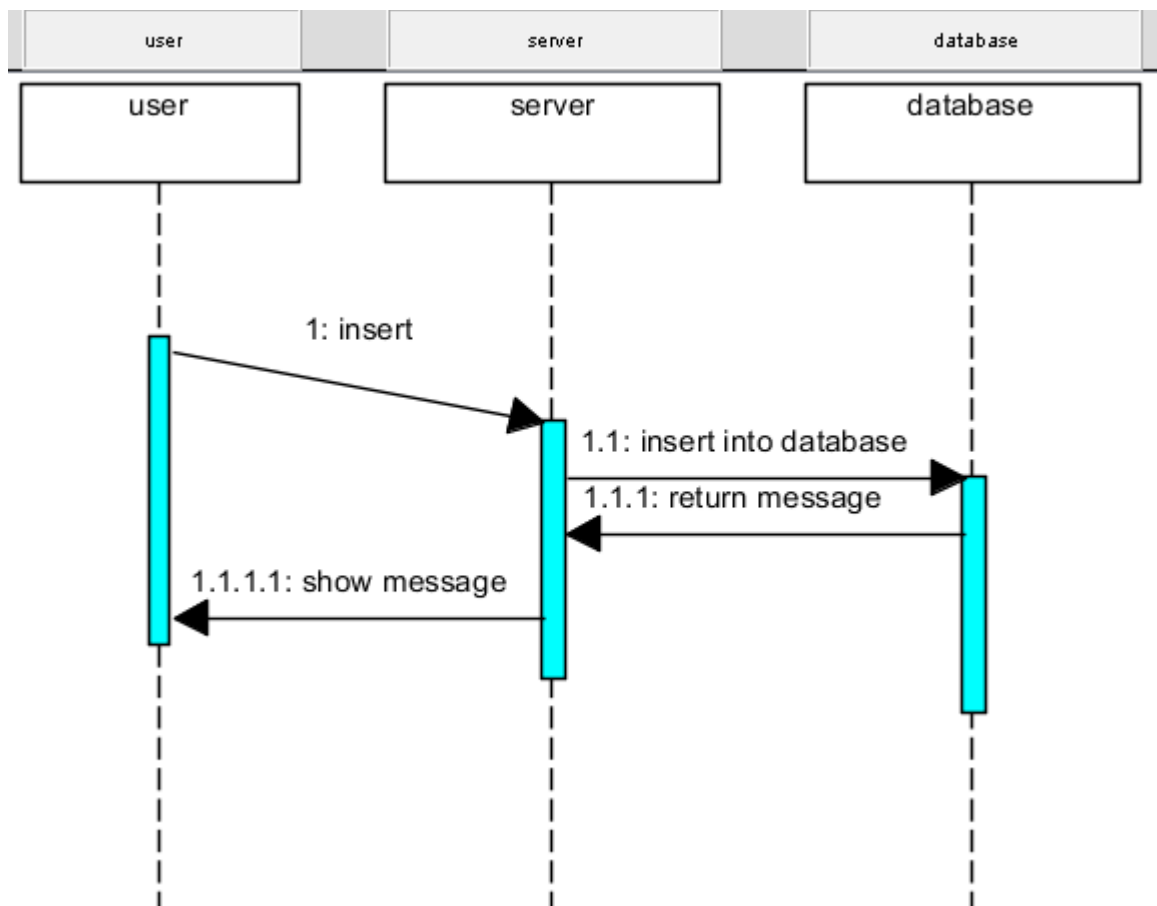




۱۲-۲-۴ شکل نمودار مربوط به فعالیت فعال کردن سرویس با نمایش وظایف هر بخش

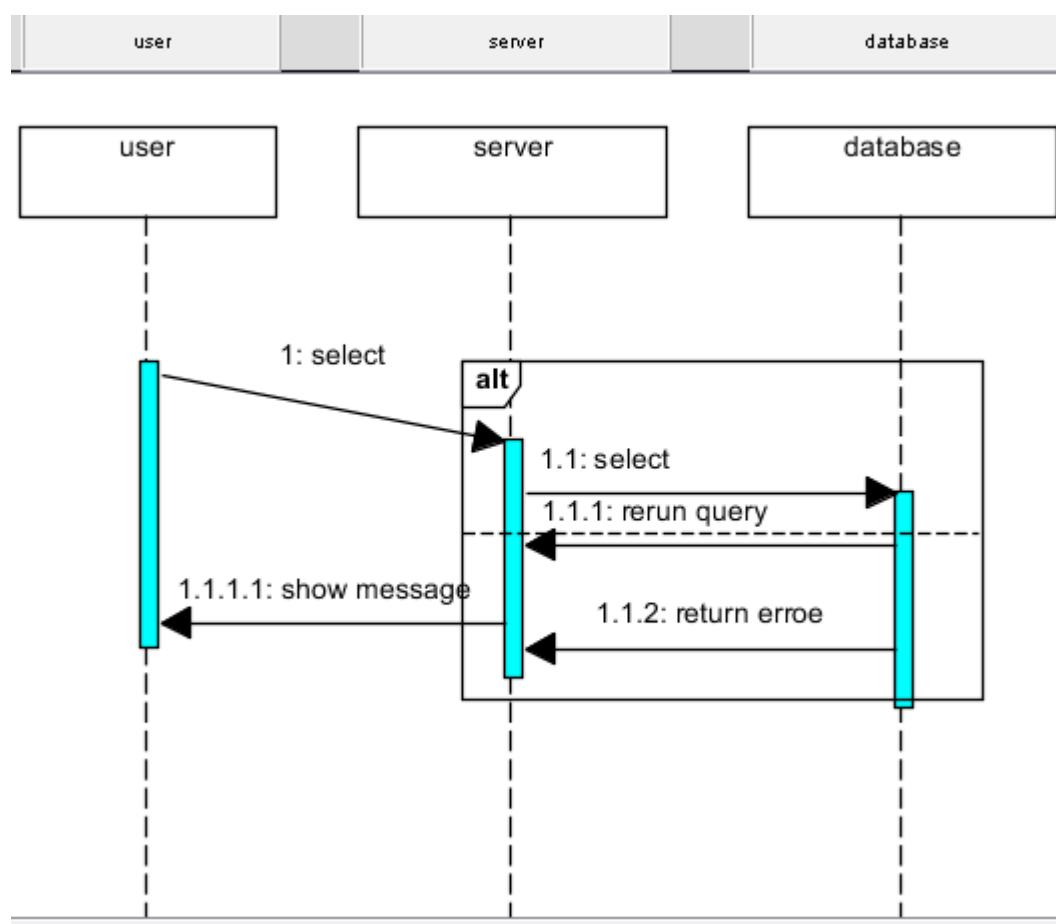
## Sequence diagram

این نمودار ۴-۲-۱۳ ارتباط بین کلاس های و بخش های سیستم را به صورت فراخوانی توابع نشان می دهد. ابتدا کاربر تابع درج از برنامه سمت سرور را فراخوانی می کند و این تابع درج نیز خود در دل خود یک تابع درج در دیتابیس را دارد که درخواست درج اطلاعات را به دیتابیس می فرستد و بعد از عمل درج پیغام مربوط به عمل درج را ابتدا به برنامه سرور ارسال می کند و سپس به برنامه کاربر برای نمایش به کاربر نشان داده می شود.



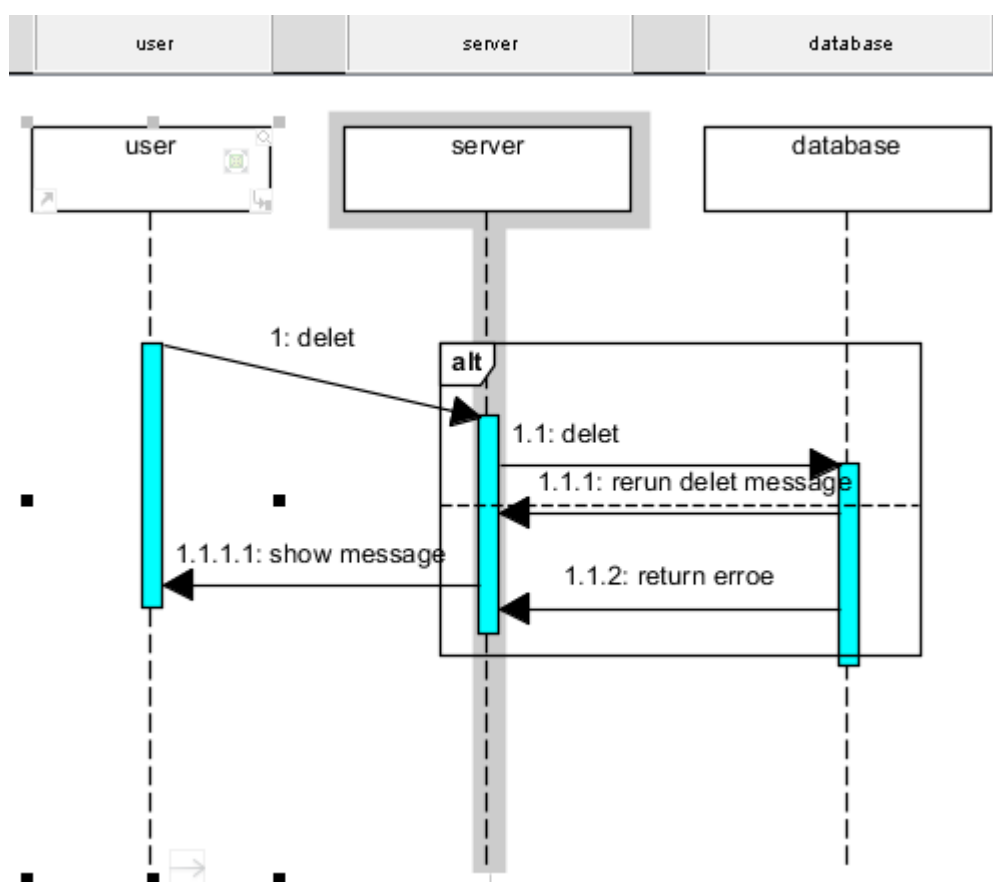
۴-۲-۱۳ شکل نمودار مربوط به درخواست درج اطلاعات با توجه به فراخوانی های هر بخش

این نمودار ۱۴-۲-۴ عملیات نمایش دادن اطلاعات را نشان می دهد. ابتدا تابع نمایش از برنامه سمت سرور فراخوانی می شود و این تابع نیز خود تابع دیگری برای برداشتن اطلاعات از دیتابیس فراخوانی می کند اگر id وارد شده درست بود اطلاعات به نرم افزار سمت سرور برگردانده می شود و بعد از آن جهت نمایش به کاربر به برنامه سمت کاربر انتقال داده می شود و اگر هم id وارد شده درست نباشد پیغام خطا به کاربر نمایش داده می شود.



۱۴-۲-۴ شکل نمودار مربوط به درخواست نمایش اطلاعات با توجه به فراخوانی های هر بخش

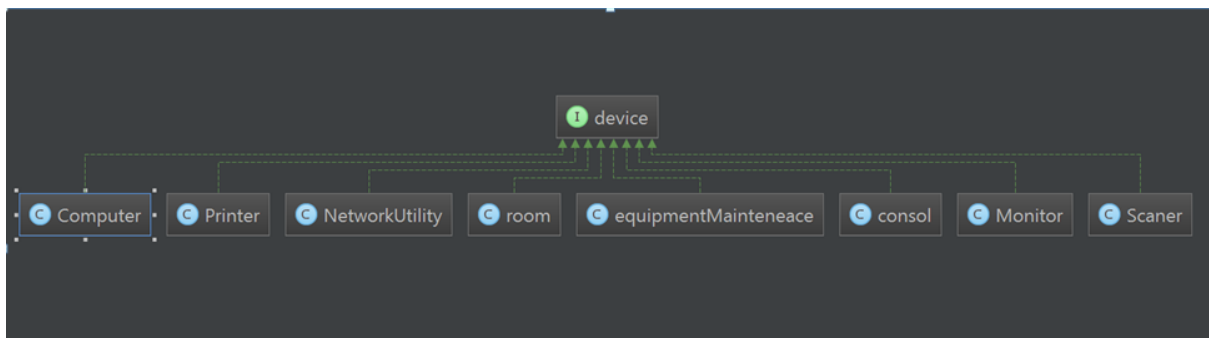
در این نمودار ۴-۲-۱۴ مراحل مربوط به حذف اطلاعات نشان داده شده است. سیستم به این صورت عمل می کند که ابتدا تابع حذف از برنامه سمت سرور فراخوانی می شود و این تابع خود نیز تابع حذف از پایگاه داده را فراخوانی می کند؛ در صورتی که id وارد شده درست باشد اطلاعات از دیتابیس حذف خواهد شد و پیام آن به کاربر نمایش داده می شود و اگر id وارد شده اشتباه باشد، پیغام خطای آن به کاربر نمایش داده می شود.



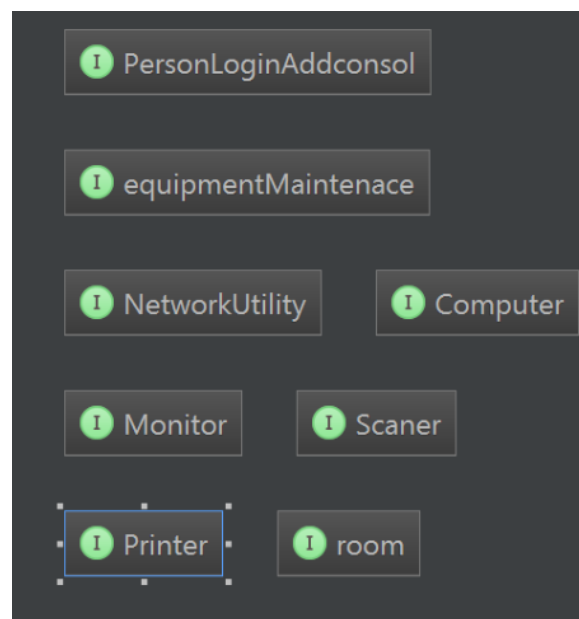
۴-۲-۱۵ شکل نمودار مربوط به درخواست حذف اطلاعات با توجه به فراخوانی های هر بخش

## Class diagram

این نمودار ۱۵-۲-۴ کلاس های موجود در سیستم را نشان می دهد . در این سیستم ما کلاس های مختلفی داریم . یکسری از این کلاس ها مربوط به پایگاه داده می باشد و یکسری از کلاس ها برای ارتباط بین بخش نرم افزار سرور و نرم افزار کاربر است و تعدادی از کلاس ها در اینجا برای نمایش دادن و ساخت واسط کاربر است که تعداد آن نیز در این سیستم کم است . تعدادی دیگر از این کلاس ها وظیفه ی کنترل بخش های مختلف را بر عهده دارند. در این شکل کلاس های مربوط به این پروژه را می بینیم .



نمودار ۱۵-۲-۴ نمایش کلاس های سیستم (الف)



نمودار ۱۵-۲-۴ نمایش کلاس های سیستم (ب)



## تحلیل مربوط به دیتابیس

۳-۴

در این بخش از فصل چهارم به بررسی و تحلیل مربوط به دیتابیس خواهیم پرداخت برای بررسی دقیق تر می توانید به کتاب اصول طراحی پایگاه داده اثر آقای دکتر روحانی رانکوهی مراجعه فرمایید. در ادامه به توضیح برخی از پیشنهادها برای طراحی پایگاه داده می پردازیم. برای طراحی پایگاه داده ها باید با برخی از مفاهیم های آن آشنا شویم و در آخر این بخش طراحی این سیستم را نیز نشان خواهیم داد.

تمامی این مباحثی که مطرح می شود بین تمامی سیستم های مدیریت پایگاه داده ها مشترک می باشد و تمامی طراحی ها را می توان با سیستم ها مختلف پیاده سازی نمود . برای مطالعه بیشتر به بخش پیوست مراجعه فرمایید.

### Table ( جدول )

در سیستم های مدیریت پایگاه داده رابطه ای اطلاعات درون جداول ذخیره می شوند و از جدول ها باز یابی می شوند . جدول ها شامل تعدادی ستون هستند که درواقع صفت های ما هستند ؛ همچنین دارای تعدادی سطر می باشد که تعداد سطر ها نشان دهنده ی تعداد رکورد های ما می باشد.

### Entity (موجودیت )

در سیستم های پایگاه داده ای رابطه ای موجودیت به چیزی گفته می شود که می خواهیم اطلاعاتی در مورد آن در پایگاه داده ذخیره کنیم . موجودیت ها دارای تعدادی صفات هستند و صفت ها نیز انواع مختلفی دارند که برای مطالعه بیشتر می توانید به پیوست مراجعه فرمایید. همچنین موجودیت ها نیز دارای انواع مختلفی هستند که در ادامه توضیح خواهیم داد

موجودیت ضعیف : به موجودیتی گفته می شود که وجود آن وابسته به موجودیت دیگر است .

موجودیت قوی : به موجودیتی گفته می شود که وجود آن مستقل می باشد و وابسته نیست.

## Relation (رابطه )

به تعامل بین چند موجودیت رابطه گفته می شود . یعنی ۲ یا چند موجودیت با هم در ارتباط هستند و

عملی بین این دو اتفاق می افتد. در مدل رابطه ای سه نوع رابطه بین موجودیت ها وجود دارد

۱- رابطه ی ۱:۱      ۲- رابطه ی ۱:n      ۳- رابطه ی n:m

برای طراحی پایگاه داده های باید تمامی اجزای طراحی را بشناسم و همچنین باید تمامی روش ها را نیز آشنا باشیم چون این طراحی که انجام می دهیم در آخر به جدول های دیتابیس ما تبدیل می شود .

طراحی که انجام می دهیم باید طراحی خوبی باشد چون برای توسعه دیتابیس در آینده اگر طراحی خوب باشد توسعه به راحتی امکان پذیر خواهد بود و اگر طراحی بد باشد باعث می شود که توسعه به سختی امکان پذیر باشد و در بیشتر مواقع نیز امکان پذیر نباشد . از دیگر مزایای طراحی خوب این است که سرعت اجرای یک دستور و یک کوئری را کاهش می دهد لذا در طراحی پایگاه داده باید بسیار دقت شود زیرا به طور مستقیم بر روی کارایی سیستم تاثیر می گذارد.

در ادامه بخش به معرفی نمودار طراحی پایگاه داده خواهیم پرداخت و در آخر فصل نیز نمودار مربوط به این سیستم را نشان خواهیم داد . بدیهی است که این سیستم به دلیل کار کردن مدام با پایگاه داده باید از طراحی خوبی برخوردار باشد و چون این سیستم یک سیستم توزیع شده است دیتابیس باید از سرعت بالایی برای پاسخ گویی برخوردار باشد و بخش اعظم این کار را در طراحی

خوب می توانیم فراهم کنیم و بخش دگر را نیز می توانیم با تنظیمات مربوط به query performance انجام دهیم که برای مطالعه بیشتر به کتب پایگاه داده مراجعه فرمایید.

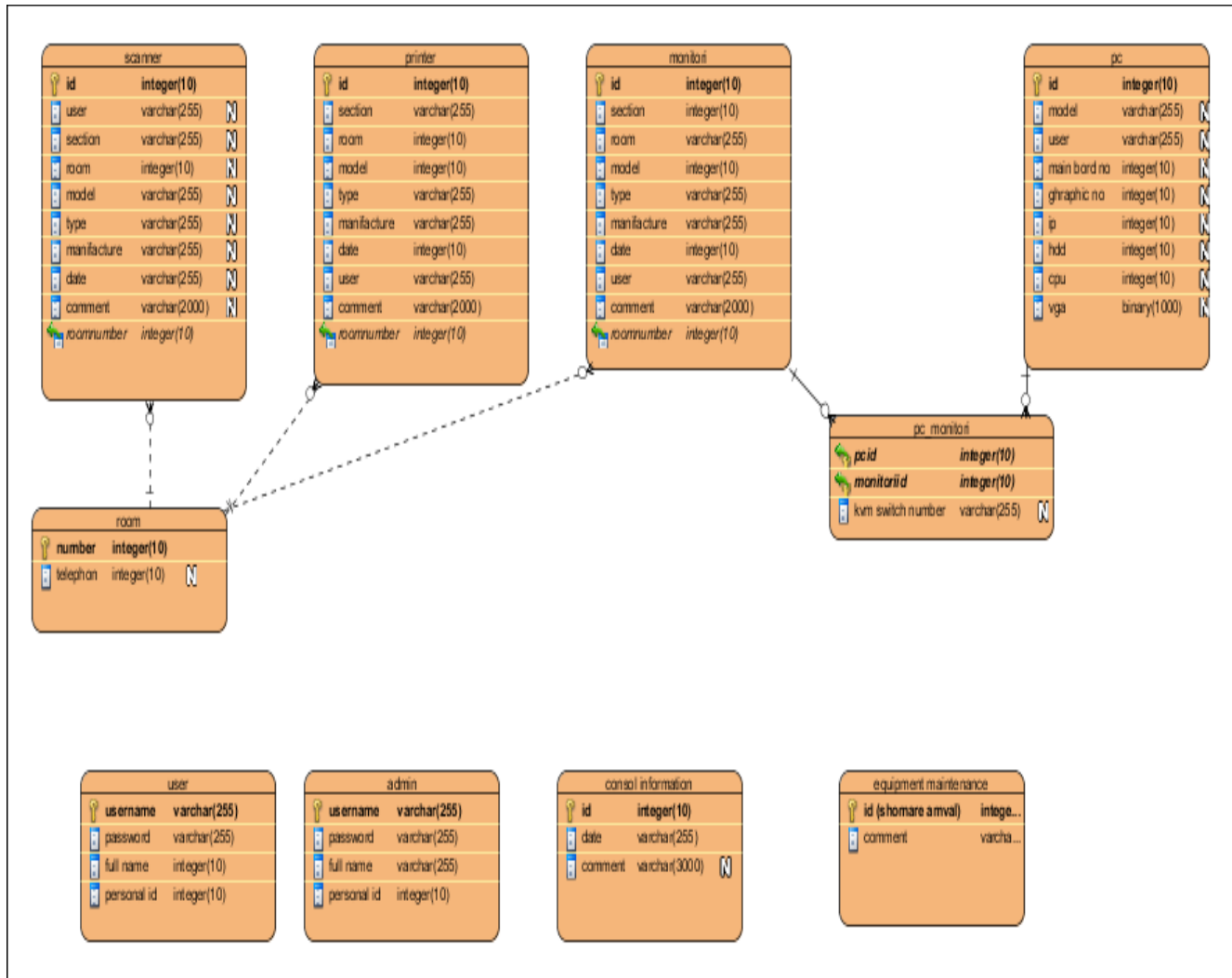
## **(Entity Relation Diagram) ERD**

این نوع از نمودار به بررسی موجودیت های موجود در پایگاه داده می پردازد . این نمودار طراحی پایگاه داده را نشان می دهد که شامل یک سری از موجودیت های سیستم می باشد . در این نمودار ویژگی ها و صفت های هر یک از موجودیت ها نشان داده می شود ، همچنین رابطه بین هر یک از موجودیت ها نیز نشان می دهد.

با استفاده از این نمودار می توانیم جدول های دیتابیس را طراحی کنیم و کلید های اصلی را در آن مشخص کنیم . این نمودار نحوه ذخیره سازی اطلاعات در سیستم را با استفاده از روابط بین موجودیت ها و روابط بین جدول ها نشان می دهد.



## نمودار ERD طراحی شده برای این سیستم



۱-۳-۴ نمودار مربوط طراحی پایگاه داده پروژه

## فصل پنجم

### جمع بندی و پیش بینی کارهای آتی

آنچه خواهید خواند :

۱-۵ جمع بندی مطالب و نتیجه گیری

۲-۵ پیش بینی کارهای آتی

## ۵-۱ جمع بندی مطالب و نتیجه گیری

در این بخش جمع بندی مطالب مطرح شده در این فصول را بیان خواهیم نمود.

در این پایان نامه به بررسی تعدادی از زبان های برنامه نویسی پرداختیم و نقاط ضعف و قوت هر یک را بیان کردیم و محیط ها و شرایط استفاده از آن ها را نیز بیان کردیم و بررسی کردیم که کدام یک برای پروژه ما مناسب تر است. همچنین معیار های بررسی زبان های برنامه نویسی را نیز بیان کردیم. در ادامه بخش ها به تعریف پایگاه داده ها پرداختیم و مفاهیم پایه ای آن را بررسی کردیم و طراحی خوب را توضیح دادیم و در آخر بخش طراحی پایگاه داده ی مربوط به پروژه را انجام دادیم. همچنین اصول مهندسی نرم افزار را در این پایان نامه شرح دادیم و دیاگرام های مربوط به پروژه را نیز رسم کردیم، همچنین از فناوری های استفاده در این پروژه هم نام بردیم و به بررسی آن ها پرداختیم. در ابتدای این پایان نامه به بررسی سیستم های موجود پرداختیم و انواع آن ها را معرفی کردیم. با توجه به مطالب توضیح داده شده و بحث شده در این پایان نامه می توانیم به صراحت ببینیم که سیستم های کامپیوتری و الکترونیکی چه تاثیر به سزایی در پیشرفت زندگی و ارتقاء سطح کیفی زندگی انسان ها دارند و باعث صرفه جویی در زمان و هزینه می شود. البته در این جا باید نکته ای را بیان کنیم بعضی از این سیستم های الکترونیکی مانند شبکه های اجتماعی گرچه سودمند می باشند و جذاب برای مخاطبین اما مشکلاتی نیز دارند که یکی از مشکلات بزرگ آن ها اعتیادی است که استفاده کنندگان از اینگونه سیستم ها به آن پیدا می کنند. لذا باید به نوع استفاده از این سیستم ها توجه نمود و نحوه استفاده از این سیستم ها را کنترل نمود چون استفاده بیش از حد از این سیستم ها باعث اتلاف زمان خواهد شد.

البته سیستم هایی که به عنوان سیستم های مدیریت و یا اتوماسیون استفاده می شوند بسیار در بهبود عملکرد کارها و سرعت بخشیدن به امور موثر هستند. و فراگیر شدن اینگونه سیستم ها باعث افزایش کارایی می شود.

مطلب دیگری که باید در طراحی اینگونه سیستم ها در نظر گرفت این است که امنیت این سیستم ها باید بالا باشد. در طراحی این سیستم ها باید به نکات امنیتی توجه داشت و مهمترین بخش امنیت مربوط به امنیت پایگاه داده می باشد.

با اصول و ابزار های مورد بررسی در این پایان نامه می توانیم سیستم های مدیرتی و نرم افزار های تجاری تولید کنیم . البته لازم به ذکر است که برای تولید انواع نرم افزار ها با توجه به نوع نرم افزار باید ابزار های لازم را انتخاب کنیم . برای مثال برای تولید نرم افزار های سیستمی مانند سیستم های عامل و یا داریور ها باید از زبان برنامه نویسی C استفاده کنیم . با بررسی مطالب در می یابیم که در دنیای کامپیوتر ابزار های زیادی وجود دارد و هنر یک مهندس خوب این است که با توجه به نیاز های موجود و نوع پروژه، بهترین ابزار را برای تولید انتخاب کند ؛ لذا برای این کار نیاز به دانش بالا در رشته کامپیوتر و نرم افزار ضروری می باشد.

امیدوار هستم که توانسته باشم در این چند فصل اطلاعات خوبی را به خواننده عزیز در مورد با رشته کامپیوتر و تولید نرم افزار ها و ابزار های لازم انتقال داده باشم و گامی موثر در انتقال دانش خود به دیگران برداشته باشم .

## ۵-۲ پیش بینی کار های آتی

در این بخش از فصل پنجم به پیش بینی کار های آتی این سیستم خواهیم پرداخت

این سیستم که دارای اطلاعاتی از کامپیوتر ها و تجهیزات الکترونیکی است ، قابلیت هایی دارد که می توان آن را در بسیاری از مراکز استفاده نمود لذا پیش بینی می شود پس از تست های مربوط به این سیستم و پس از معرفی آن به روش درست ؛ این سیستم فراگیر خواهد شد.

از این سیستم نیز می توانیم با اندکی تغییر سیستم دیگر تولید کنیم . برای مثال می توانیم با توسعه دادن آن یک سیستم مدیریت اموال که قدرتمند و جامع است تولید نماییم. همچنین می توانیم با استفاده از این نرم افزار یک پیش بینی در مورد با آینده محصولاتی که در مورد آن ها در این سیستم اطلاعات داریم استفاده کنیم .البته همانطور که گفته شد در فصل های قبل، این سیستم برای مراکز بزرگ بسیار مفید خواهد بود . همچنین با استفاده از این سیستم می توان در بسیاری از هزینه های جاری صرفه جویی نمود.

البته بدیهی است که با پیشرفتی که سیستم های و ابزار های کامپیوتری دارند باید در آینده نزدیک نسبت به ارتقاء این سیستم و استفاده از فناوری های پیشرفته اقدام شود . لذا این سیستم از ورژن های مختلفی بر خوردار خواهد بود .

با تولید این پروژه ، این نرم افزار به مدت ۳ تا ۶ ماه در یک مرکز که دارای تعدادی زیادی کامپیوتر است تست خواهد شد و بعد از بررسی هایی که صورت خواهد گرفت عیب های آن مشخص خواهد شد و عیب یابی آن صورت خواهد گرفت در فاز های بعدی این سیستم می توانیم آن را توسعه دهیم و به یک سیستم جامع در مدیریت اموال تبدیل کنیم که اطلاعات تمامی اموال را نگه دارد . برای فراگیر شدن این سیستم باید از یک واسط گرافیکی مناسب برای آن استفاده نمود که بعد از انجام مراحل تست این نرم افزار، یک واسط گرافیکی خوب و زیبا که تحت وب هست را با استفاده از زبان های برنامه نویسی سمت کابر برای آن تولید خواهیم نمود تا این سیستم گسترش پیدا کند. . همچنین چون بخش سرور این سیستم به زبان جاوا می باشد این قابلیت را دارد که بر روی تمامی سیستم های عامل اجرا شود و با توجه به این که بسیاری از سرور ها لینوکس می باشد می توانیم آن را بر روی سرور های لینوکس نیز اجرا کنیم .

از دیگر کار هایی پیش روی آینده این سیستم ساخت و طراحی یک برنامه اندروید است که به این سیستم متصل می شود و این امکان را به کاربر می دهد تا با نصب این نرم افزار بر روی موبایل خود تمامی تراکنش ها خود را به راحتی انجام دهد.. ان شا الله بعد از انجام مراحل تست و عیب یابی مراحل مربوط به توسعه این سیستم انجام خواهد شد. همچنین در آینده می توان عکس هایی از تجهیزات را در سیستم ذخیره کنیم که با توجه به دیتابیس استفاده شده در این پروژه انجام دادن این کار به سهولت امکان پذیر می باشد .

## فصل ششم

### مراجع

۱-۶ مراجع انگلیسی

۲-۶ مراجع فارسی

۳-۶ ادرس سایت ها

- 1) J. Graba, "Remote Method Invocation, java database connectivity" in introduction to Network programming in java, south torkshire, UK, Sheffield hallam university, extera pub, 2013
- 2) V. Sarcar, "singleton pattern, factory pattern, builder pattern" in java design patterns, new York, USA, Appress pub, 2015
- 3) H. Deitel and p. Deitel, "java programming" in Java daitel & daitel 10<sup>th</sup> edition, California, USA, deitel pub, 2014
- 4) M. weiss, "sorting" in data structures and algorithm analysis in java 3rd edition, florida, USA, florida international university, pearson pub, 2011
- 5) A. B. Downey and C. Mayfield, Think Java, California, USA, oreilly pub, 2016
- 6) B. Dathan and S. Ramnath, "basics of object-oriented programming, relation between classes" in Object-Oriented Analysis, Design and Implementation, 2nd Edition, new York, USA, springer pub, 2016
- 7) R. S. Pressman, "modeling, the software process, agile development" in Software engineering A practitioner's Approach 7<sup>th</sup> edition, new York, NY 10020, USA, McGraw-hill companies Inc, 2010



(۱) . جعفر نژاد قمی " طراحی و پیاده سازی زبان های برنامه سازی " . ۲۰۱۱

(۲) ابراهیم زاده قلزم " آشنایی با الگوریتم ها (CLRS) . " ۲۰۱۰

[www.codeproject.com](http://www.codeproject.com)

[www.oracle.com](http://www.oracle.com)

[www.stackoverflow.com](http://www.stackoverflow.com)

[www.stachexchange.com](http://www.stachexchange.com)

[www.p30download.com](http://www.p30download.com)

[www.github.com](http://www.github.com)

[www.tutorialspoint.com](http://www.tutorialspoint.com)

[www.javaworld.com](http://www.javaworld.com)

[www.dzone.com](http://www.dzone.com)

[www.wikipedia.com](http://www.wikipedia.com)

[www.coursera.org](http://www.coursera.org)

[www.apple.com](http://www.apple.com)

[www.java.net](http://www.java.net)

[www.java2s.com](http://www.java2s.com)

[www.javaos.com](http://www.javaos.com)

[www.lynda.com](http://www.lynda.com)

[www.javacoffeebreak.com](http://www.javacoffeebreak.com)

[www.javapoint.com](http://www.javapoint.com)

[www.w3schools.com](http://www.w3schools.com)

[www.mysql.com](http://www.mysql.com)

[www.msdn.microsoft.com](http://www.msdn.microsoft.com)

[www.allitbooks.com](http://www.allitbooks.com)

[www.bookboon.com](http://www.bookboon.com)

[www.onlineprogrammingbooks.com](http://www.onlineprogrammingbooks.com)

[www.sourcebaran.com](http://www.sourcebaran.com)

[www.barnamenevice.com](http://www.barnamenevice.com)

[www.maktabkhooneh.org](http://www.maktabkhooneh.org)

[www.javacup.ir](http://www.javacup.ir)

[www.docs.oracle.com](http://www.docs.oracle.com)

[www.jetbrain.com](http://www.jetbrain.com)

[www.netbeans.com](http://www.netbeans.com)

[www.planetcassandra.org](http://www.planetcassandra.org)

[www.nosql-database.org](http://www.nosql-database.org)

<http://www.hpkclasses.ir>

## پیوست

مطالب و توضیحات اضافه

فرم های مربوط به نرم افزار

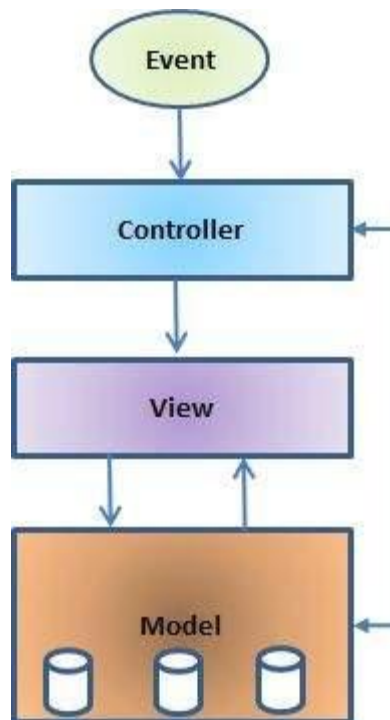
کد های پیاده سازی شده

## معماری MVC (MVC ARCHITECTURE)

Model View Controller or MVC as it is popularly called, is a software design pattern for developing web applications. A Model View Controller pattern is made up of the following three parts:

- **Model** - The lowest level of the pattern which is responsible for maintaining data.
- **View** - This is responsible for displaying all or a portion of the data to the user.
- **Controller** - Software Code that controls the interactions between the Model and View.

MVC is popular as it isolates the application logic from the user interface layer and supports separation of concerns. Here the Controller receives all requests for the application and then works with the Model to prepare any data needed by the View. The View then uses the data prepared by the Controller to generate a final presentable response. The MVC abstraction can be graphically represented as follows.



### The model

The model is responsible for managing the data of the application. It responds to the request from the view and it also responds to instructions from the controller to update itself.

### The view

A presentation of data in a particular format, triggered by a controller's decision to present the data. They are script based templating systems like JSP, ASP, PHP and very easy to integrate with AJAX technology.

### The controller

The controller is responsible for responding to user input and perform interactions on the data model objects. The controller receives the input, it validates the input and then performs the business operation that modifies the state of the data model.

روش \_ RMI (Remote method invocation)

# An Overview of RMI Applications

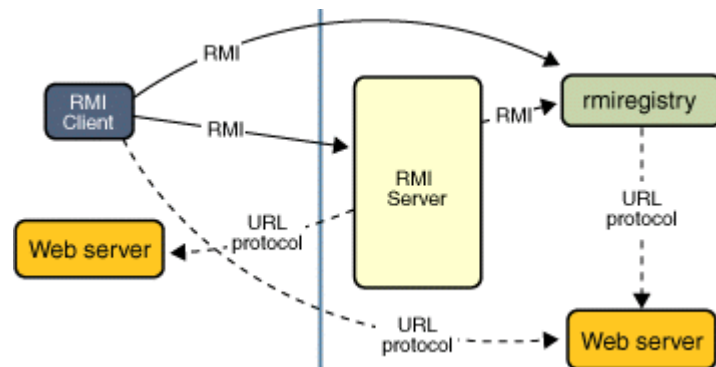
RMI applications often comprise two separate programs, a server and a client. A typical server program creates some remote objects, makes references to these objects accessible, and waits for clients to invoke methods on these objects. A typical client program obtains a remote reference to one or more remote objects on a server and then invokes methods on them. RMI provides the mechanism by which the server and the client communicate and pass information back and forth. Such an application is sometimes referred to as a *distributed object application*.

Distributed object applications need to do the following:

- **Locate remote objects.** Applications can use various mechanisms to obtain references to remote objects. For example, an application can register its remote objects with RMI's simple naming facility, the RMI registry. Alternatively, an application can pass and return remote object references as part of other remote invocations.
- **Communicate with remote objects.** Details of communication between remote objects are handled by RMI. To the programmer, remote communication looks similar to regular Java method invocations.
- **Load class definitions for objects that are passed around.** Because RMI enables objects to be passed back and forth, it provides mechanisms for loading an object's class definitions as well as for transmitting an object's data.

The following illustration depicts an RMI distributed application that uses the RMI registry to obtain a reference to a remote object. The server calls the registry to associate (or bind) a name with a remote object. The client looks up the remote object by its name in the server's registry and then invokes a method

on it. The illustration also shows that the RMI system uses an existing web server to load class definitions, from server to client and from client to server, for objects when needed.



## Advantages of Dynamic Code Loading

One of the central and unique features of RMI is its ability to download the definition of an object's class if the class is not defined in the receiver's Java virtual machine. All of the types and behavior of an object, previously available only in a single Java virtual machine, can be transmitted to another, possibly remote, Java virtual machine. RMI passes objects by their actual classes, so the behavior of the objects is not changed when they are sent to another Java virtual machine. This capability enables new types and behaviors to be introduced into a remote Java virtual machine, thus dynamically extending the behavior of an application. The compute engine example in this trail uses this capability to introduce new behavior to a distributed program.

## Remote Interfaces, Objects, and Methods

Like any other Java application, a distributed application built by using Java RMI is made up of interfaces and classes. The interfaces declare methods. The classes implement the methods declared in the interfaces and, perhaps, declare additional methods as well. In a distributed application, some implementations might reside in some Java virtual machines but not others. Objects with

methods that can be invoked across Java virtual machines are called *remote objects*.

An object becomes remote by implementing a *remote interface*, which has the following characteristics:

- A remote interface extends the interface `java.rmi.Remote`.
- Each method of the interface declares `java.rmi.RemoteException` in its throws clause, in addition to any application-specific exceptions.

RMI treats a remote object differently from a non-remote object when the object is passed from one Java virtual machine to another Java virtual machine. Rather than making a copy of the implementation object in the receiving Java virtual machine, RMI passes a remote *stub* for a remote object. The stub acts as the local representative, or proxy, for the remote object and basically is, to the client, the remote reference. The client invokes a method on the local stub, which is responsible for carrying out the method invocation on the remote object.

A stub for a remote object implements the same set of remote interfaces that the remote object implements. This property enables a stub to be cast to any of the interfaces that the remote object implements. However, *only* those methods defined in a remote interface are available to be called from the receiving Java virtual machine.

## **Creating Distributed Applications by Using RMI**

Using RMI to develop a distributed application involves these general steps:

1. Designing and implementing the components of your distributed application.
2. Compiling sources.



3. Making classes network accessible.
4. Starting the application.

## Designing and Implementing the Application Components

First, determine your application architecture, including which components are local objects and which components are remotely accessible. This step includes:

- **Defining the remote interfaces.** A remote interface specifies the methods that can be invoked remotely by a client. Clients program to remote interfaces, not to the implementation classes of those interfaces. The design of such interfaces includes the determination of the types of objects that will be used as the parameters and return values for these methods. If any of these interfaces or classes do not yet exist, you need to define them as well.
- **Implementing the remote objects.** Remote objects must implement one or more remote interfaces. The remote object class may include implementations of other interfaces and methods that are available only locally. If any local classes are to be used for parameters or return values of any of these methods, they must be implemented as well.
- **Implementing the clients.** Clients that use remote objects can be implemented at any time after the remote interfaces are defined, including after the remote objects have been deployed.

## Compiling Sources

As with any Java program, you use the javac compiler to compile the source files. The source files contain the declarations of the remote interfaces, their implementations, any other server classes, and the client classes.

**Note:** With versions prior to Java Platform, Standard Edition 5.0, an additional step was required to build stub classes, by using the `rmic` compiler. However, this step is no longer necessary.

---

## **Making Classes Network Accessible**

In this step, you make certain class definitions network accessible, such as the definitions for the remote interfaces and their associated types, and the definitions for classes that need to be downloaded to the clients or servers. Classes definitions are typically made network accessible through a web server.

## **Starting the Application**

Starting the application includes running the RMI remote object registry, the server, and the client.

The rest of this section walks through the steps used to create a compute engine.

## **Building a Generic Compute Engine**

This trail focuses on a simple, yet powerful, distributed application called a *compute engine*. The compute engine is a remote object on the server that takes tasks from clients, runs the tasks, and returns any results. The tasks are run on the machine where the server is running. This type of distributed application can enable a number of client machines to make use of a particularly powerful machine or a machine that has specialized hardware.

The novel aspect of the compute engine is that the tasks it runs do not need to be defined when the compute engine is written or started. New kinds of tasks can be created at any time and then given to the compute engine to be run. The only requirement of a task is that its class implement a particular interface. The

code needed to accomplish the task can be downloaded by the RMI system to the compute engine. Then, the compute engine runs the task, using the resources on the machine on which the compute engine is running.

The ability to perform arbitrary tasks is enabled by the dynamic nature of the Java platform, which is extended to the network by RMI. RMI dynamically loads the task code into the compute engine's Java virtual machine and runs the task without prior knowledge of the class that implements the task. Such an application, which has the ability to download code dynamically, is often called a *behavior-based application*. Such applications usually require full agent-enabled infrastructures. With RMI, such applications are part of the basic mechanisms for distributed computing on the Java platform.

**: Web service**

## Web Services

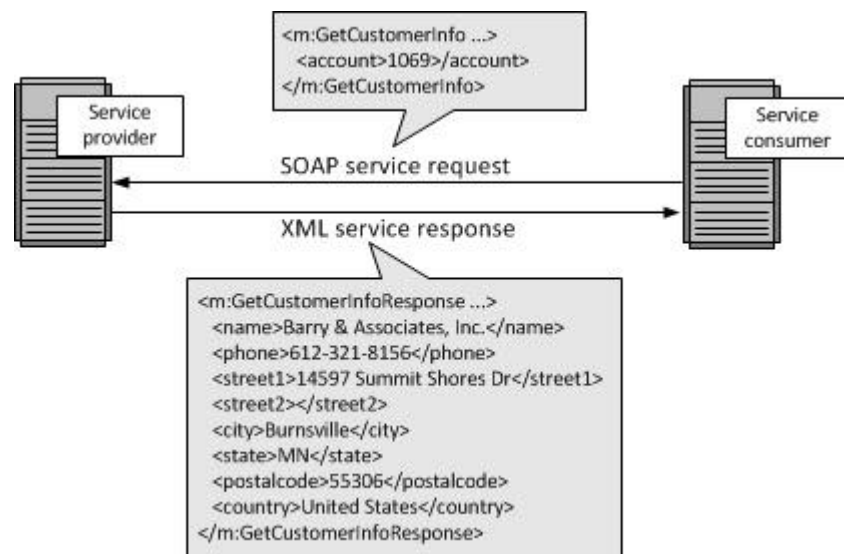
First, Web Services using SOAP, REST, and JSON are discussed. This is followed by a history of Web Services covering the Web Services Description Language (WSDL) and Universal Description, Discovery, and Integration (UDDI).

### Web Services Specifications

Three specifications for Web Services are illustrated in this section: SOAP, REST, and JSON.

## SOAP

SOAP was originally part of the specification that included the Web Services Description Language (WSDL) and Universal Description, Discovery, and Integration (UDDI). It is used now without WSDL and UDDI. Instead of the discovery process described in the History of the Web Services Specification section below, SOAP messages are hard-coded or generated without the use of a repository. The interaction is illustrated in the figure below. More on [SOAP](#).



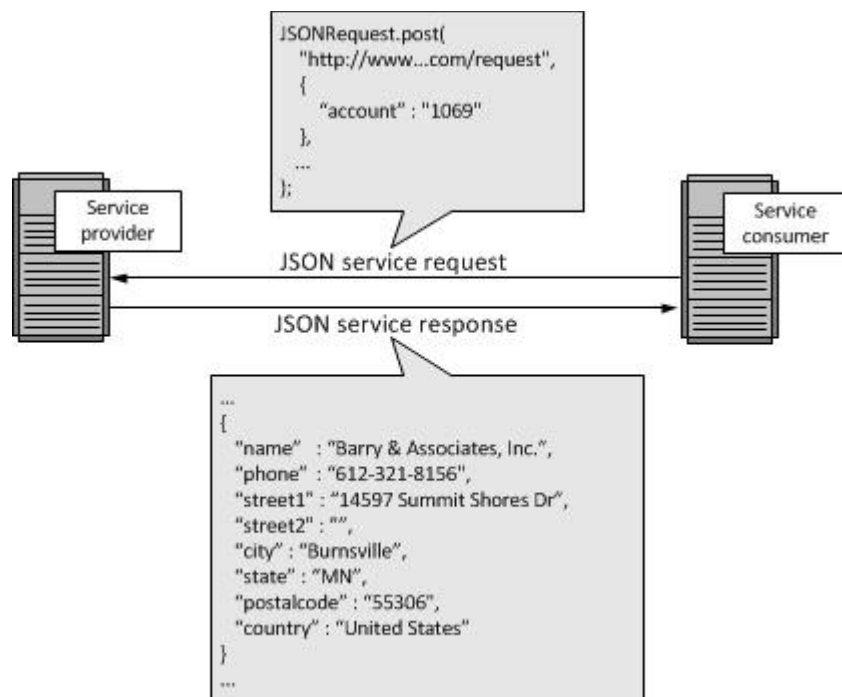
## Representation State Transfer (REST)

Representation State Transfer (REST) appeals to developers because it has a simpler style that makes it easier to use than SOAP. It also less verbose so that less volume is sent when communicating. The interaction is illustrated in the figure below. More on [REST](#).



## JavaScript Object Notation (JSON)

While both SOAP and REST use XML for interchange, JavaScript Object Notation (JSON) uses a subset of JavaScript. This is illustrated in the figure below. More on [JSON](#).



## When to Use SOAP, REST, JSON or Other Options

There really is no "best" option for Web Services. Generally, you will use whatever your service provider supports. If you use multiple service providers,

it is easily possible that you will be using all three Web Services specifications: SOAP, REST, and JSON.

## History of the Web Services Specification

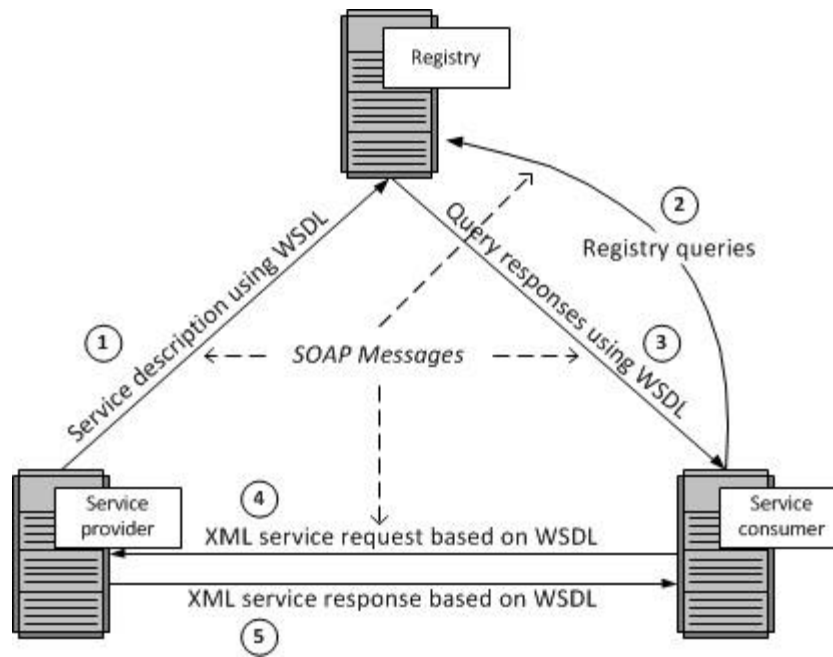
Web Services Description Language (WSDL); Universal Description and Discovery (UDDI); and SOAP formed the original Web Services specification. This section provides a history.

## Web Services Description Language (WSDL)

The Web Services Description Language (WSDL) forms the basis for the original Web Services specification. The following figure illustrates the use of WSDL. At the left is a service provider. At the right is a service consumer. The steps involved in providing and consuming a service are:

1. A service provider describes its service using WSDL. This definition is published to a repository of services. The repository could use Universal Description, Discovery, and Integration (UDDI). Other forms of directories could also be used.
2. A service consumer issues one or more queries to the repository to locate a service and determine how to communicate with that service.
3. Part of the WSDL provided by the service provider is passed to the service consumer. This tells the service consumer what the requests and responses are for the service provider.
4. The service consumer uses the WSDL to send a request to the service provider.
5. The service provider provides the expected response to the service consumer.

More on [Web Services Description Language](#).



## Universal Description, Discovery, and Integration (UDDI)

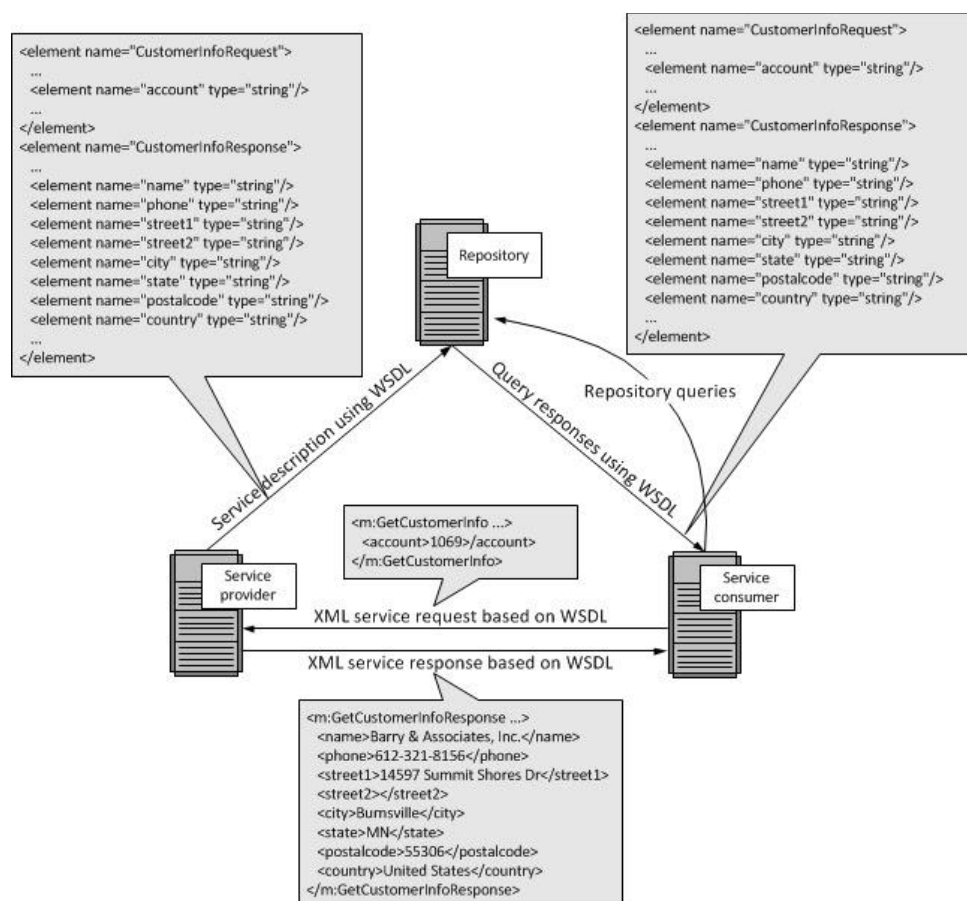
The repository shown in the above figure could be a UDDI registry. The UDDI registry was intended to eventually serve as a means of "discovering" Web Services described using WSDL. The idea is that the UDDI registry can be searched in various ways to obtain contact information and the Web Services available for various organizations. How much "discovery" was ever used is open to discussion. Nevertheless, even without the discovery portion, the UDDI registry is a way to keep up-to-date on the Web Services your organization currently uses. It can be used at design time and with governance. An alternative to UDDI is the [ebXML Registry](#). More on [Universal Description, Discovery, and Integration](#).

## SOAP

All the messages shown in the above figure are sent using SOAP. (SOAP at one time stood for Simple Object Access Protocol. Now, the letters in the acronym have no particular meaning .) SOAP essentially provides the envelope for sending the Web Services messages. SOAP generally uses HTTP , but other

means of connection may be used. HTTP is the familiar connection we all use for the Internet. In fact, it is the pervasiveness of HTTP connections that will help drive the adoption of Web Services. More on SOAP and [Messaging](#).

The next figure provides more detail on the messages sent using Web Services. At the left of the figure is a fragment of the WSDL sent to the repository. It shows a CustomerInfoRequest that requires the customer's account to object information. Also shown is the CustomerInfoResponse that provides a series of items on customer including name, phone, and address items.



At the right of this figure is a fragment of the WSDL being sent to the service consumer. This is the same fragment sent to the repository by the service provider. The service consumer uses this WSDL to create the service request shown above the arrow connecting the service consumer to the service provider. Upon receiving the request, the service provider returns a message using the



format described in the original WSDL. That message appears at the bottom of the figure.

## **(Java Database connectivity) JDBC**

# **JDBC Overview**

## **Contents**

- JDBC API Overview
- JDBC Architecture
- Partnering for Progress
- Industry Momentum
- Advantages of JDBC Technology
- Key Features
- Related Documents

The JDBC API is the industry standard for database-independent connectivity between the Java programming language and a wide range of databases. The JDBC API provides a call-level API for SQL-based database access. JDBC technology allows you to use the Java programming language to exploit "Write Once, Run Anywhere" capabilities for applications that require access to enterprise data.

## **JDBC API Overview**

The JDBC API makes it possible to do three things:

- Establish a connection with a database or access any tabular data source
- Send SQL statements
- Process the results

## **JDBC Architecture**

The JDBC API contains two major sets of interfaces: the first is the JDBC API for application writers, and the second is the lower-level JDBC driver API for driver writers. JDBC technology drivers fit into one of four categories.

Applications and applets can access databases via the JDBC API using pure Java JDBC technology-based drivers, as shown in this figure:

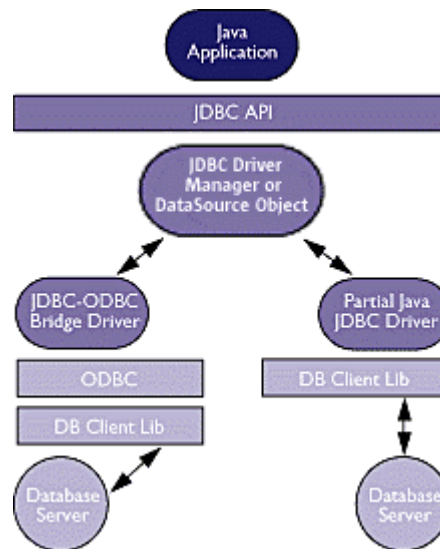
### *Left side, Type 4: Direct-to-Database Pure Java Driver*

This style of driver converts JDBC calls into the network protocol used directly by DBMSs, allowing a direct call from the client machine to the DBMS server and providing a practical solution for intranet access.

### *Right side, Type 3: Pure Java Driver for Database Middleware*

This style of driver translates JDBC calls into the middleware vendor's protocol, which is then translated to a DBMS protocol by a middleware server. The middleware provides connectivity to many different databases.

The graphic below illustrates JDBC connectivity using ODBC drivers and existing database client libraries.



*Left side, Type 1: JDBC-ODBC Bridge plus ODBC Driver*

This combination provides JDBC access via ODBC drivers. ODBC binary code -- and in many cases, database client code -- must be loaded on each client machine that uses a JDBC-ODBC Bridge. Sun provides a JDBC-ODBC Bridge driver, which is appropriate for experimental use and for situations in which no other driver is available.

*Right side, Type 2: A native API partly Java technology-enabled driver*

This type of driver converts JDBC calls into calls on the client API for Oracle, Sybase, Informix, DB2, or other DBMS. Note that, like the bridge driver, this style of driver requires that some binary code be loaded on each client machine.

For comparison of driver types, please see the article published in Computerworld.

## **Partnering for Progress**

Sun worked with an array of companies in the industry to create and rapidly establish the JDBC API as the industry-standard, open interface for Java applications to access databases.

## **Industry Momentum**

Leading database, middleware and tool vendors have been building support for JDBC technology into many new products. This ensures that customers can build portable Java applications while choosing from a wide range of competitive products for the solution best suited to their needs. See the [Industry Support](#) page for a list of companies that are shipping products with support for JDBC technology.

## **Advantages of JDBC Technology**

### *Leverage Existing Enterprise Data*

With JDBC technology, businesses are not locked in any proprietary architecture, and can continue to use their installed databases and access information easily -- even if it is stored on different database management systems.

### *Simplified Enterprise Development*

The combination of the Java API and the JDBC API makes application development easy and economical. JDBC hides the complexity of many data access tasks, doing most of the "heavy lifting" for the programmer behind the scenes. The JDBC API is simple to learn, easy to deploy, and inexpensive to maintain.

### *Zero Configuration for Network Computers*

With the JDBC API, no configuration is required on the client side. With a driver written in the Java programming language, all the information needed to make a connection is completely defined by the JDBC URL or by a DataSource object registered with a Java Naming and Directory Interface (JNDI) naming service. Zero configuration for clients supports the network computing paradigm and centralizes software maintenance.

## **Key Features**

### *Full Access to Metadata*

The JDBC API provides metadata access that enables the development of sophisticated applications that need to understand the underlying facilities and capabilities of a specific database connection.

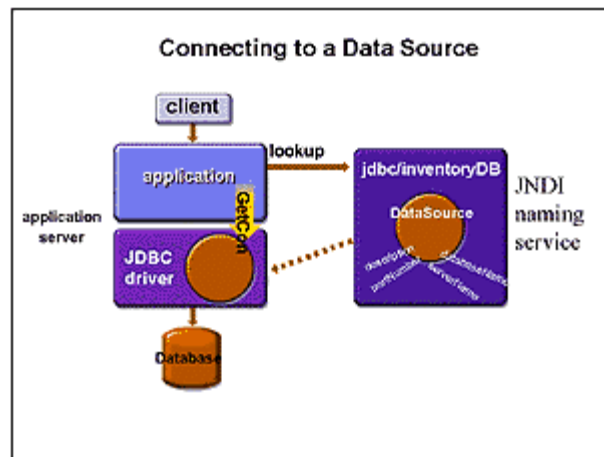
### *No Installation*

A pure JDBC technology-based driver does not require special installation; it is automatically downloaded as part of the applet that makes the JDBC calls.

### *Database Connection Identified by URL*

JDBC technology exploits the advantages of Internet-standard URLs to identify database connections. The JDBC API includes an even better way to identify and connect to a data source, using a DataSource object, that makes code even more portable and easier to maintain.

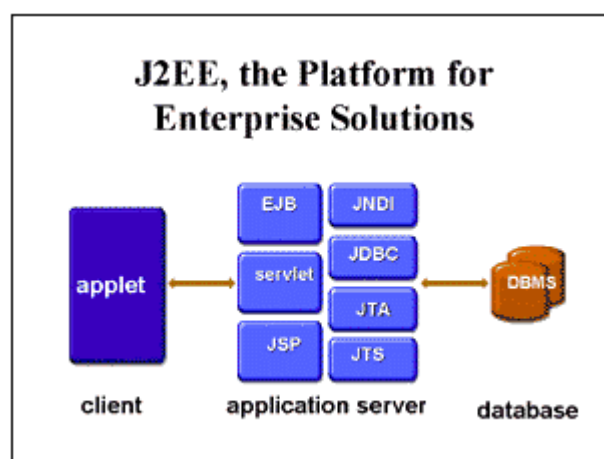
In addition to this important advantage, DataSource objects can provide connection pooling and distributed transactions, essential for enterprise database computing. This functionality is provided transparently to the programmer.



### *Included in the Java Platform*

As a core part of the Java 2 Platform, the JDBC API is available anywhere that the platform is. This means that your applications can truly write database applications once and access data anywhere. The JDBC API is included in both the Java 2 Platform, Standard Edition (J2SE) and the Java 2 Platform, Enterprise Edition (J2EE), providing server-side functionality for industrial strength scalability.

An example of a J2EE based architecture that includes a JDBC implementation:



### *Requirements*

- Software: The Java 2 Platform (either the Java 2 SDK, Standard Edition, or the Java 2 SDK, Enterprise Edition), an SQL database, and a JDBC technology-based driver for that database.
- Hardware: Same as for the Java 2 Platform.

## **What is NoSQL**

### **What is a NoSQL (Not Only SQL) Database?**

A NoSQL database environment is, simply put, a non-relational and largely distributed database system that enables rapid, ad-hoc organization and analysis of extremely high-volume, disparate data types. NoSQL databases are sometimes referred to as cloud databases, non-relational databases, Big Data databases and a myriad of other terms and were developed in response to the sheer volume of data being generated, stored and analyzed by modern users (user-generated data) and their applications (machine-generated data).

In general, NoSQL databases have become the first alternative to relational databases, with scalability, availability, and fault tolerance being key deciding factors. They go well beyond the more widely understood legacy, relational databases (such as Oracle, SQL Server and DB2 databases) in satisfying the needs of today's modern business applications. A very flexible and [schema-less data model](#), horizontal scalability, distributed architectures, and the use of languages and interfaces that are “not only” SQL typically characterize this technology.

From a business standpoint, considering a NoSQL or ‘Big Data’ environment has been shown to provide a clear competitive advantage in numerous industries. In the ‘age of data’, this is compelling information as a great saying

about the importance of data is summed up with the following “if your data isn’t growing then neither is your business”.



## Types of NoSQL Databases

There are four general types of NoSQL databases, each with their own specific attributes:

- Graph database – Based on [graph theory](#), these databases are designed for data whose relations are well represented as a graph and has elements which are interconnected, with an undetermined number of relations between them. Examples include: Neo4j and Titan.
- Key-Value store – we start with this type of database because these are some of the least complex NoSQL options. These databases are designed for storing data in a schema-less way. In a key-value store, all of the data within consists of an indexed key and a value, hence the name. Examples of this type of database include: [Cassandra](#), DyanmoDB, Azure Table Storage (ATS), Riak, BerkeleyDB.
- Column store – (also known as wide-column stores) instead of storing data in rows, these databases are designed for storing data tables as sections of columns of data, rather than as rows of data. While this simple description sounds like the inverse of a standard database, wide-column stores offer very high performance and a highly scalable architecture. Examples include: HBase, BigTable and HyperTable.
- Document database – expands on the basic idea of key-value stores where “documents” contain more complex in that they contain data and each document



is assigned a unique key, which is used to retrieve the document. These are designed for storing, retrieving, and managing document-oriented information, also known as semi-structured data. Examples include: MongoDB and CouchDB.

The following table lays out some of the key attributes that should be considered when evaluating NoSQL databases.

Datamodel	Performance	Scalability	Flexibility	Complexity	Functionality
Key-value store	High	High	High	None	Variable (None)
Column Store	High	High	Moderate	Low	Minimal
Document Store	High	Variable (High)	High	Low	Variable (Low)
Graph Database	Variable	Variable	High	High	Graph Theory

## Why NoSQL?

### Advantages of NoSQL Databases over Relational Databases

The reasons for businesses to adopt [a NoSQL database environment over a relational database](#) have almost everything to do with the following market drivers and technical requirements.

When making the switch, consider checking out this roadmap [relational database to NoSQL database](#) for a walkthrough of NoSQL education, migration and success.

## *The Growth of Big Data*

Big Data is one of the key forces driving the growth and popularity of NoSQL for business. The almost limitless array of data collection technologies ranging from simple online actions to point of sale systems to GPS tools to smartphones and tablets to sophisticated sensors – and many more – act as force multipliers for data growth.

In fact, one of the first reasons to use NoSQL is because you have a Big Data project to tackle. A [Big Data](#) project is normally typified by:

1. High data velocity – lots of data coming in very quickly, possibly from different locations.
2. Data variety – storage of data that is structured, semi-structured and unstructured.
3. Data volume – data that involves many terabytes or petabytes in size.
4. Data complexity – data that is stored and managed in different locations or data centers.



## *Continuous Data Availability*

In today's marketplace, where the competition is just a click away, downtime can be deadly to a company's bottom line and reputation. Hardware failures can and will occur, fortunately NoSQL database environments are built with a distributed architecture so there are no single points of failure and there is built-

in redundancy of both function and data. If one or more database servers, or ‘nodes’ goes down, the other nodes in the system are able to continue with operations without data loss, thereby showing true fault tolerance. In this way, NoSQL database environments are able to provide continuous availability whether in single locations, across data centers and in the cloud. When deployed appropriately, NoSQL databases can supply high performance at massive scale, which never go down. This is immensely beneficial as any system updates or modifications can be made without having to take the database offline. This fact alone draws the attention of businesses that are serving customers who expect availability of applications and where downtime equates to real dollars lost.

### ***Real Location Independence***

The term “location independence” means the ability to read and write to a database regardless of where that I/O operation physically occurs and to have any write functionality propagated out from that location, so that it’s available to users and machines at other sites. Such functionality is very difficult to architect for relational databases. Some techniques can be employed such as master/slave architectures and database sharding can sometimes meet the need for location independent read operations, but writing data everywhere is a different matter, especially when those data volumes are high. Other scenarios where location independence is an advantage are many and include servicing customers in many different geographies and needing to keep data local at those sites for fast access.

### ***Modern Transactional Capabilities***

The concept of transactions appears to be changing in the Internet age, and it’s been demonstrated that ACID transactions are no longer a requirement in database driven systems. At first blush, this assertion sounds extreme, as

transactional integrity is a characteristic of most every data system – especially those with information requirements that demand accuracy and safety. However, what this refers to is not the jeopardizing of data, but rather the new way modern applications ensure transactional consistency across widely distributed systems. The “C” in ACID refers to [data Consistency](#) in relational database management systems which is enforced via foreign keys/referential integrity constraints. This type of consistency is not utilized in progressive data management systems such as NoSQL databases because there are no JOIN operations, as this would require more rigid enforcement of consistency.

Instead, the “Consistency” that concerns NoSQL databases is found in the CAP theorem, which signifies the immediate or eventual consistency of data across all nodes that participate in a distributed database. The data is still safe and meets the AID portion of the RDBMS ACID definition, but its consistency is maintained differently given the nature and architecture of the system.

### ***Flexible Data Models***

One of the major reasons businesses move to a NoSQL database system from a relational database management system (RDBMS) is the more [flexible data model](#) that’s found in most NoSQL databases. The relational data model is based on defined relationships between tables, which themselves are defined by a determined column structure, all of which are explicitly organized in a database schema – all very strict and uniform. Problems begin to arise with the relational model around scalability and performance when trying to manage the large data volumes that are becoming a fact of life in a modern IT and business environment. A NoSQL data model – often referred to as schema-less – can support many of these use cases and others that don’t fit well into a RDBMS. A NoSQL database is able to accept all types of data – structured, semi-structured, and unstructured – much more easily than a relational database which rely on a predefined schema. This characteristic of a relational database

can be a hindrance on flexibility because a predefined schema rigidly determines how the database and database data are organized. Many of today's business applications actually have the ability to enforce rules on data usage themselves making a schema-less database platform a viable option.

Finally, performance factors come into play with an RDBMS' data model, especially where "wide rows" are involved and update actions are many, which can have real implications on performance. However, a NoSQL data model easily handles such situations and delivers very fast performance for both read and write operations.

### ***Better Architecture***

Another reason to use a NoSQL database is because you need a more suitable architecture for a particular application. It's critical that organizations adopt a NoSQL platform that allows them to keep their very high volume data in the context of their applications. Some, but not all, NoSQL solutions provide modern architectures that can tackle the type of applications that require high degrees of scale, data distribution, and continuous availability. Data center support, and as is more common, multiple data center support, should be a use case with which a NoSQL environment complies. It's not just what your big data needs look like today but also out to greater time horizons that decisions should be made.

### ***Analytics and Business Intelligence***

A key strategic driver of implementing a NoSQL database environment is the ability to mine the data that is being collected so as to derive insights that puts your business at a competitive advantage. Extracting meaningful business intelligence from very high volumes of data is a very difficult task to achieve with traditional relational database systems. Modern NoSQL database systems not only provide storage and management of business application data but also

deliver integrated data analytics that deliver instant understanding of complex data sets and facilitate flexible decision-making.

## **Which NoSQL database should you use?**

### **Choosing the Right NoSQL Database**

To this point we have been able to detail many [reasons for a business to adopt a NoSQL](#), or non-relational database platform, and also covered different types of NoSQL database options. Determining which NoSQL database to adopt is an exercise that requires a business to a long hard look at itself and its applications – which is always good. Understand clearly what your business goals are, what your application(s) needs – and challenges – are today, and what those needs may be on the horizon. Also, make sure that both the business and the technology goals are aligned so that the platform decision made delivers for all key stakeholders. Then work back to the NoSQL platform that will satisfy these needs and provide the most solid foundation for you to build on.

Key considerations when choosing your NoSQL platform include:

- **Workload diversity** – Big Data comes in all shapes, colors and sizes. Rigid schemas have no place here; instead you need a more flexible design. You want your technology to fit your data, not the other way around. And you want to be able to do more with all of that data – perform transactions in real-time, run analytics just as fast and find anything you want in an instant from oceans of data, no matter what from that data may take.
- **Scalability** – With big data you want to be able to scale very rapidly and elastically, whenever and wherever you want. This applies to all situations, whether scaling across multiple data centers and even to the cloud if needed.

- **Performance** – As has already been discussed, in an online world where nanosecond delays can cost you sales, Big Data must move at extremely high velocities no matter how much you scale or what workloads your database must perform. Performance of your environment, namely your applications, should be high on the list of requirements for deploying a NoSQL platform.
- **Continuous Availability** – Building off of the performance consideration, when you rely on big data to feed your essential, revenue-generating 24/7 business applications, even high availability is not high enough. Your data can never go down, therefore there should be no single point of failure in your NoSQL environment, thus ensuring applications are always available.
- **Manageability** – Operational complexity of a NoSQL platform should be kept at a minimum. Make sure that the administration and development required to both maintain and maximize the benefits of moving to a NoSQL environment are achievable.
- **Cost** – This is certainly a glaring reason for making the move to a NoSQL platform as meeting even one of the considerations presented here with relational database technology can cost become prohibitively expensive. Deploying NoSQL properly allows for all of the benefits above while also lowering operational costs.
- **Strong Community** – This is perhaps one of the more important factors to keep in mind as you move to a NoSQL platform. Make sure there is a solid and capable community around the technology, as this will provide an invaluable resource for the individuals and teams that will be managing the environment. Involvement on the part of the vendor should not only include strong support and technical resource availability, but also consistent outreach to the user base. Good local user groups and meetups will provide many opportunities for communicating with other individuals and teams that will provide great insight into how to work best with the platform of choice.

## Singleton Pattern

Singleton pattern is one of the simplest design patterns in Java. This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object.

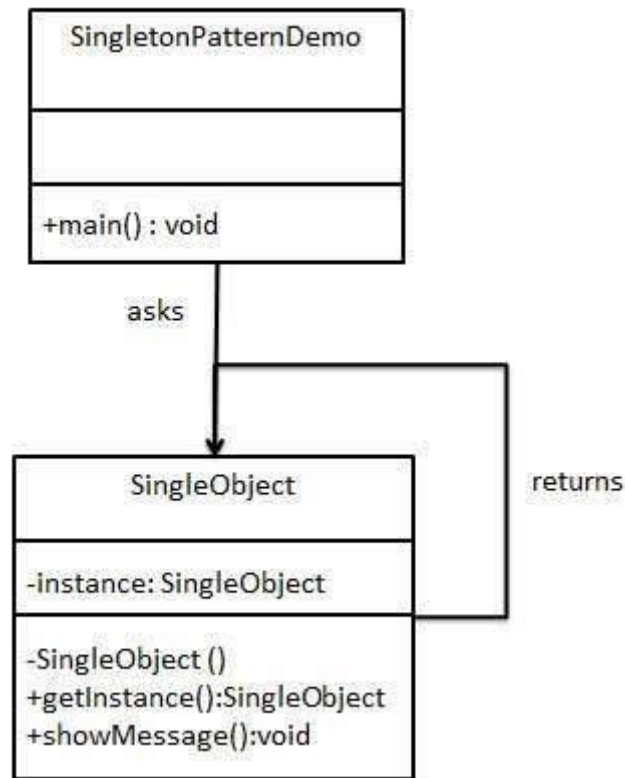
This pattern involves a single class which is responsible to create an object while making sure that only single object gets created. This class provides a way to access its only object which can be accessed directly without need to instantiate the object of the class.

### Implementation

We're going to create a *SingleObject* class. *SingleObject* class have its constructor as private and have a static instance of itself.

*SingleObject* class provides a static method to get its static instance to outside world. *SingletonPatternDemo*, our demo class will use *SingleObject* class to get a *SingleObject* object.





Step 1

Create a Singleton Class.

*SingletonObject.java*

```
Public class Single Object {

    //create an object of Single Object

    Private static Single Object instance = new Single Object();

    //make the constructor private so that this class cannot be
    //instantiated

    private SingleObject(){ }
```

```
//Get the only object available

public static SingleObject getInstance(){

    return instance;

}

public void showMessage(){

    System.out.println("Hello World!");

}

}
```

## Step 2

Get the only object from the singleton class.

*SingletonPatternDemo.java*

```
public class SingletonPatternDemo {

    public static void main(String[] args) {

        //illegal construct

        //Compile Time Error: The constructor SingleObject() is not visible

        //SingleObject object = new SingleObject();

        //Get the only object available

        SingleObject object = SingleObject.getInstance();

    }

}
```

```
//show the message  
object.showMessage();  
}  
}
```

Step 3

Verify the output.

Hello World!

## Factory Pattern

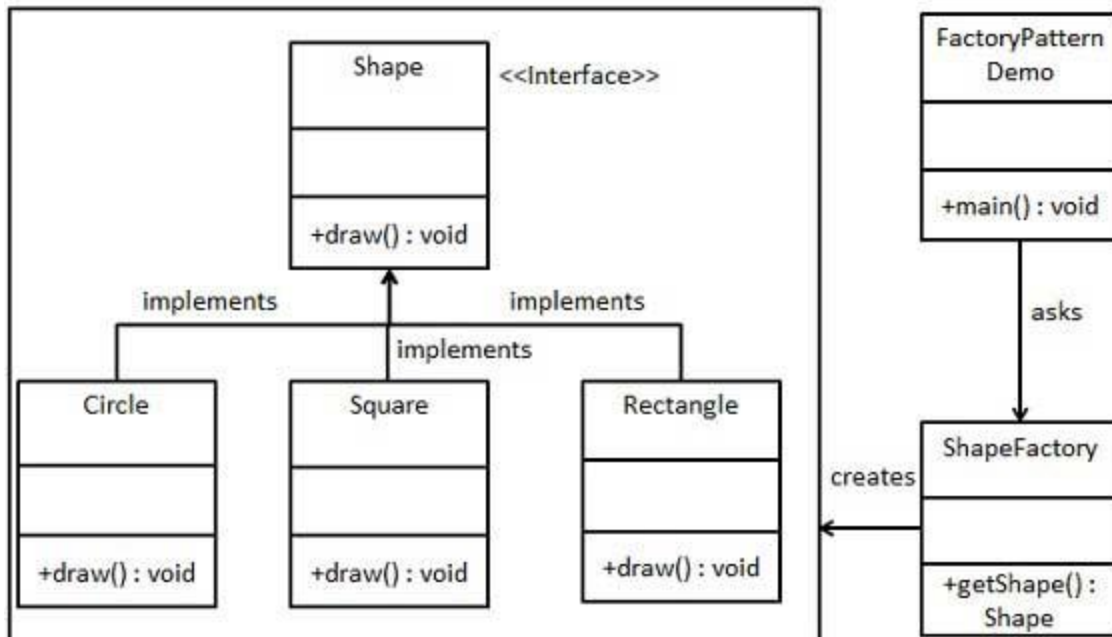
Factory pattern is one of most used design pattern in Java. This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object.

In Factory pattern, we create object without exposing the creation logic to the client and refer to newly created object using a common interface.

### Implementation

We're going to create a *Shape* interface and concrete classes implementing the *Shape* interface. A factory class *ShapeFactory* is defined as a next step.

*FactoryPatternDemo*, our demo class will use *ShapeFactory* to get a *Shape* object. It will pass information (*CIRCLE* / *RECTANGLE* / *SQUARE*) to *ShapeFactory* to get the type of object it needs.



Step 1

Create an interface.

*Shape.java*

```

public interface Shape {

    void draw();

}
  
```

Step 2

Create concrete classes implementing the same interface.

*Rectangle.java*

```

public class Rectangle implements Shape {

    @Override

    public void draw() {

    }

}
  
```

```
        System.out.println("Inside Rectangle::draw() method.");  
    }  
}
```

*Square.java*

```
public class Square implements Shape {  
  
    @Override  
    public void draw() {  
        System.out.println("Inside Square::draw() method.");  
    }  
}
```

*Circle.java*

```
public class Circle implements Shape {  
  
    @Override  
    public void draw() {  
        System.out.println("Inside Circle::draw() method.");  
    }  
}
```

Step 3

Create a Factory to generate object of concrete class based on given information.

*ShapeFactory.java*

```
public class ShapeFactory {  
  
    //use getShape method to get object of type shape  
  
    public Shape getShape(String shapeType){  
        if(shapeType == null){  
            return null;  
        }  
  
        if(shapeType.equalsIgnoreCase("CIRCLE")){  
            return new Circle();  
  
        } else if(shapeType.equalsIgnoreCase("RECTANGLE")){  
            return new Rectangle();  
  
        } else if(shapeType.equalsIgnoreCase("SQUARE")){  
            return new Square();  
        }  
  
        return null;  
    }  
}
```

```
}
```

#### Step 4

Use the Factory to get object of concrete class by passing an information such as type.

*FactoryPatternDemo.java*

```
public class FactoryPatternDemo {  
  
    public static void main(String[] args) {  
  
        ShapeFactory shapeFactory = new ShapeFactory();  
  
        //get an object of Circle and call its draw method.  
        Shape shape1 = shapeFactory.getShape("CIRCLE");  
  
        //call draw method of Circle  
        shape1.draw();  
  
        //get an object of Rectangle and call its draw method.  
        Shape shape2 = shapeFactory.getShape("RECTANGLE");  
  
        //call draw method of Rectangle  
        shape2.draw();  
    }  
}
```

```
//get an object of Square and call its draw method.  
  
Shape shape3 = shapeFactory.getShape("SQUARE");  
  
//call draw method of circle  
  
shape3.draw();  
  
}  
  
}
```

Step 5

Verify the output.

```
Inside Circle::draw() method.  
Inside Rectangle::draw() method.  
Inside Square::draw() method.
```

## Builder Pattern

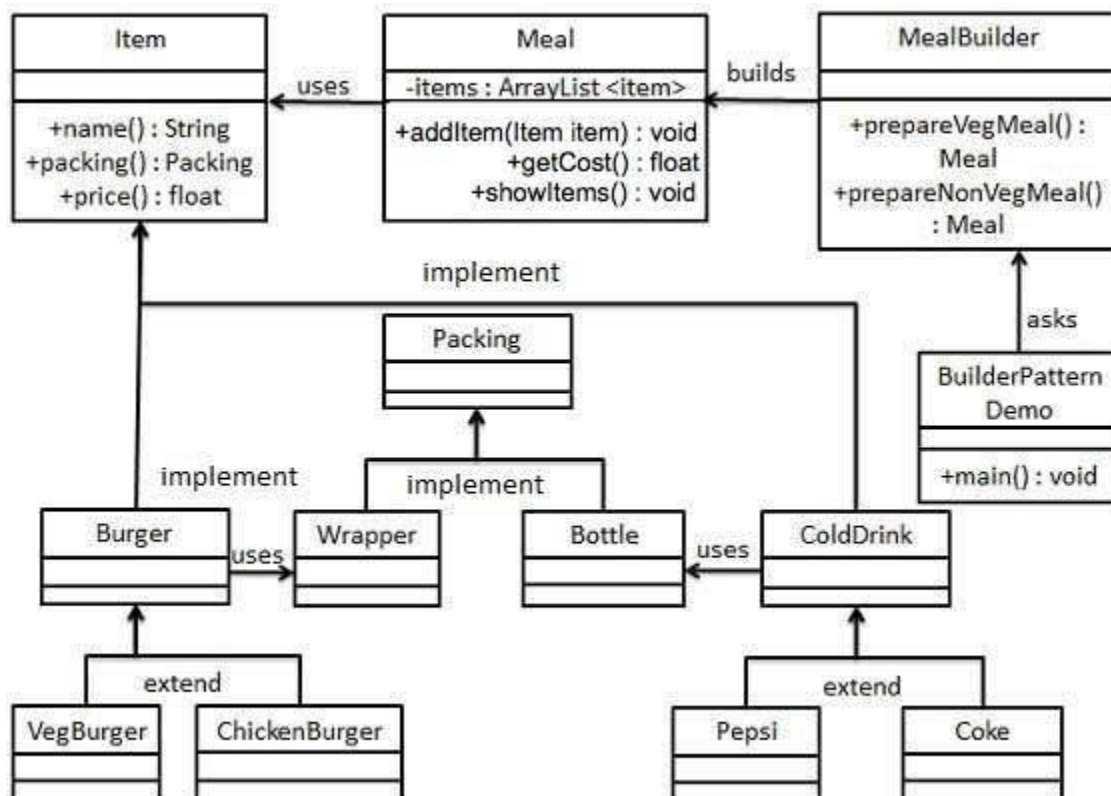
Builder pattern builds a complex object using simple objects and using a step by step approach. This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object.

A Builder class builds the final object step by step. This builder is independent of other objects.

### Implementation



We then create a *Meal* class having *ArrayList* of *Item* and a *MealBuilder* to build different types of *Meal* objects by combining *Item*. *BuilderPatternDemo*, our demo class will use *MealBuilder* to build a *Meal*.



Create an interface `Item` representing food item and packing.

*Item.java*

```
public interface Item {  
    public String name();  
    public Packing packing();  
    public float price();  
}
```

*Packing.java*

```
public interface Packing {  
    public String pack();  
}
```

Step 2

Create concrete classes implementing the Packing interface.

*Wrapper.java*

```
public class Wrapper implements Packing {  
  
    @Override  
    public String pack() {  
        return "Wrapper";  
    }  
}
```

*Bottle.java*

```
public class Bottle implements Packing {  
  
    @Override  
    public String pack() {  
        return "Bottle";  
    }  
}
```

### Step 3

Create abstract classes implementing the item interface providing default functionalities.

*Burger.java*

```
public abstract class Burger implements Item {  
  
    @Override  
    public Packing packing() {  
        return new Wrapper();  
    }  
  
    @Override  
    public abstract float price();  
}
```

*ColdDrink.java*

```
public abstract class ColdDrink implements Item {

    @Override

    public Packing packing() {

return new Bottle();

    }

    @Override

    public abstract float price();

}
```

Step 4

Create concrete classes extending Burger and ColdDrink classes

*VegBurger.java*

```
public class VegBurger extends Burger {

    @Override

    public float price() {

        return 25.0f;

    }

    @Override
```

```
public String name() {  
    return "Veg Burger";  
}  
}
```

*ChickenBurger.java*

```
public class ChickenBurger extends Burger {  
  
    @Override  
    public float price() {  
        return 50.5f;  
    }  
  
    @Override  
    public String name() {  
        return "Chicken Burger";  
    }  
}
```

*Coke.java*

```
public class Coke extends ColdDrink {  
  
    @Override
```

```
public float price() {  
    return 30.0f;  
}  
  
@Override  
public String name() {  
    return "Coke";  
}  
}
```

*Pepsi.java*

```
public class Pepsi extends ColdDrink {  
  
    @Override  
    public float price() {  
        return 35.0f;  
    }  
  
    @Override  
    public String name() {  
        return "Pepsi";  
    }  
}
```

```
}
```

### Step 5

Create a Meal class having Item objects defined above.

*Meal.java*

```
import java.util.ArrayList;
import java.util.List;

public class Meal {
    private List<Item> items = new ArrayList<Item>();

    public void addItem(Item item){
        items.add(item);
    }

    public float getCost(){
        float cost = 0.0f;

        for (Item item : items) {
            cost += item.price();
        }

        return cost;
    }
}
```

```

    }

    public void showItems(){

        for (Item item : items) {

            System.out.print("Item : " + item.name());

            System.out.print(", Packing : " + item.packing().pack());

            System.out.println(", Price : " + item.price());

        }

    }

}

```

#### Step 6

Create a MealBuilder class, the actual builder class responsible to create Meal objects.

*MealBuilder.java*

```

public class MealBuilder {

    public Meal prepareVegMeal (){

        Meal meal = new Meal();

        meal.addItem(new VegBurger());

        meal.addItem(new Coke());

        return meal;
    }
}

```



```

    }

    public Meal prepareNonVegMeal (){

        Meal meal = new Meal();

        meal.addItem(new ChickenBurger());

        meal.addItem(new Pepsi());

        return meal;

    }

}

```

#### Step 7

BuilderPatternDemo uses MealBuilder to demonstrate builder pattern.

*BuilderPatternDemo.java*

```

public class BuilderPatternDemo {

    public static void main(String[] args) {

        MealBuilder mealBuilder = new MealBuilder();

        Meal vegMeal = mealBuilder.prepareVegMeal();

        System.out.println("Veg Meal");

        vegMeal.showItems();

        System.out.println("Total Cost: " + vegMeal.getCost());
    }
}

```

```
Meal nonVegMeal = mealBuilder.prepareNonVegMeal();

System.out.println("\n\nNon-Veg Meal");

nonVegMeal.showItems();

System.out.println("Total Cost: " + nonVegMeal.getCost());

}

}
```

Step 8

Verify the output.

Veg Meal

Item : Veg Burger, Packing : Wrapper, Price : 25.0

Item : Coke, Packing : Bottle, Price : 30.0

Total Cost: 55.0

Non-Veg Meal

Item : Chicken Burger, Packing : Wrapper, Price : 50.5

Item : Pepsi, Packing : Bottle, Price : 35.0

Total Cost: 85.5

مدل رابطه ای

مدل رابطه ای متداول ترین مدل داده است که داده ها و ارتباطات بین آنها را به صورت مجموعه ای از جداول نمایش می دهد.

### تبدیل نمودار ER به مدل رابطه ای

مدل رابطه ای (relational model) در سال ۱۹۷۰ توسط ریاضیدانی به نام Edgar.F.Codd طراحی شد. مدل داده پیشنهادی یک مدل منطقی بر مبنای ریاضیات است که از منطق گزاره ها و تئوری مجموعه ها به عنوان زیربنا استفاده شده است.

یک پایگاه داده رابطه ای (relational database) پایگاه داده ای است که با مدل رابطه ای مطابقت داشته باشد و به صورت مجموعه ای از جدول هائی که از دید کاربر قابل درک هستند دیده می شود.

یک سیستم مدیریت پایگاه داده رابطه ای (RDBMS) سیستمی است که داده را طبق مدل رابطه ای مدیریت می کند.

RDBMS ها معمول ترین نوع سیستم های مدیریتی پایگاه داده امروزی هستند (نظیر Microsoft SQL Server, Microsoft Access, Oracle, MySQL, Sybase, DB ۲ و Informix).

اکثر RDBMS ها SQL را به عنوان زبان پرس و جوی خود بکار می برند.

### اصطلاحات

#### جدول (رابطه)

پایگاه داده رابطه ای مجموعه ای از آرایه های دو بعدی است که جداول (table) یا رابطه (relation) نامیده می شوند. جدول مهمترین ساختار داده ای در سیستم پایگاه داده رابطه ای است.

هر جدول (یا رابطه) داده‌ها را به صورت سطرها و ستون‌ها شکل می‌دهد. هر سطر شامل یک نمونه منحصر بفرد داده و مربوط به یک نمونه موجودیت است. ستون‌ها صفات خاصه آن موجودیت را بیان می‌کنند.

ترتیب سطرها و ستون‌ها در جداول اهمیت ندارد.

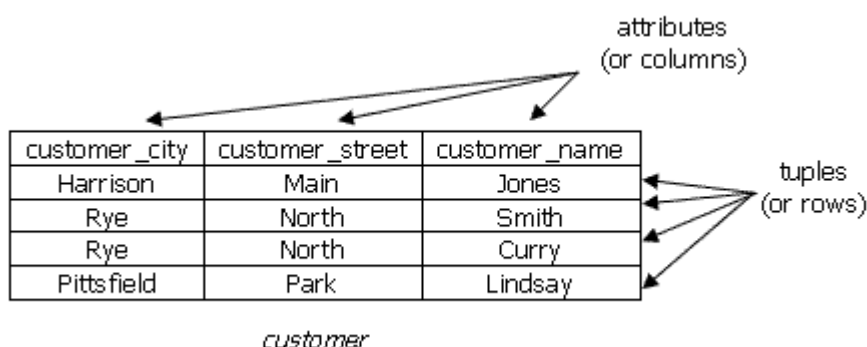
تعداد ستون‌های هر جدول را درجه (Degree) و تعداد سطرهای آن را کاردینالیتی (Cardinality) می‌نامند.

### تاپل (رکورد)

یک سطر از یک رابطه را یک تاپل (tuple) می‌نامند. هر تاپل در جدول نمایانگر یک نمونه از یک موجودیت است که رکورد هم گفته می‌شود.

تاپل‌ها ممکن است روی یکی از صفات خاصه به طور مرتب ذخیره شوند. ولی به طور کلی لازم نیست مرتب باشند.

مثال. رابطه Customer را در نظر بگیرید.



## فیلد (صفت خاصه)

هر ستون در جدول نشان دهنده یک صفت خاصه از یک نوع موجودیت است که فیلد (field) نامیده می شود. در هر فیلد نوع خاصی از داده ها ذخیره می شود.

مقادیر هر سطر باید با فیلدهای جدول نظیر به نظیر باشد به عبارت دیگر ترتیب مقادیر فیلدها در همه رکوردها باید یکسان است. ولی ترتیب ستون ها اهمیت ندارد.

## دامنه

مجموعه ای از مقادیر مجاز یک ستون دامنه (domain) نام دارد. معمولاً دامنه یک ستون از یک نوع داده است.

محصولات RDBMS مستقیماً دامنه را حمایت نمی کنند بلکه مجموعه ای از انواع داده عددی، متن، تاریخ و غیره را دارند که نحوه ذخیره سازی داده ها را مشخص می کنند. تاثیر دامنه را بیشتر می توان روی قیدها و مکانیسم های ورود داده مشاهده کرد.

مقادیر صفات خاصه معمولاً باید اتمیک باشند یعنی غیرقابل تفکیک باشند.

مقدار خاص null عضوی از هر دامنه است.

## NULL

null یا <null> یک علامت خاص است که تهی بودن فیلدی را نشان می دهد، برای زمانی که مقداری برای فیلد وجود ندارد یا آنرا نمی دانیم استفاده می شود (برای مثال مشتری تلفن ندارد یا شماره آنرا نداریم).

اکثر اوقات باید مشخص کنیم که آیا یک فیلد می تواند تهی باشد یا خیر. سعی در درج null در فیلدی که مجاز نیست ایجاد مشکل می کند.

دانستن چگونگی برخورد RDBMS با null اهمیت دارد زیرا مقادیر تهی نمی توانند در عملیات داده ای شرکت کنند.

بعضی از RDBMS ها null را اصلاً ندارند. راه حل آنها تعریف یک ستون اضافی برای ستونی است که می تواند تهی باشد. این ستون معین می کند آیا ستون مرتبط به آن دارای مقدار معتبر است یا خیر.

یک راه دیگر پر کردن فیلد با یک مقدار پیش فرض است. اگر مقدار پیش فرض قابل مشاهده ای وجود ندارد یا مشکل را حل نمی کند از null می توان استفاده کرد ولی باید مطمئن شد که در عملیات مسئله ساز نمی شود.

### پایگاه داده

یک پایگاه داده شامل چند جدول است. هر جدول بخشی از داده های سازمان را نمایش می دهد. ذخیره کلیه اطلاعات در یک جدول باعث تکرار داده یا نیاز به مقدار null می شود.

مثال. رابطه های محصول، مشتری و فروش را در یک سیستم فروش به صورت زیر تعریف می شوند:

PRODUCT				
ProductNo	Description	ReorderLevel	Price	QtyInStock
AQX88916	Flush widget	1000	23.95	4937
AAD62726	Left-handed doodad	0	4.95	10673
FGE91822	Grunge nut	5000	0.50	155016
AHL46785	Flange bracket	25000	1.35	51745
DHU69863	Self-righting socket	5	2.37	52

CUSTOMER				
CustomerNo	First	Last	Address	CreditLimit
4649-4673	Richard	Johnston	14 West Avenue	1000
1166-3461	Amelia	Waverley	995 Forth Street	<null>
7671-3496	Antonio	Gonzales	55B Granary Lane	<null>
6794-1674	Diane	Adams	364 East Road	1500
1113-7741	Wayne	Jones	42 York Street	<null>

SALE						
SaleNo	SaleDate	CustomerNo	ProductNo	Qty	Amount	Salesrep
12345	Aug 12 2002	4649-4673	AQX88916	1	23.95	Dave Williams
12346	Aug 12 2002	1113-7741	AQX88916	7	167.65	Sara Thompson
12347	Aug 13 2002	1166-3461	AHL46785	3705	5001.75	Li Qing
12348	Aug 13 2002	<null>	DHU69863	50	118.50	Sara Thompson

به جای رسم جدول به صورت فوق می توان رابطه ها را به شکل زیر تعریف کرد:

**PRODUCT(ProductNo, Description, ReorderLevel, Price, QtyInStock)**

**CUSTOMER(CustomerNo, First, Last, Address, CreditLimit)**

**SALE(SaleNo, SaleDate, CustomerNo, ProductNo, Qty, Amount, Salesrep)**

### کلید

در مدل رابطه ای هیچ دو سطری در جدول نباید مشابه باشند. این در واقع یک ویژگی اساسی جدول است. اگر دو سطر دو نمونه موجودیت متفاوت را در دنیای واقعی نشان دهند به نحوی باید از هم متمایز شوند تا به هر کدام در جدول بتوان جداگانه رجوع کرد. بنابراین حداقل یک مقدار منحصر به فرد باید وجود داشته باشد که باعث متمایز شدن سطرها از یکدیگر شود. ستونی که حاوی این مقدار است کلید نامیده می شود.

کلید داری دو خاصیت را باید دارا باشد؛ منحصر به فرد بودن و غیر تهی بودن (قانون اول جامعیت).

در یک رابطه انواع مختلفی از کلید ممکن است وجود داشته باشد:

- کلید کاندید
- کلید ترکیبی
- کلید اصلی
- کلید خارجی

#### کلید کاندید

از مجموعه صفات خاصه یک رابطه آنهایی که دارای دو ویژگی زیر هستند به عنوان کلید کاندید ( candid key) در رابطه مذکور مطرح می شوند:

- منحصر به فرد و غیر تهی بودن
- غیر قابل کاهش بودن، یعنی هیچ زیر مجموعه مناسبی از صفات خاصه تشکیل دهنده کلید، دارای خاصیت منحصر به فرد بودن نباشد.

مثال. شماره دانشجویی و کد ملی کلیدهای کاندید در جدول مشخصات دانشجو در دانشگاه می توانند باشند.

#### کلید ترکیبی

کلید ترکیبی (compound key) کلیدی است که از ترکیب چند صفت خاصه ساخته می شود.

مثال. در رابطه دانشجو مجموعه نام و شماره شناسنامه می توانند به عنوان کلید ترکیبی در نظر گرفته شوند.



## کلید اصلی

کلید اصلی (primary key)، کلید کاندیدی است که توسط طراح پایگاه داده انتخاب و معرفی می شود.

به عبارتی طراح بانک، یکی از کلیدهای کاندید را به عنوان کلید اصلی بر میگزیند.

در تعیین کلید اصلی از بین کلیدهای کاندید باید دو ضابطه زیر را در نظر گرفت:

- اهمیت کلید اصلی نسبت به سایر کلیدهای کاندید در پرس و جوها

- کوتاهتر بودن طول کلید کاندید از نظر تعداد بایت

نکته. هر جدول تنها یک کلید اولیه دارد اما به این معنی نیست که تنها یک شناسه منحصر به فرد دارد.

نکته. کلید می تواند صفات طبیعی موجودیت انتخاب شود، ولی اگر هیچ کدام از صفات خاصه مناسب

نبودند یک کلید جانشین نسبت داده شود (مانند شماره کارمندی برای جدول کارمند).

نکته. در جدول، زیر کلید اولیه یک خط کشیده می شود.

نکته. اگرچه در مدل رابطه ای کلیه جداول باید دارای کلید اولیه باشند، ولی تعدادی از RDBMS ها

اجباری در تعیین کلید برای هر رابطه نمی کنند، در اینصورت ترکیب کلیه صفات خاصه به عنوان کلید

در نظر گرفته می شود.

مثال. شماره دانشجویی در جدول مشخصات دانشجو را می توان به عنوان کلید اصلی انتخاب کرد.

## کلید خارجی

کلید خارجی (foreign key) صفت خاصه ای در یک جدول است که در جدول دیگر نقش کلید اصلی

یا کاندید را بازی کند.

کلید خارجی ارتباط بین دو جدول را برقرار می کند.

کلید خارجی بر خلاف کلید اصلی می تواند تکراری یا null باشد و ممکن است در یک جدول بیشتر از یک کلید خارجی وجود داشته باشد.

جدول شامل کلید خارجی را گاهی جدول فرزند و جدولی که به آن ارجاع دارد را جدول والد می نامند.

مثال. شماره مشتری در جدول SALE کلید خارجی است زیرا در جدول CUSTOMER کلید اصلی است. شماره مشتری که در جدول فروش بدست می آید در جدول مشتری جستجو می شود تا تعیین شود محصول به کدام مشتری فروخته شده است.

### خواص مدل

مدل رابطه ای دارای ویژگی های زیر است:

- متداول ترین مدل است
- بر اساس تئوری ریاضی است
- داده ها و ارتباطات بین آنها در پایگاه داده به صورت مجموعه ای از جداول دیده می شود
- هیچ جدولی دارای سطرهای تکراری نیست
- ترتیب سطرها و ستون ها در هر جدول مهم نیست
- ستون ها اتمیک هستند یعنی مقادیر ستون ها غیر قابل تجزیه اند
- هر مقدار که در دو رکورد مختلف واقع می شود رابطه ای را بین دو آن رکورد می فهماند
- ارتباط رابطه ها با یکدیگر از طریق صفات خاصه مشترک انجام می گیرد
- ایجاد، دسترسی و توسعه آن آسان است. بعد از ایجاد پایگاه داده اولیه، جداول جدید می توانند اضافه

نشوند بدون اینکه نیاز به تغییر کاربردهای موجود باشد

- مدل دید کاربر است نه روشی که داده بطور داخلی سازماندهی می شود

## نرمالسازی

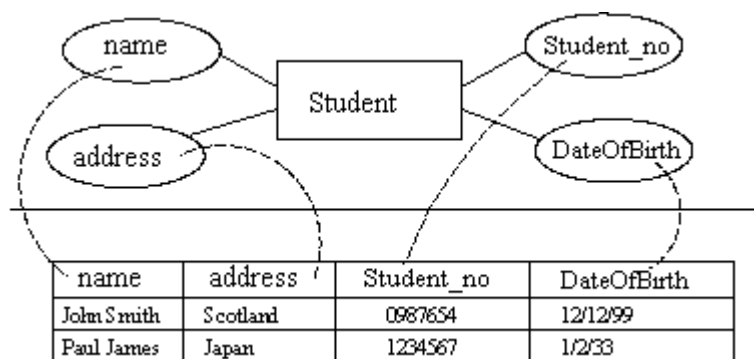
نرمالسازی (normalization) با نحوه تقسیم جداول در پایگاه داده رابطه ای سروکار دارد. نرمالسازی فرآیند تغییر ساختار پایگاه داده به منظور اجرای بهتر و راحتی کارکردن با داده است. فرم های مختلف نرمالسازی که روی پایگاه داده اعمال می شوند را فرم های نرمال می نامند.

## تبدیل نمودار ER به مدل رابطه ای

هر موجودیت در مدل ER به یک جدول در مدل رابطه ای تبدیل می شود. صفات خاصه موجودیت ستون های جدول و هر نمونه موجودیت سطرهای آنرا را می سازند.

قبل از اینکه فرآیند تبدیل انجام شود باید مطمئن شویم که مدل ER تا حد ممکن ساده شده است.

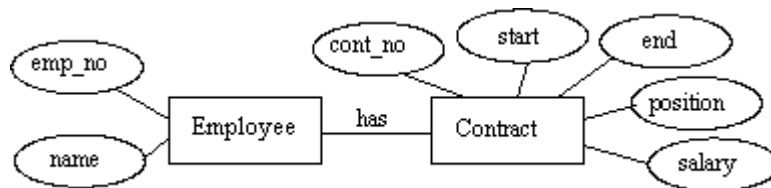
مثال. موجودیت دانشجو را در نظر بگیرید.



student(student\_no, name, address, date\_of\_birth)

برای تبدیل ارتباط یک به یک به رابطه، با توجه به اختیاری بودن یا اجباری بودن ارتباط، موجودیت ها یا ترکیب می شوند یا کلید اصلی یک موجودیت به عنوان کلید خارجی در دیگری قرار می گیرد.

مثال. ارتباط یک به یک بین موجودیت های کارمند و قرارداد را در نظر بگیرید. هر کارمند یک قرارداد دارد و هر قرارداد مربوط به یک کارمند است.



به رابطه زیر تبدیل می شود:

Employee(emp\_no, name, cont\_no, start, end, position, salary)

یا می تواند به صورت دو رابطه زیر تبدیل شود:

Employee (emp\_no, name, contract\_no)

Contract(cont\_no, start, end, position, salary)

یا

Employee (emp\_no, name)

Contract(cont\_no, start, end, position, salary, emp\_no)

در تبدیل ارتباط ها یک به چند کلید اصلی موجودیت سمت "یک" به عنوان کلید خارجی در سمت دیگر قرار می گیرد.

مثال. ارتباط یک به چند دانشجو و رشته تحصیلی را در نظر بگیرید:

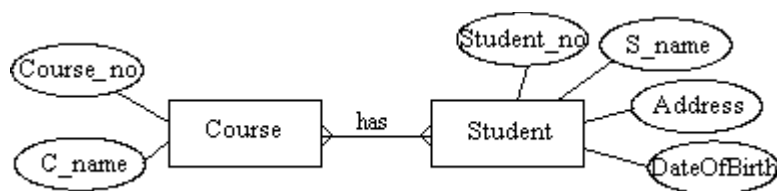
که به رابطه های زیر تبدیل می شود:

Student(student\_no, s\_name, address, DateOfBirth, module\_no)

Module(module\_no, m\_name)

در ارتباط چند به چند یک رابطه جدید با کلید های اولیه هر دو موجودیت ساخته می شود.

مثال. ارتباط دانشجو و درس را در نظر بگیرید.



که به رابطه های زیر تبدیل می شود:

Student(student\_no, s\_name, Address, DateOfBirth)

Course(Course\_no, C\_name)

Study(student\_no, Course\_no)

فرم های مربوط به نرم افزار

تاریخ : / /	باسمه تعالی	توصیه ها :
<div></div>		

## باسمه تعالی

### ● مشخصات کاربری :

نام کاربر :  بخش :  تاریخ اولین بررسی :   
 تاریخ بررسی :  شماره احوال :  شماره اتاق :

### ● مشخصات سیستم :

شماره هاردیورد :  شماره هارد :   
 تعداد هارد :  ظرفیت هارد :  رم :   
 گرافیک :  پردازنده :  سیستم عامل :   
 نام سیستم :  تعداد کاربرها :

### ● ملحقات سیستم :

کارت شبکه : ☐ کارت تصویر (وی جی ای) : ☐  
 کارت صدا : ☐ کارت تصویر (دی وی ای) : ☐  
 کارت یو اس بی : ☐

### ● توضیحات و توصیه ها:

## باسمه تعالی

### ● مشخصات کاربری :

نام کاربر :  بخش :  تاریخ اولین بررسی :   
تاریخ بررسی :  شماره اموال :  شماره اتاق :

### ● مشخصات ماینیور/اسکتر/پرینتر :

مدل :  نام شرکت سازنده :   
نوع :  اسکتر : ☐ پرینتر : ☐ ماینیور : ☐

### ● توضیحات و توصیه ها :



کد های پیاده سازی شده

بخش سرور قسمت model

```
package managment.model.TO;

import java.io.Serializable;

/**
 * Created by saeedtavana on 6/5/16.
 */
public class Computer implements Serializable ,device{

    public int getNumberOfdisk() {
        return numberOfdisk;
    }

    public void setNumberOfdisk(int numberOfdisk) {
        this.numberOfdisk = numberOfdisk;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getUser() {
        return uuser;
    }

    public void setUser(String uuser) {
        this.uuser = uuser;
    }

    public int getRoomNumber() {
        return roomNumber;
    }

    public void setRoomNumber(int roomNumber) {
        this.roomNumber = roomNumber;
    }

    public String getSection() {
        return section;
    }

    public void setSection(String section) {
        this.section = section;
    }

    public String getMotherboardnumber() {
        return motherboardnumber;
    }

    public void setMotherboardnumber(String motherboardnumber) {
        this.motherboardnumber = motherboardnumber;
    }

    public String getDisknumber() {
        return disknumber;
    }
}
```

```

public void setDisknumber(String disknumber) {
    this.disknumber = disknumber;
}

public int getCapacityOfDiskinG() {
    return capacityOfDiskinG;
}

public void setCapacityOfDiskinG(int capacityOfDiskinG) {
    this.capacityOfDiskinG = capacityOfDiskinG;
}

public int getRamInG() {
    return RamInG;
}

public void setRamInG(int ramInG) {
    RamInG = ramInG;
}

public String getCpu() {
    return cpu;
}

public void setCpu(String cpu) {
    this.cpu = cpu;
}

public String getOs() {
    return Os;
}

public void setOs(String os) {
    Os = os;
}

public int getNumberOfUser() {
    return numberOfUser;
}

public void setNumberOfUser(int numberOfUser) {
    this.numberOfUser = numberOfUser;
}

public String getComments() {
    return Comments;
}

public void setComments(String comments) {
    Comments = comments;
}

private int id;
private String uuser;
private int roomNumber;
private String section;
private String motherboardnumber;
private String disknumber;
private int numberOfdisk;
private int capacityOfDiskinG;
private int RamInG;
private String cpu;
private String Os;
private int numberOfUser;
private String Comments;

public String getDate() {
    return date;
}

```

```

    }

    public void setDate(String date) {
        this.date = date;
    }

    private String date;
}

```

```

package managment.model.TO;

import java.io.Serializable;

/**
 * Created by saeedtavana on 7/1/16.
 */
public class consol implements Serializable,device {
    public String getDate() {
        return date;
    }

    public void setDate(String date) {
        this.date = date;
    }

    public String getComments() {
        return comments;
    }

    public void setComments(String comments) {
        this.comments = comments;
    }

    private String date;
    private String comments;
}

```

```

package managment.model.TO;

/**
 * Created by saeedtavana on 7/1/16.
 */
public interface device {
}

```

```

package managment.model.TO;

import java.io.Serializable;
import java.util.Scanner;

/**
 * Created by saeedtavana on 7/3/16.
 */
public class equipmentMainteneace implements device,Serializable {

```

```

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getTetx() {
        return tetx;
    }

    public void setTetx(String tetx) {
        this.tetx = tetx;
    }


    public String getDate() {
        return date;
    }

    public void setDate(String date) {
        this.date = date;
    }

    private String date;
    private int id;
    private String tetx;
}

```

```

package managment.model.TO;

import java.io.Serializable;

/**
 * Created by saeedtavana on 6/5/16.
 */
public class Monitor implements Serializable,device {
    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getUser() {
        return uuser;
    }

    public void setUser(String uuser) {
        this.uuser = uuser;
    }

    public int getRoomNumber() {
        return roomNumber;
    }

    public void setRoomNumber(int roomNumber) {
        this.roomNumber = roomNumber;
    }

    public String getSection() {
        return section;
    }

```

```

    }

    public void setSection(String section) {
        this.section = section;
    }

    public String getDate() {
        return date;
    }

    public void setDate(String date) {
        this.date = date;
    }

    public String getModel() {
        return model;
    }

    public void setModel(String model) {
        this.model = model;
    }

    public String getBrand() {
        return brand;
    }

    public void setBrand(String brand) {
        this.brand = brand;
    }

    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }

    public String getCommand() {
        return command;
    }

    public void setCommand(String command) {
        this.command = command;
    }

    private int id;
    private String user;
    private int roomNumber;
    private String section;
    private String date;
    private String model;
    private String brand;
    private String type;
    private String command;
}

```

```

package managment.model.TO;

import java.io.Serializable;

/**
 * Created by saeedtavana on 6/5/16.
 */
public class NetworkUtility implements Serializable,device {
    public String getBrand() {

```

```

        return brand;
    }

    public void setBrand(String brand) {
        this.brand = brand;
    }

    public String getModel() {
        return model;
    }

    public void setModel(String model) {
        this.model = model;
    }

    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }

    public String getDate() {
        return date;
    }

    public void setDate(String date) {
        this.date = date;
    }

    public String getComments() {
        return comments;
    }

    public void setComments(String comments) {
        this.comments = comments;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public int id;
    private String brand;
    private String model;
    private String type;
    private String date;
    private String comments;
}

```

```

package managment.model.TO;

import java.io.Serializable;

/**
 * Created by saeedtavana on 6/5/16.
 */
public class Printer implements Serializable, device {
    public int getId() {
        return id;
    }
}

```

```

public void setId(int id) {
    this.id = id;
}

public String getUser() {
    return uuser;
}

public void setUser(String uuser) {
    this.uuser = uuser;
}

public int getRoomNumber() {
    return roomNumber;
}

public void setRoomNumber(int roomNumber) {
    this.roomNumber = roomNumber;
}

public String getSection() {
    return section;
}

public void setSection(String section) {
    this.section = section;
}

public String getDate() {
    return date;
}

public void setDate(String date) {
    this.date = date;
}

public String getModel() {
    return model;
}

public void setModel(String model) {
    this.model = model;
}

public String getBrand() {
    return brand;
}

public void setBrand(String brand) {
    this.brand = brand;
}

public String getType() {
    return type;
}

public void setType(String type) {
    this.type = type;
}

public String getCommand() {
    return command;
}

public void setCommand(String command) {
    this.command = command;
}

```

```

    private int id;
    private String uuser;
    private int roomNumber;
    private String section;
    private String date;
    private String model;
    private String brand;
    private String type;
    private String command;
}

```

```

package managment.model.TO;

import java.io.Serializable;

/**
 * Created by saeedtavana on 7/3/16.
 */
public class room implements device ,Serializable {
    public int getNumber() {
        return number;
    }

    public void setNumber(int number) {
        this.number = number;
    }

    public int getTelephone() {
        return telephone;
    }

    public void setTelephone(int telephone) {
        this.telephone = telephone;
    }

    private int number;
    private int telephone;
}

```

```

package managment.model.TO;

import java.io.Serializable;

/**
 * Created by saeedtavana on 6/5/16.
 */
public class Scanner implements Serializable,device {
    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getUser() {
        return uuser;
    }

    public void setUser(String uuser) {
        this.uuser = uuser;
    }

    public int getRoomNumber() {

```



```

        return roomNumber;
    }

    public void setRoomNumber(int roomNumber) {
        this.roomNumber = roomNumber;
    }

    public String getSection() {
        return section;
    }

    public void setSection(String section) {
        this.section = section;
    }

    public String getDate() {
        return date;
    }

    public void setDate(String date) {
        this.date = date;
    }

    public String getModel() {
        return model;
    }

    public void setModel(String model) {
        this.model = model;
    }

    public String getBrand() {
        return brand;
    }

    public void setBrand(String brand) {
        this.brand = brand;
    }

    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }

    public String getCommand() {
        return command;
    }

    public void setCommand(String command) {
        this.command = command;
    }

    private int id;
    private String user;
    private int roomNumber;
    private String section;
    private String date;
    private String model;
    private String brand;
    private String type;
    private String command;
}

```

```

package managment.model.DA;

import managment.model.TO.Computer;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;

/**
 * Created by saeedtavana on 7/5/16.
 */
public class computerDA {

    private Connection connection;
    private PreparedStatement preparedStatement1;

    public computerDA ()throws Exception
    {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        connection =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl","managment","13
74");
        connection.setAutoCommit(false);
    }
    public boolean insert(Computer obj)throws Exception
    {
        preparedStatement1=connection.prepareStatement("INSERT INTO computer VALUES
(?,?,?,?,?,?,?,?,?,?,?,?,?)");
        preparedStatement1.setInt(1,obj.getId());
        preparedStatement1.setString(2,obj.getUser());
        preparedStatement1.setInt(3,obj.getRoomNumber());
        preparedStatement1.setString(4,obj.getSection());
        preparedStatement1.setString(5,obj.getMotherboardnumber());
        preparedStatement1.setString(6,obj.getDisknumber());
        preparedStatement1.setInt(7,obj.getNumberOfdisk());
        preparedStatement1.setInt(8,obj.getCapacityOfDiskinG());
        preparedStatement1.setInt(9,obj.getRamInG());
        preparedStatement1.setString(10,obj.getCpu());
        preparedStatement1.setString(11,obj.getOs());
        preparedStatement1.setInt(12,obj.getNumberOfUser());
        preparedStatement1.setString(13,obj.getDate());
        preparedStatement1.setString(14,obj.getComments());
        int res=preparedStatement1.executeUpdate();
        return true;

    }
    public boolean delete(int id)throws Exception
    {
        preparedStatement1 =connection.prepareStatement("DELETE FROM computer WHERE
id=?");
        preparedStatement1.setInt(1,id);
        int res=preparedStatement1.executeUpdate();
        return true;

    }
    public boolean update(int id,String modle)throws Exception
    {
        preparedStatement1=connection.prepareStatement("UPDATE computer SET
motherboard=? WHERE id=?");
        preparedStatement1.setString(1,modle);
        preparedStatement1.setInt(2,id);
        int res=preparedStatement1.executeUpdate();
        return true;

    }
}

```

```

public ArrayList<Computer> select (int id) throws Exception
{
    PreparedStatement1=connection.prepareStatement("SELECT * FROM computer
WHERE id=?");
    PreparedStatement1.setInt(1,id);
    ResultSet resultSet=PreparedStatement1.executeQuery();
    ArrayList<Computer> list=new ArrayList<>();
    while (resultSet.next())
    {
        Computer computer=new Computer();
        computer.setId(resultSet.getInt("id"));
        computer.setUser(resultSet.getString("users"));
        computer.setRoomNumber(resultSet.getInt("room"));
        computer.setSection(resultSet.getString("section"));
        computer.setMotherboardnumber(resultSet.getString("motherboard"));
        computer.setDisknumber(resultSet.getString("disk"));
        computer.setNumberOfdisk(resultSet.getInt("countofdisk"));
        computer.setCapacityOfDiskInG(resultSet.getInt("capacityofdisk"));
        computer.setRamInG(resultSet.getInt("ram"));
        computer.setCpu(resultSet.getString("cpu"));
        computer.setOs(resultSet.getString("os"));
        computer.setNumberOfUser(resultSet.getInt("numberofuser"));
        computer.setComments(resultSet.getString("comments"));
        computer.setDate(resultSet.getString("date1"));
        list.add(computer);
    }
    resultSet.close();
    return list;
}

public void close() throws Exception
{
    connection.commit();
    PreparedStatement1.close();
    connection.close();
}
}

```

```

package managment.model.DA;

import managment.model.TO.Computer;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;

/**
 * Created by saeedtavana on 7/5/16.
 */
public class consoldA {
    private Connection connection;
    private PreparedStatement PreparedStatement1;
    public Date d=new Date();
}

```

```

    public SimpleDateFormat simpleDateFormat=new SimpleDateFormat("YYYY/MM/dd");
    public static String username;
    public consolDA ()throws Exception
    {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        connection =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl","managment","13
74");
        connection.setAutoCommit(false);
    }
    public void addconsol(String s)throws Exception
    {
        String date=simpleDateFormat.format(d)+" ";
        preparedStatement1=connection.prepareStatement("INSERT INTO console VALUES
(?,?)");
        preparedStatement1.setString(1,date);
        preparedStatement1.setString(2,s);
        int res=preparedStatement1.executeUpdate();

    }

    public void close()throws Exception
    {
        connection.commit();
        preparedStatement1.close();
        connection.close();

    }

}

```

```

package managment.model.DA;

import managment.model.TO.Computer;
import managment.model.TO.equipmentMainteneace;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;

/**
 * Created by saeedtavana on 7/5/16.
 */
public class equipmentDA {
    private Connection connection;
    private PreparedStatement preparedStatement1;

    public equipmentDA ()throws Exception
    {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        connection =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl","managment","13
74");
        connection.setAutoCommit(false);
    }
    public boolean addmaintenance(equipmentMainteneace obj)throws Exception
    {
        preparedStatement1=connection.prepareStatement("INSERT INTO maintenance
VALUES (?, ?, ?)");
        preparedStatement1.setInt(1,obj.getId());
        preparedStatement1.setString(2,obj.getDate());
        preparedStatement1.setString(3,obj.getTetx());
    }
}

```

```

        int res=preparedStatement1.executeUpdate();
        return true;
    }
    public ArrayList<equipmentMainteneace> selectAllmaintenance () throws Exception
    {
        preparedStatement1=connection.prepareStatement("SELECT * FROM
maintenance");
        ResultSet resultSet=preparedStatement1.executeQuery();
        ArrayList<equipmentMainteneace> list=new ArrayList<equipmentMainteneace>();
        while (resultSet.next())
        {
            equipmentMainteneace eq=new equipmentMainteneace();
            eq.setId(resultSet.getInt("id"));
            eq.setDate(resultSet.getString("date1"));
            eq.setTetx(resultSet.getString("comments"));
            list.add(eq);

        }
        resultSet.close();
        return list;
    }
    public int select no add maintenance()throws Exception
    {
        int res=0;
        preparedStatement1=connection.prepareStatement("select count(*) as tedat
from maintenance");
        ResultSet resultSet=preparedStatement1.executeQuery();
        if (resultSet==null)
        {
            res=0;
        }
        while(resultSet.next())
        {
            res=resultSet.getInt("tedat");
        }
        return res;
    }
    public ArrayList<equipmentMainteneace> select_maintenance_of_device (int id)
throws Exception
    {
        preparedStatement1=connection.prepareStatement("SELECT * FROM maintenance
WHERE id=?");
        preparedStatement1.setInt(1,id);
        ResultSet resultSet=preparedStatement1.executeQuery();
        ArrayList<equipmentMainteneace> list=new ArrayList<>();
        while (resultSet.next())
        {
            equipmentMainteneace eq=new equipmentMainteneace();
            eq.setId(resultSet.getInt("id"));
            eq.setDate(resultSet.getString("date1"));
            eq.setTetx(resultSet.getString("comments"));
            list.add(eq);

        }
        resultSet.close();
        return list;
    }
    public void close()throws Exception
    {
        connection.commit();
        preparedStatement1.close();
        connection.close();
    }

```

```

    }
}

```

```

package managment.model.DA;

import managment.model.TO.Computer;
import managment.model.TO.Monitor;
import managment.model.TO.Scaner;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;

/**
 * Created by saeedtavana on 7/5/16.
 */
public class monitorDA {
    private Connection connection;
    private PreparedStatement preparedStatement1;

    public monitorDA ()throws Exception
    {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        connection =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl","managment","13
74");
        connection.setAutoCommit(false);
    }
    public boolean insert(Monitor obj)throws Exception
    {
        preparedStatement1=connection.prepareStatement("INSERT INTO monitor VALUES
(?,?,?,?,?,?,?,?,?)");
        preparedStatement1.setInt(1,obj.getId());
        preparedStatement1.setString(2,obj.getUser());
        preparedStatement1.setInt(3,obj.getRoomNumber());
        preparedStatement1.setString(4,obj.getSection());
        preparedStatement1.setString(5,obj.getModel());
        preparedStatement1.setString(6,obj.getBrand());
        preparedStatement1.setString(7,obj.getType());
        preparedStatement1.setString(8,obj.getDate());
        preparedStatement1.setString(9,obj.getCommand());
        int res=preparedStatement1.executeUpdate();
        return true;
    }
    public boolean delete(int id)throws Exception
    {
        preparedStatement1=connection.prepareStatement("DELETE FROM monitor WHERE
id=?");
        preparedStatement1.setInt(1,id);
        int rs=preparedStatement1.executeUpdate();
        return true;
    }
    public boolean update(int id,String modle)throws Exception
    {
        preparedStatement1=connection.prepareStatement("UPDATE monitor SET model=?
WHERE id=?");
        preparedStatement1.setString(1,modle);
        preparedStatement1.setInt(2,id);
        int res=preparedStatement1.executeUpdate();
        return true;
    }
}

```

```

    }
    public ArrayList<Monitor> select (int id)throws Exception
    {
        preparedStatement1=connection.prepareStatement("SELECT * FROM monitor WHERE
id=?");
        preparedStatement1.setInt(1,id);
        ResultSet resultSet=preparedStatement1.executeQuery();
        ArrayList<Monitor> list=new ArrayList<>();
        while (resultSet.next())
        {
            Monitor monitor=new Monitor();
            monitor.setId(resultSet.getInt("id"));
            monitor.setUser(resultSet.getString("users"));
            monitor.setRoomNumber(resultSet.getInt("room"));
            monitor.setSection(resultSet.getString("section"));
            monitor.setModel(resultSet.getString("model"));
            monitor.setBrand(resultSet.getString("brand"));
            monitor.setType(resultSet.getString("type"));
            monitor.setDate(resultSet.getString("date1"));
            monitor.setCommand(resultSet.getString("comments"));
            list.add(monitor);
        }
        resultSet.close();
        return list;
    }
    public void close()throws Exception
    {
        connection.commit();
        preparedStatement1.close();
        connection.close();
    }
}

```

```

package managment.model.DA;

import managment.model.TO.Computer;
import managment.model.TO.NetworkUtility;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.Random;

/**
 * Created by saeedtavana on 7/5/16.
 */
public class networkDA {
    private Connection connection;
    private PreparedStatement preparedStatement1;

    public networkDA ()throws Exception
    {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        connection =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl","managment","13
74");
        connection.setAutoCommit(false);
    }
    public boolean insert(NetworkUtility obj)throws Exception
    {

```

```

        preparedStatement1=connection.prepareStatement("INSERT INTO network VALUES
(?,?,?,?,?,?)");
        preparedStatement1.setInt(1,obj.getId());
        preparedStatement1.setString(2,obj.getBrand());
        preparedStatement1.setString(3,obj.getModel());
        preparedStatement1.setString(4,obj.getType());
        preparedStatement1.setString(5,obj.getDate());
        preparedStatement1.setString(6,obj.getComments());
        int res=preparedStatement1.executeUpdate();
        return true;

    }

    public boolean delete(int id)throws Exception
    {
        preparedStatement1=connection.prepareStatement("DELETE FROM network WHERE
id=?");
        preparedStatement1.setInt(1,id);
        int re=preparedStatement1.executeUpdate();
        return true;

    }

    public boolean update(int id,String modle)throws Exception
    {
        preparedStatement1=connection.prepareStatement("UPDATE network SET model=?
WHERE id=?");
        preparedStatement1.setString(1,modle);
        preparedStatement1.setInt(2,id);
        int res=preparedStatement1.executeUpdate();
        return true;

    }

    public ArrayList<NetworkUtility> select (int id)throws Exception
    {
        preparedStatement1=connection.prepareStatement("SELECT * FROM network WHERE
id=?");
        preparedStatement1.setInt(1,id);
        ResultSet resultSet=preparedStatement1.executeQuery();
        ArrayList<NetworkUtility> list=new ArrayList<>();
        while (resultSet.next())
        {
            NetworkUtility nw=new NetworkUtility();
            nw.setId(resultSet.getInt("id"));
            nw.setBrand(resultSet.getString("brand"));
            nw.setModel(resultSet.getString("model"));
            nw.setType(resultSet.getString("type"));
            nw.setDate(resultSet.getString("date1"));
            nw.setComments(resultSet.getString("comments"));
            list.add(nw);
        }
        resultSet.close();
        return list;

    }

    public void close()throws Exception
    {
        connection.commit();
        preparedStatement1.close();
        connection.close();

    }

}

```



```

package managment.model.DA;

import managment.model.TO.Computer;
import managment.model.TO.Monitor;
import managment.model.TO.Printer;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;

/**
 * Created by saeedtavana on 7/5/16.
 */
public class printerDA {
    private Connection connection;
    private PreparedStatement preparedStatement1;

    public printerDA () throws Exception
    {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        connection =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl","managment","13
74");
        connection.setAutoCommit(false);
    }
    public boolean insert(Printer obj) throws Exception
    {
        preparedStatement1=connection.prepareStatement("INSERT INTO printer VALUES
(?,?,?,?,?,?,?,?,?)");
        preparedStatement1.setInt(1,obj.getId());
        preparedStatement1.setString(2,obj.getUser());
        preparedStatement1.setInt(3,obj.getRoomNumber());
        preparedStatement1.setString(4,obj.getSection());
        preparedStatement1.setString(5,obj.getModel());
        preparedStatement1.setString(6,obj.getBrand());
        preparedStatement1.setString(7,obj.getType());
        preparedStatement1.setString(8,obj.getDate());
        preparedStatement1.setString(9,obj.getCommand());
        int res=preparedStatement1.executeUpdate();
        return true;
    }
    public boolean delete(int id) throws Exception
    {
        preparedStatement1=connection.prepareStatement("DELETE FROM printer WHERE
id=?");
        preparedStatement1.setInt(1,id);
        int rs=preparedStatement1.executeUpdate();
        return true;
    }
    public boolean update(int id,String modle) throws Exception
    {
        preparedStatement1=connection.prepareStatement("UPDATE printer SET model=?
WHERE id=?");
        preparedStatement1.setString(1,modle);
        preparedStatement1.setInt(2,id);
        int res=preparedStatement1.executeUpdate();
        return true;
    }
    public ArrayList<Printer> select (int id) throws Exception
    {
        ArrayList<Printer> list=new ArrayList<>();
        preparedStatement1=connection.prepareStatement("SELECT * FROM printer WHERE
id=?");

```

```

        preparedStatement1.setInt(1, id);
        ResultSet resultSet=preparedStatement1.executeQuery();
        if (resultSet==null)
        {
            list=null;
        }

        else {
            while (resultSet.next()) {
                Printer monitor = new Printer();
                monitor.setId(resultSet.getInt("id"));
                monitor.setUser(resultSet.getString("users"));
                monitor.setRoomNumber(resultSet.getInt("room"));
                monitor.setSection(resultSet.getString("section"));
                monitor.setModel(resultSet.getString("model"));
                monitor.setBrand(resultSet.getString("brand"));
                monitor.setType(resultSet.getString("type"));
                monitor.setDate(resultSet.getString("date1"));
                monitor.setCommand(resultSet.getString("comments"));
                list.add(monitor);
            }
        }
        resultSet.close();
        return list;
    }

    public void close()throws Exception
    {

        connection.commit();
        preparedStatement1.close();
        connection.close();
    }
}

```

```

package managment.model.DA;

import managment.model.TO.Computer;
import managment.model.TO.Scaner;
import managment.model.TO.room;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;

/**
 * Created by saeedtavana on 7/5/16.
 */
public class roomDA {
    private Connection connection;
    private PreparedStatement preparedStatement1;

    public roomDA ()throws Exception
    {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        connection =
        DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl","managment","1374");
        connection.setAutoCommit(false);
    }
    public boolean insert(room obj)throws Exception
    {

```

```

        preparedStatement1=connection.prepareStatement("INSERT INTO room VALUES
(?,?)");
        preparedStatement1.setInt(1,obj.getNumber());
        preparedStatement1.setInt(2,obj.getTelephone());
        int re=preparedStatement1.executeUpdate();
        return true;

    }

    public boolean delete(int id)throws Exception
    {
        preparedStatement1=connection.prepareStatement("DELETE FROM room WHERE
roomnumber=?");
        preparedStatement1.setInt(1,id);
        int res=preparedStatement1.executeUpdate();
        return true;

    }

    public boolean update(int id,int tele)throws Exception
    {
        preparedStatement1=connection.prepareStatement("UPDATE room SET telephone=?
WHERE roomnumber=?");
        preparedStatement1.setInt(1,tele);
        preparedStatement1.setInt(2,id);
        int res=preparedStatement1.executeUpdate();
        return true;

    }

    public ArrayList<room> select (int id)throws Exception
    {
        preparedStatement1=connection.prepareStatement("SELECT * FROM room WHERE
roomnumber=?");
        preparedStatement1.setInt(1,id);
        ResultSet resultSet=preparedStatement1.executeQuery();
        ArrayList<room> list=new ArrayList<>();
        while (resultSet.next())
        {
            room room=new room();
            room.setNumber(resultSet.getInt("roomnumber"));
            room.setTelephone(resultSet.getInt("telephone"));
            list.add(room);
        }
        resultSet.close();
        return list;

    }

    public void close()throws Exception
    {
        connection.commit();
        preparedStatement1.close();
        connection.close();

    }

}

```

```

package managment.model.DA;

import managment.model.TO.Computer;
import managment.model.TO.Monitor;
import managment.model.TO.Printer;
import managment.model.TO.Scanner;

import java.sql.Connection;

```

```

import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;

/**
 * Created by saeedtavana on 7/5/16.
 */
public class ScannerDA {
    private Connection connection;
    private PreparedStatement preparedStatement1;

    public ScannerDA () throws Exception
    {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        connection =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl","managment","13
74");
        connection.setAutoCommit(false);
    }
    public boolean insert(Scanner obj) throws Exception
    {
        preparedStatement1=connection.prepareStatement("INSERT INTO scanner VALUES
(?,?,?, ?, ?, ?, ?, ?)");
        preparedStatement1.setInt(1,obj.getId());
        preparedStatement1.setString(2,obj.getUser());
        preparedStatement1.setInt(3,obj.getRoomNumber());
        preparedStatement1.setString(4,obj.getSection());
        preparedStatement1.setString(5,obj.getModel());
        preparedStatement1.setString(6,obj.getBrand());
        preparedStatement1.setString(7,obj.getType());
        preparedStatement1.setString(8,obj.getDate());
        preparedStatement1.setString(9,obj.getCommand());
        int res=preparedStatement1.executeUpdate();
        return true;
    }
    public boolean delete(int id) throws Exception
    {
        preparedStatement1=connection.prepareStatement("DELETE FROM scanner WHERE
id=?");
        preparedStatement1.setInt(1,id);
        int rs=preparedStatement1.executeUpdate();
        return true;
    }
    public boolean update(int id,String modle) throws Exception
    {
        preparedStatement1=connection.prepareStatement("UPDATE scanner SET model=?
WHERE id=?");
        preparedStatement1.setString(1,modle);
        preparedStatement1.setInt(2,id);
        int res=preparedStatement1.executeUpdate();
        return true;
    }
    public ArrayList<Scanner> select (int id) throws Exception
    {
        ArrayList<Scanner> list=new ArrayList<>();
        preparedStatement1=connection.prepareStatement("SELECT * FROM scanner WHERE
id=?");
        preparedStatement1.setInt(1,id);
        ResultSet resultSet=preparedStatement1.executeQuery();
        if (resultSet==null)
        {
            list=null;
        }
        else {

```

```

        while (resultSet.next()) {
            Scanner monitor = new Scanner();
            monitor.setId(resultSet.getInt("id"));
            monitor.setUser(resultSet.getString("users"));
            monitor.setRoomNumber(resultSet.getInt("room"));
            monitor.setSection(resultSet.getString("section"));
            monitor.setModel(resultSet.getString("model"));
            monitor.setBrand(resultSet.getString("brand"));
            monitor.setType(resultSet.getString("type"));
            monitor.setDate(resultSet.getString("date1"));
            monitor.setCommand(resultSet.getString("comments"));
            list.add(monitor);
        }
    }
    resultSet.close();
    return list;
}

public void close() throws Exception
{
    connection.commit();
    preparedStatement1.close();
    connection.close();
}
}
}

```

```

package managment.model.DA;

import managment.model.TO.Computer;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;

/**
 * Created by saeedtavana on 7/5/16.
 */
public class userDA {
    private Connection connection;
    private PreparedStatement preparedStatement1;

    public userDA () throws Exception
    {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        connection =
        DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl", "managment",
        "1374");
    }

    public boolean checkuser(String uname,String pass) throws Exception
    {
        boolean ans=false;
        preparedStatement1 = connection.prepareStatement("SELECT * FROM
managment.users");
        ResultSet resultSet = preparedStatement1.executeQuery();
        while (resultSet.next())
        {
            String username=resultSet.getString("username");
            String password=resultSet.getString("password");
            if (username.equals(uname) && password.equals(pass))

```

```

        {
            ans=true;
        }

    }
    return ans;

}

public boolean checkadmin(String uname,String pass)throws Exception
{
    boolean ans=false;
    preparedStatement1=connection.prepareStatement("SELECT * FROM
managment.admin ");
    ResultSet resultSet=preparedStatement1.executeQuery();
    while (resultSet.next())
    {
        String username=resultSet.getString("username");
        String password=resultSet.getString("password");
        if (uname.equals(username) && pass.equals(password))
        {
            ans=true;
        }
    }
    return ans;
}

public void close()throws Exception
{
    // connection.commit();
    preparedStatement1.close();
    connection.close();
}
}
}

```

```

package managment.model.BL;

import managment.model.DA.computerDA;
import managment.model.TO.Computer;

import java.util.ArrayList;

/**
 * Created by saeedtavana on 7/5/16.
 */
public class computerManager {
    public boolean addcomputer (Computer obj)throws Exception
    {
        computerDA computerDA=new computerDA();
        boolean ans=computerDA.insert(obj);
        computerDA.close();
        return ans;
    }

    public boolean deletecomputer(int id)throws Exception
    {
        computerDA computerDA=new computerDA();
    }
}

```

```

        boolean ans=computerDA.delete(id) ;
        computerDA.close() ;
        return ans;

    }
    public boolean updatecomputer(int id,String model)throws Exception
    {
        computerDA computerDA=new computerDA() ;
        boolean ans=computerDA.update(id,model) ;
        computerDA.close() ;
        return ans;

    }
    public ArrayList<Computer> selectcomputer(int id)throws Exception
    {
        computerDA computerDA=new computerDA() ;
        ArrayList<Computer> objA=computerDA.select(id) ;
        computerDA.close() ;
        return objA;

    }

}

```

```

package managment.model.BL;

import managment.model.DA.consoldA;

/**
 * Created by saeedtavana on 7/5/16.
 */
public class consolManager {
    public void addconsol(String s) throws Exception
    {
        consoldA consoldA=new consoldA() ;
        consoldA.addconsol(s) ;
        consoldA.close() ;

    }

}

```

```

package managment.model.BL;

import managment.model.DA.equipmentDA;
import managment.model.TO.equipmentMainteneace;

import java.util.ArrayList;

/**
 * Created by saeedtavana on 7/5/16.
 */
public class equipmentManager {

    public boolean addmaintenance(equipmentMainteneace obj)throws Exception
    {
        equipmentDA equipmentDA=new equipmentDA() ;
        boolean ans=equipmentDA.addmaintenance(obj) ;
        equipmentDA.close() ;
        return ans;

    }
    public ArrayList<equipmentMainteneace> selectAllmaintenance () throws Exception
    {
        equipmentDA equipmentDA=new equipmentDA() ;
        ArrayList<equipmentMainteneace> equobj=equipmentDA.selectAllmaintenance() ;

    }

}

```

```

        equipmentDA.close();
        return equobj;
    }
    public int select_no_add_maintenance() throws Exception
    {
        int ans=0;
        equipmentDA equipmentDA=new equipmentDA();
        ans=equipmentDA.select_no_add_maintenance();
        equipmentDA.close();
        return ans;
    }
    public ArrayList<equipmentMainteneace> select_maintenance_of_device (int id)
    throws Exception
    {
        equipmentDA equipmentDA=new equipmentDA();
        ArrayList<equipmentMainteneace>
obj=equipmentDA.select maintenance of device(id);
        equipmentDA.close();
        return obj;
    }
}

```

```

package managment.model.BL;

import managment.model.DA.monitorDA;
import managment.model.TO.Monitor;
import managment.model.TO.Scaner;

import java.util.ArrayList;

/**
 * Created by saeedtavana on 7/5/16.
 */
public class monitorManager {
    public boolean addmonitor(Monitor obj) throws Exception
    {
        monitorDA monitorDA=new monitorDA();
        boolean ans=monitorDA.insert(obj);
        monitorDA.close();
        return ans;
    }
    public boolean deletemonitor(int id) throws Exception
    {
        monitorDA monitorDA=new monitorDA();
        boolean ans=monitorDA.delete(id);
        monitorDA.close();
        return ans;
    }
    public boolean updatemonitor(int id,String modle) throws Exception
    {
        monitorDA monitorDA=new monitorDA();
        boolean ans=monitorDA.update(id,modle);
        monitorDA.close();
        return ans;
    }
    public ArrayList<Monitor> selectmonitor (int id) throws Exception
    {
        monitorDA monitorDA=new monitorDA();
        ArrayList<Monitor> obj=monitorDA.select(id);
        monitorDA.close();
    }
}

```



```

        return obj;
    }
}

```

```

package managment.model.BL;

import managment.model.DA.networkDA;
import managment.model.TO.NetworkUtility;

import java.util.ArrayList;

/**
 * Created by saeedtavana on 7/5/16.
 */
public class networkManager {

    public boolean addnetwor(NetworkUtility obj)throws Exception
    {
        networkDA networkDA=new networkDA();
        boolean ans=networkDA.insert(obj);
        networkDA.close();
        return ans;
    }

    public boolean deletenetwor(int id)throws Exception
    {
        networkDA networkDA=new networkDA();
        boolean ans=networkDA.delete(id);
        networkDA.close();
        return ans;
    }

    public boolean updatenetwor(int id,String modle)throws Exception
    {
        networkDA networkDA=new networkDA();
        boolean ans=networkDA.update(id,modle);
        networkDA.close();
        return ans;
    }

    public ArrayList<NetworkUtility> selectnetwor (int id)throws Exception
    {
        networkDA networkDA=new networkDA();
        ArrayList<NetworkUtility> obj=networkDA.select(id);
        networkDA.close();
        return obj;
    }

}

```

```

package managment.model.BL;

import managment.model.DA.printerDA;
import managment.model.TO.Printer;

import java.util.ArrayList;

/**
 * Created by saeedtavana on 7/5/16.
 */
public class printermanager {
    public boolean addprinter(Printer obj)throws Exception
    {
        printerDA printerDA=new printerDA();
    }
}

```

```

        boolean ans=printerDA.insert(obj);
        printerDA.close();
        return ans;

    }
    public boolean deleteprinter(int id)throws Exception
    {
        printerDA printerDA=new printerDA();
        boolean ans=printerDA.delete(id);
        printerDA.close();
        return ans;

    }
    public boolean updateprinter(int id,String modle)throws Exception
    {
        printerDA printerDA=new printerDA();
        boolean ans=printerDA.update(id,modle);
        printerDA.close();
        return ans;

    }
    public ArrayList<Printer> selectprinte (int id)throws Exception
    {
        printerDA printerDA=new printerDA();
        ArrayList<Printer> obj=printerDA.select(id);
        printerDA.close();
        return obj;

    }
}

```

```

package managment.model.BL;

import managment.model.DA.roomDA;
import managment.model.TO.room;

import java.util.ArrayList;

/**
 * Created by saeedtavana on 7/5/16.
 */
public class roomManager {
    public boolean addroom(room obj)throws Exception
    {
        roomDA roomDA=new roomDA();
        boolean ans =roomDA.insert(obj);
        roomDA.close();
        return ans;

    }
    public boolean deleteroom(int id)throws Exception
    {
        roomDA roomDA=new roomDA();
        boolean ans=roomDA.delete(id);
        roomDA.close();
        return ans;

    }
    public boolean updateroom(int id,int modle)throws Exception
    {
        roomDA roomDA=new roomDA();
        boolean ans=roomDA.update(id,modle);
        roomDA.close();
        return ans;

    }
}

```

```

public ArrayList<room> selectrooms (int id) throws Exception
{
    roomDA roomDA=new roomDA();
    ArrayList<room> obj=roomDA.select(id);
    roomDA.close();
    return obj;
}
}

```

```

package managment.model.BL;

import managment.model.DA.scannerDA;
import managment.model.TO.Scanner;

import java.util.ArrayList;

/**
 * Created by saeedtavana on 7/5/16.
 */
public class scannerManager {
    public boolean addscanner(Scanner obj) throws Exception
    {
        scannerDA scannerDA=new scannerDA();
        boolean ans=scannerDA.insert(obj);
        scannerDA.close();
        return ans;
    }

    public boolean deletescanner(int id) throws Exception
    {
        scannerDA scannerDA=new scannerDA();
        boolean ans=scannerDA.delete(id);
        scannerDA.close();
        return ans;
    }

    public boolean updatescanner(int id,String modle) throws Exception
    {
        scannerDA scannerDA=new scannerDA();
        boolean ans=scannerDA.update(id,modle);
        scannerDA.close();
        return ans;
    }

    public ArrayList<Scanner> selectscanner (int id) throws Exception
    {
        scannerDA scannerDA=new scannerDA();
        ArrayList<Scanner> obj=scannerDA.select(id);
        scannerDA.close();
        return obj;
    }
}

```

```

package managment.model.BL;

import managment.model.DA.userDA;

import java.sql.SQLException;

/**
 * Created by saeedtavana on 7/5/16.
 */
public class userManager {

```

```

public boolean checkuser2(String uname,String pass)throws Exception
{

    boolean ans;
    userDA userDA = new userDA();
    ans = userDA.checkuser(uname,pass);
    userDA.close();
    return ans;

}
public boolean checkadmin(String uname,String pass)throws Exception
{
    userDA userDA=new userDA();
    boolean ans=userDA.checkadmin(uname,pass);
    userDA.close();
    return ans;

}
}

```

## server controller

```

package managment.controller.logincontroler;

import managment.model.BL.userManager;
import managment.view.ServerPage;
import managment.view.login;

import javax.swing.*;
import java.awt.*;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;

/**
 * Created by saeedtavana on 5/9/16.
 */
public class buttonnloginControler implements KeyListener {
    public login login3;
    @Override
    public void keyTyped(KeyEvent e) {

    }

    @Override
    public void keyPressed(KeyEvent e) {
        String username= login3.jTextField.getText();
        String password=login3.jPasswordField.getText();
        boolean ans=false;
        if (username.equals("") ||password.equals(""))
        {
            JOptionPane.showMessageDialog(login3.jFrame,"the username or password
must not empty","Warning : empty input",
JOptionPane.WARNING_MESSAGE);
        }

        try {

            userManager userManager = new userManager();
            ans = userManager.checkadmin(username, password);
            if (ans==true)
            {
                ServerPage serverPage=new ServerPage();
                serverPage.showServerPage();
                ServerPage.logedname.setText(username);
            }
        }
    }
}

```

```

        login3.jFrame.setVisible(false);
    }
    else if (ans==false)
    {
        JOptionPane.showMessageDialog(login3.jFrame,"the user name or
password is incorrect","Wrong input",
        JOptionPane.ERROR_MESSAGE);
        login3.jTextField.setText("");
        login3.jPasswordField.setText("");
    }
}
catch (Exception e1)
{
}

}

@Override
public void keyReleased(KeyEvent e) {

}
}

```

```

package managment.controller.logincontroller;

import managment.view.login;

import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;

/**
 * Created by saeedtavana on 5/9/16.
 */
public class button2longController implements KeyListener {
    public login login2;
    @Override
    public void keyTyped(KeyEvent e) {

    }

    @Override
    public void keyPressed(KeyEvent e) {

        login2.jPasswordField.setText("");
        login2.jTextField.setText("");
    }

    @Override
    public void keyReleased(KeyEvent e) {

    }
}

```

```

package managment.controller.logincontroller;

```

```

import com.sun.corba.se.impl.iior.JIDLObjectKeyTemplate;
import com.sun.scenario.effect.impl.sw.sse.SSEBlend_SRC_OUTPeer;
import javafx.scene.input.KeyCode;
import managment.model.BL.userManager;
import managment.view.ServerPage;
import managment.view.login;

import javax.swing.*;
import javax.swing.plaf.basic.BasicOptionPaneUI;
import java.awt.*;
import java.awt.event.*;

/**
 * Created by saeedtavana on 5/9/16.
 */
public class loginControler implements ActionListener ,MouseListener{

    public login login1;

    @Override
    public void mouseClicked(MouseEvent e) {

    }

    @Override
    public void mousePressed(MouseEvent e) {

    }

    @Override
    public void mouseReleased(MouseEvent e) {

    }

    @Override
    public void mouseEntered(MouseEvent e) {

    }

    @Override
    public void mouseExited(MouseEvent e) {

    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if (e.getActionCommand().equals("clear button")) {
            login1.jTextField.setText("");
            login1.jPasswordField.setText("");
        } else if (e.getActionCommand().equals("enter button")) {

            String username = login1.jTextField.getText();
            String password = login1.jPasswordField.getText();
            boolean ans = false;
            if (username.equals("") || password.equals("")) {
                JOptionPane.showMessageDialog(login1.jFrame, "the username or
password must not empty", "Warning : empty input",
                JOptionPane.WARNING_MESSAGE);
            }

            try {

```



```

        if (e.getActionCommand().equals("exitserver"))
        {
            int dialog= JOptionPane.showConfirmDialog(ServerPage.serverpage,"do you
want to exit ?", "question",
                JOptionPane.YES_NO_OPTION);
            if (dialog==JOptionPane.YES_OPTION)
            {
                System.exit(0);
            }
        }
    }
}

```

```

package managment.controller.ServerConroler;

import managment.RmiServer.RmiImplimentation.RmiserverControl;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;

/**
 * Created by user on 5/13/2016.
 */
public class startserverbutton implements KeyListener,ActionListener
{
    @Override
    public void actionPerformed(ActionEvent e) {
        if (e.getActionCommand().equals("startserver"))
        {
            try {
                RmiserverControl rmiserverControl = RmiserverControl.getInstance();
                rmiserverControl.start();
            }
            catch (Exception e1)
            {
            }
        }
    }

    @Override
    public void keyTyped(KeyEvent e) {

    }

    @Override
    public void keyPressed(KeyEvent e) {
        try {
            RmiserverControl rmiserverControl = RmiserverControl.getInstance();
            rmiserverControl.start();
        }
        catch (Exception e1)
        {
        }
    }

    @Override
    public void keyReleased(KeyEvent e) {

```



```

    }
}

```

```

package managment.controller.ServerConroler;

import managment.RmiServer.RmiImplimentation.RmiserverControl;
import managment.view.ServerPage;

import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;

/**
 * Created by user on 5/13/2016.
 */
public class stopserverbutton implements KeyListener,ActionListener
{
    @Override
    public void actionPerformed(ActionEvent e) {
        if (e.getActionCommand().equals("stopserver"))
        {
            int dialog= JOptionPane.showConfirmDialog(ServerPage.serverpage,"do you
want to exit ?","question",
                JOptionPane.YES_NO_OPTION);
            if (dialog==JOptionPane.YES_OPTION)
            {
                try
                {
                    RmiserverControl
rmiserverControl=RmiserverControl.getInstance();
                    rmiserverControl.stop();
                }
                catch (Exception e1)
                {
                }
            }
        }

        @Override
        public void keyTyped(KeyEvent e) {

        }

        @Override
        public void keyPressed(KeyEvent e) {
            int dialog= JOptionPane.showConfirmDialog(ServerPage.serverpage,"do you
want to exit ?","question",
                JOptionPane.YES_NO_OPTION);
            if (dialog==JOptionPane.YES_OPTION)
            {
                try
                {
                    RmiserverControl rmiserverControl=RmiserverControl.getInstance();
                    rmiserverControl.stop();
                }
                catch (Exception e1)
                {
                }
            }
        }
    }
}

```

```

    }

    @Override
    public void keyReleased(KeyEvent e) {

    }

}

```

```

package managment.controller.ServerConroler;

import javafx.application.Application;
import managment.view.ServerPage;
import managment.view.helpfourm;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import java.io.IOException;

/**
 * Created by user on 5/20/2016.
 */
public class sysfielscontroler implements KeyListener,ActionListener{

    public helpfourm helpfourm=new helpfourm();

    @Override
    public void actionPerformed(ActionEvent e) {

    }

    @Override
    public void keyTyped(KeyEvent e) {

    }

    @Override
    public void keyPressed(KeyEvent e) {

        String command = ServerPage.sys.getText();
        String ss=ServerPage.seconds.getText();
        if (e.getKeyCode()==KeyEvent.VK_ENTER) {
            switch (command) {
                case "apptimer":
                    if (ss.equals("")) {
                        JOptionPane.showMessageDialog(ServerPage.serverpage,
"please choose the minute", "warning",
JOptionPane.WARNING_MESSAGE);
                    }
                    try {
                        long l = Long.parseLong(ss);
                        Thread.sleep(l);
                        System.exit(0);
                    } catch (InterruptedException e1) {
                        e1.printStackTrace();
                    }
                    break;
                case "system_user":
                    try {
                        Runtime.getRuntime().exec("cmd.exe /c start");
                    }

```

```

        ServerPage.sys.setText("");

    } catch (IOException ex) {
        ex.printStackTrace();
    }
    break;
case "exit":
    System.exit(0);
    break;
case "timera":
    try {
        Runtime.getRuntime().exec("shutdown.exe -a");
        JOptionPane.showMessageDialog(ServerPage.serverpage,
String.format("server shutdown canceled"), "Info"
            , JOptionPane.NO_OPTION);
        ServerPage.sys.setText("");

    } catch (IOException ex) {
        ex.printStackTrace();
    }
    break;

    case "timer":
        try {
            if (ss.equals("")) {
                JOptionPane.showMessageDialog(ServerPage.serverpage,
"please choose the minute", "warning",
                    JOptionPane.WARNING_MESSAGE);
            }
            int sec = Integer.parseInt(ss);
            int minut = sec / 60;

            Runtime.getRuntime().exec("shutdown.exe -s -t " + ss);
            JOptionPane.showMessageDialog(ServerPage.serverpage,
String.format("server will shutdown after " +
                ":" + minut + " minutes"), "Info",
JOptionPane.NO_OPTION);
            ServerPage.seconds.setText("");

        } catch (IOException ex) {
            ex.printStackTrace();
        }
        break;
case "dba":
    try {
        Runtime.getRuntime().exec("cmd.exe /c start");
        ServerPage.sys.setText("");
    } catch (IOException ex) {
        ex.printStackTrace();
    }

    break;
case "DBA":
    try {
        Runtime.getRuntime().exec("cmd.exe /c start");
        ServerPage.sys.setText("");
    } catch (IOException ex) {
        ex.printStackTrace();
    }
    break;
case "cmd":
    try {
        Runtime.getRuntime().exec("cmd.exe /c start");
        // System.out.println("ok");
    } catch (IOException ex) {
        ex.printStackTrace();
    }
}

```

```

        break;
    case "help":
        helpfourm.show1();
        break;
    case "HELP":
        helpfourm.show1();
        break;
    case "Help":
        helpfourm.show1();
        break;
    case "?":
        helpfourm.show1();
        break;

    default:
        //JOptionPane.showMessageDialog (ServerPage.serverpage,"wrong
input","error",JOptionPane.ERROR_MESSAGE);
        break;
    }
}

@Override
public void keyReleased(KeyEvent e) {
}
}

```

```

package managment.controller;

import managment.RmiServer.RmiImplimentation.RmiserverControl;
import managment.RmiServer.RmiImplimentation.personalLoginAddconsoleImp;
import managment.model.BL.userManager;
import managment.view.ServerPage;
import managment.view.helpfourm;
import managment.view.login;

import javax.swing.text.BadLocationException;
import javax.swing.text.Document;

/**
 * Created by saeedtavana on 5/8/16.
 */
public class main {
    public static void main(String[] args) throws Exception {

        //helpfourm h=new helpfourm();
        //h.show1();
        login l=new login();
        l.showLogin();
        //ServerPage s=new ServerPage();
        //s.showServerPage();

        //RmiserverControl rm=RmiserverControl.getInstance();
        //rm.start();
        //System.out.println("the server is up ....");
    }
}

```

```
}  
}
```

## rmi server

```
package managment.RmiServer.Remote;  
  
import java.rmi.Remote;  
import java.util.ArrayList;  
  
/**  
 * Created by user on 5/12/2016.  
 */  
public interface Computer extends Remote {  
    public boolean addComputer(managment.model.TO.Computer obj) throws Exception;  
    public boolean delecomputer(int id) throws Exception;  
    public boolean updatecomputer(int id, String motherb ) throws Exception;  
    public ArrayList<managment.model.TO.Computer> selectcomputer(int id) throws  
Exception;  
    public int show(int id) throws Exception;  
}
```

```
package managment.RmiServer.Remote;  
  
import managment.model.TO.equipmentMainteneace;  
  
import java.rmi.Remote;  
import java.util.ArrayList;  
  
/**  
 * Created by saeedtavana on 7/3/16.  
 */  
public interface equipmentMaintenance extends Remote{  
    public boolean addmaintenance(equipmentMainteneace obj) throws Exception;  
    public ArrayList<equipmentMaintenance> selectAllmaintenance () throws  
Exception;  
    public int select_no_add_maintenance() throws Exception;  
    public ArrayList<equipmentMaintenance> select_maintenance_of_device (int id)  
throws Exception;  
}
```

```
package managment.RmiServer.Remote;  
  
import managment.model.TO.*;  
  
import java.rmi.Remote;  
import java.util.ArrayList;  
  
/**  
 * Created by user on 5/12/2016.  
 */  
public interface Monitor extends Remote {  
    public boolean addmonitor(managment.model.TO.Monitor obj) throws Exception;  
    public boolean delemonitor(int id) throws Exception;  
    public boolean updatemonitor(int id, String model) throws Exception;  
    public ArrayList<managment.model.TO.Monitor> selectmonitro(int id) throws  
Exception;  
}
```

```

package managment.RmiServer.Remote;

import managment.model.TO.*;

import java.rmi.Remote;
import java.util.ArrayList;

/**
 * Created by user on 5/12/2016.
 */
public interface NetworkUtility extends Remote {
    public boolean addpnetwork(managment.model.TO.NetworkUtility obj) throws
Exception;
    public boolean delenetwork(int id)throws Exception;
    public boolean updatenetwork(int id,String motherb )throws Exception;
    public ArrayList<managment.model.TO.NetworkUtility> selecttnetwork(int id)throws
Exception;
}

```

```

package managment.RmiServer.Remote;

import java.rmi.Remote;

/**
 * Created by user on 5/13/2016.
 */
public interface PersonLoginAddconsol extends Remote{
    public void addconsole(String s)throws Exception;
    public boolean checkuser(String user,String pass) throws Exception;
}

```

```

package managment.RmiServer.Remote;

import managment.model.TO.*;

import java.rmi.Remote;
import java.util.ArrayList;

/**
 * Created by user on 5/12/2016.
 */
public interface Printer extends Remote {
    public boolean addprinter(managment.model.TO.Printer obj) throws Exception;
    public boolean deleprinter(int id)throws Exception;
    public boolean updatprinter(int id,String motherb )throws Exception;
    public ArrayList<managment.model.TO.Printer> selectprinter(int id)throws
Exception;
}

```

```

package managment.RmiServer.Remote;

import java.rmi.Remote;
import java.util.ArrayList;

/**
 * Created by saeedtavana on 7/3/16.
 */
public interface room extends Remote {
    public boolean addroom(managment.model.TO.room obj) throws Exception;
    public boolean deleteroom(int id) throws Exception;
}

```

```

        public boolean updateroom(int id,int telephone)throws Exception;
        public ArrayList<managment.model.TO.room> selectroom(int id)throws Exception;
    }

```

```

package managment.RmiServer.Remote;

import managment.model.TO.*;

import java.rmi.Remote;
import java.util.ArrayList;

/**
 * Created by user on 5/12/2016.
 */
public interface Scanner extends Remote {
    public boolean addscanneer(managment.model.TO.Scanner obj) throws Exception;
    public boolean delescanner(int id)throws Exception;
    public boolean updatescanner(int id,String motherb )throws Exception;
    public ArrayList<managment.model.TO.Scanner> selectscanner(int id)throws
Exception;
}

```

```

package managment.RmiServer.RmiImplimentation;

import jdk.nashorn.internal.runtime.regex.joni.Syntax;
import managment.RmiServer.Remote.Computer;
import managment.model.BL.computerManager;
import managment.view.ServerPage;

import javax.swing.text.Document;
import java.rmi.server.UnicastRemoteObject;
import java.util.ArrayList;

/**
 * Created by saeedtavana on 7/1/16.
 */
public class computerImp extends UnicastRemoteObject implements Computer {
    public computerImp() throws Exception
    {
        java.rmi.Naming.rebind("computerremote",this);
    }
    @Override
    public boolean addComputer(managment.model.TO.Computer obj) throws Exception {

        managment.model.TO.Computer computer=obj;
        computerManager computerManager=new computerManager();
        boolean ans=computerManager.addcomputer(computer);
        return ans;

    }

    @Override
    public boolean delecomputer(int id) throws Exception {
        int id2=id;

        computerManager computerManager=new computerManager();
        boolean ans=computerManager.deletecomputer(id2);
        return ans;
    }
}

```

```

    }

    @Override
    public boolean updatecomputer(int id, String motherb) throws Exception
    {
        int id1=id;
        String mo=motherb;
        computerManager computerManager=new computerManager();
        boolean ans=computerManager.updatecomputer(id1,mo);
        return ans;

    }

    @Override
    public ArrayList<managment.model.TO.Computer> selectcomputer(int id) throws
Exception {

        int id1=id;
        computerManager computerManager=new computerManager();
        ArrayList<managment.model.TO.Computer>
obj=computerManager.selectcomputer(id1);
        return obj;

    }

    @Override
    public int show(int id) throws Exception {

        return id+10;

    }
}

```

```

package managment.RmiServer.RmiImplimentation;

import managment.RmiServer.Remote.equipmentMaintenace;
import managment.model.BL.computerManager;
import managment.model.BL.equipmentManager;
import managment.model.TO.equipmentMainteneace;
import managment.view.ServerPage;

import javax.swing.text.Document;
import java.rmi.server.UnicastRemoteObject;
import java.util.ArrayList;

/**
 * Created by saeedtavana on 7/3/16.
 */
public class equimpentMaintenanceImp extends UnicastRemoteObject implements
equipmentMaintenace {

    public equimpentMaintenanceImp() throws Exception
    {
        java.rmi.Naming.rebind("equipmentremote",this);
    }
    @Override
    public boolean addmaintenance(equipmentMainteneace obj) throws Exception {
        equipmentMainteneace equipmentMainteneace=obj;

        equipmentManager equipmentManager=new equipmentManager();
        boolean ans=equipmentManager.addmaintenance(equipmentMainteneace);
        return ans;
    }
}

```



```

    }

    @Override
    public ArrayList<equipmentMainteneace> selectAllmaintenance() throws Exception
    {
        equipmentManager equipmentManager=new equipmentManager();
        ArrayList<equipmentMainteneace>
obj=equipmentManager.selectAllmaintenance();
        return obj;
    }

    @Override
    public int select no add maintenance() throws Exception {
        int number;

        equipmentManager equipmentManager=new equipmentManager();
        number=equipmentManager.select_no_add_maintenance();

        return number;
    }

    @Override
    public ArrayList<equipmentMainteneace> select_maintenance_of_device(int id)
throws Exception {
        int id1=id;
        equipmentManager equipmentManager=new equipmentManager();
        ArrayList<equipmentMainteneace>
obj=equipmentManager.select_maintenance_of_device(id1);
        return obj;
    }
}

```

```

package managment.RmiServer.RmiImplimentation;

import managment.RmiServer.Remote.Monitor;
import managment.model.BL.equipmentManager;
import managment.model.BL.monitorManager;
import managment.view.ServerPage;

import javax.swing.text.Document;
import java.rmi.server.UnicastRemoteObject;
import java.util.ArrayList;

/**
 * Created by saeedtavana on 7/1/16.
 */
public class monitorImp extends UnicastRemoteObject implements Monitor {
    public monitorImp() throws Exception
    {
        java.rmi.Naming.rebind("monotorremote",this);
    }

    @Override
    public boolean addmonitor(managment.model.TO.Monitor obj) throws Exception {
        managment.model.TO.Monitor monitor=obj;
        monitorManager monitorManager=new monitorManager();
        boolean ans=monitorManager.addmonitor(monitor);
    }
}

```

```

        return ans;

    }

    @Override
    public boolean delemonitor(int id) throws Exception {
        int id1=id;
        monitorManager monitorManager=new monitorManager();
        boolean ans=monitorManager.deletemonitor(id1);
        return ans;

    }

    @Override
    public boolean updatemonitor(int id, String model) throws Exception {
        int id1=id;
        String s=model;

        monitorManager monitorManager=new monitorManager();
        boolean ans=monitorManager.updatemonitor(id1,s);
        return ans;

    }

    @Override
    public ArrayList<managment.model.TO.Monitor> selectmonitro(int id) throws
Exception {
        int id1=id;
        monitorManager monitorManager=new monitorManager();
        ArrayList<managment.model.TO.Monitor>
obj=monitorManager.selectmonitor(id1);
        if (obj==null)
        {
            return null;
        }
        return obj;

    }
}

```

```

package managment.RmiServer.RmiImplimentation;

import managment.RmiServer.Remote.NetworkUtility;
import managment.model.BL.networkManager;

import java.rmi.server.UnicastRemoteObject;
import java.util.ArrayList;
import java.rmi.*;

/**
 * Created by saeedtavana on 7/1/16.
 */
public class networkUtilityImp extends UnicastRemoteObject implements
NetworkUtility {
    public networkUtilityImp () throws Exception
    {
        java.rmi.Naming.rebind("networkremote",this);
    }

    @Override
    public boolean addpnetwork(managment.model.TO.NetworkUtility obj) throws
Exception {
        managment.model.TO.NetworkUtility networkUtility=obj;
    }
}

```

```

        networkManager networkManager=new networkManager();
        boolean ans=networkManager.addnetwor(networkUtility);
        return ans;
    }

    @Override
    public boolean delenetwork(int id) throws Exception {
        int id1=id;
        networkManager networkManager=new networkManager();
        boolean ans=networkManager.deletenetwor(id1);
        return ans;
    }

    @Override
    public boolean updatenetwork(int id, String motherb) throws Exception {
        int id1=id;
        String s=motherb;
        networkManager networkManager=new networkManager();
        boolean ans=networkManager.updatenetwor(id1,s);
        return ans;
    }

    @Override
    public ArrayList<managment.model.TO.NetworkUtility> selectnetwork(int id)
    throws Exception {

        networkManager networkManager=new networkManager();
        ArrayList<managment.model.TO.NetworkUtility>
obj=networkManager.selectnetwor(id);
        return obj;
    }
}

```

```

package managment.RmiServer.RmiImplimentation;

import managment.RmiServer.Remote.PersonLoginAddconsol;
import managment.model.BL.consolManager;
import managment.model.BL.userManager;
import managment.view.ServerPage;

import javax.swing.text.Document;
import java.rmi.server.UnicastRemoteObject;

/**
 * Created by saeedtavana on 7/1/16.
 */
public class personalLoginAddconsoleImp extends UnicastRemoteObject implements
PersonLoginAddconsol {
    public personalLoginAddconsoleImp() throws Exception
    {
        java.rmi.Naming.rebind("addconsole",this);
    }
    @Override
    public void addconsole(String s) throws Exception {

        /*String s1=s;
        String s2=ServerPage.INFO.getText();
        ServerPage.INFO.setText("");
        ServerPage.INFO.setText(s2+s1);*/
        Document d=ServerPage.INFO.getDocument();
        d.insertString(d.getLength(),s,null);
        d.insertString(d.getLength(),"\n",null);
        //ServerPage.INFO.setText(s);
        // function to add to database
        consolManager consolManager=new consolManager();
        consolManager.addconsol(s);
    }
}

```

```

    }

    @Override
    public boolean checkuser(String user, String pass) throws Exception {
        String u=user;
        String pa=pass;
        boolean ans=false;

        userManager userManager=new userManager();
        ans=userManager.checkuser2(u,pa);
        return ans;
    }

    public static boolean checkadmin(String user, String pass) throws Exception {
        String u=user;
        String pa=pass;

        userManager userManager=new userManager();
        boolean ans=userManager.checkadmin(u,pa);
        return ans;
    }
}

```

```

package managment.RmiServer.RmiImplimentation;

import managment.RmiServer.Remote.Printer;
import managment.model.BL.printermanager;
import managment.model.TO.Monitor;

import java.rmi.server.UnicastRemoteObject;
import java.util.ArrayList;

/**
 * Created by saeedtavana on 7/1/16.
 */
public class printerImp extends UnicastRemoteObject implements Printer {

    public printerImp() throws Exception
    {
        java.rmi.Naming.rebind("printerremote",this);
    }

    @Override
    public boolean addprinter(managment.model.TO.Printer obj) throws Exception {
        managment.model.TO.Printer printer=obj;
        printermanager printermanager=new printermanager();
        boolean ans=printermanager.addprinter(printer);
        return ans;
    }

    @Override
    public boolean deleprinter(int id) throws Exception {
        int idl=id;
        printermanager printermanager=new printermanager();
        boolean ans=printermanager.deleteprinter(idl);
        return ans;
    }

    @Override
    public boolean updatprinter(int id, String motherb) throws Exception {
        int idl=id;
        String m=motherb;
        printermanager printermanager=new printermanager();
    }
}

```

```

        boolean ans=printermanager.updateprinter(id1,m);
        return ans;
    }

    @Override
    public ArrayList<managment.model.TO.Printer> selectprinter(int id) throws
Exception {
        int id1=id;
        printermanager printermanager=new printermanager();
        ArrayList<managment.model.TO.Printer> obj=printermanager.selectprinte(id1);
        return obj;
    }
}

```

```

package managment.RmiServer.RmiImplimentation;

import managment.view.ServerPage;

import javax.swing.text.BadLocationException;
import javax.swing.text.Document;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.text.SimpleDateFormat;
import java.util.Date;

/**
 * Created by user on 5/13/2016.
 */
public class RmiserverControl {

    public Date d=new Date();
    public SimpleDateFormat simpleDateFormat=new SimpleDateFormat();
    public PreparedStatement preparedStatement1;
    public Connection connection;
    private RmiserverControl()
    {

    }

    private static RmiserverControl obj=new RmiserverControl();
    public static RmiserverControl getInstance()
    {
        return obj;
    }

    public computerImp computerImp;
    public monitorImp monitorImp;
    public printerImp printerImp;
    public networkUtilityImp networkUtilityImp;
    public personalLoginAddconsoleImp personalLoginAddconsoleImp;
    public scannerImp scannerImp;
    public roomImp roomImp;
    public equimpentMaintenanceImp equimpentMaintenanceImp;
    public void start() throws BadLocationException {
        try {

            /* Document d= ServerPage.INFO.getDocument();
            String s="the server is started";
            d.insertString(d.getLength(),s,null);*/

            java.rmi.registry.LocateRegistry.createRegistry(1099);

            monitorImp=new monitorImp();
            printerImp=new printerImp();

```

```

        networkUtilityImp=new networkUtilityImp();
        personalLoginAddconsoleImp=new personalLoginAddconsoleImp();
        scannerImp=new scannerImp();
        roomImp=new roomImp();
        equipmentMaintenanceImp=new equipmentMaintenanceImp();
        computerImp=new computerImp();
        // personalLoginAddconsoleImp personalLoginAddconsoleImp=new
personalLoginAddconsoleImp();
        // personalLoginAddconsoleImp.addconsole("the server is up on :
"+d.toString()+" ...");
        Class.forName("oracle.jdbc.driver.OracleDriver");
        connection =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl","managment","13
74");

        connection.setAutoCommit(false);
        String date=simpleDateFormat.format(d)+" ";
        preparedStatement1=connection.prepareStatement("INSERT INTO console
VALUES (?,?)");
        preparedStatement1.setString(1,date);
        preparedStatement1.setString(2,"the server is up on : "+d.toString()+"
...");

        int res=preparedStatement1.executeUpdate();
        connection.commit();
        preparedStatement1.close();
        connection.close();
        ServerPage.INFO.setText("the server is up on : "+d.toString()+"
...");

    }
    catch (Exception e)
    {
        ServerPage.INFO.setText("the server can not be start");
        // Document dl= ServerPage.INFO.getDocument();
        //String s="the server can not be start";
        //dl.insertString(dl.getLength(),s,null);
        // e.printStackTrace();
    }

}

public void stop()
{
    try
    {
        ServerPage.INFO.setText("the server is stopped");

        java.rmi.Naming.unbind("computerremote");
        java.rmi.Naming.unbind("monitorremote");
        java.rmi.Naming.unbind("addconsole");
        java.rmi.Naming.unbind("printerremote");
        java.rmi.Naming.unbind("networkremote");
        java.rmi.Naming.unbind("scannerremote");
        java.rmi.Naming.unbind("roomremote");
        java.rmi.Naming.unbind("equipmentremote");

        UnicastRemoteObject.unexportObject(computerImp,true);
        UnicastRemoteObject.unexportObject(monitorImp,true);
        UnicastRemoteObject.unexportObject(networkUtilityImp,true);
        UnicastRemoteObject.unexportObject(personalLoginAddconsoleImp,true);
        UnicastRemoteObject.unexportObject(printerImp,true);
        UnicastRemoteObject.unexportObject(scannerImp,true);
        UnicastRemoteObject.unexportObject(roomImp,true);
        UnicastRemoteObject.unexportObject(equipementMaintenanceImp,true);

    }
    catch (Exception e)

```

```

        {

        }

    }
}
package managment.RmiServer.RmiImplimentation;

import managment.RmiServer.Remote.room;
import managment.model.BL.roomManager;

import java.rmi.server.UnicastRemoteObject;
import java.util.ArrayList;

/**
 * Created by saeedtavana on 7/3/16.
 */
public class roomImp extends UnicastRemoteObject implements room {
    public roomImp() throws Exception
    {
        java.rmi.Naming.rebind("roomremote",this);
    }
    @Override
    public boolean addroom(managment.model.TO.room obj) throws Exception {
        managment.model.TO.room room=obj;
        roomManager r=new roomManager();
        boolean ans=r.addroom(room);
        return ans;
    }

    @Override
    public boolean deleteroom(int id) throws Exception {
        int idl=id;
        roomManager roomManager=new roomManager();
        boolean ans=roomManager.deleteroom(idl);
        return ans;
    }

    @Override
    public boolean updateroom(int id, int telephone) throws Exception {
        int idl=id;
        int tell=telephone;
        roomManager roomManager=new roomManager();
        boolean ans=roomManager.updateroom(idl,tell);
        return ans;
    }

    @Override
    public ArrayList<managment.model.TO.room> selectroom(int id) throws Exception {
        int idl=id;
        roomManager roomManager=new roomManager();
        ArrayList<managment.model.TO.room> obj=roomManager.selectrooms(idl);
        return obj;
    }
}

```

```

package managment.RmiServer.RmiImplimentation;

import managment.RmiServer.Remote.Scanner;
import managment.model.BL.scannerManager;

import java.rmi.server.UnicastRemoteObject;
import java.util.ArrayList;

```

```

/**
 * Created by saeedtavana on 7/1/16.
 */
public class scannerImp extends UnicastRemoteObject implements Scanner {
    public scannerImp() throws Exception
    {
        java.rmi.Naming.rebind("scannerremote",this);
    }
    @Override
    public boolean addscanneer(managment.model.TO.Scanner obj) throws Exception {
        managment.model.TO.Scanner scanner=obj;
        scannerManager scannerManager=new scannerManager();
        boolean ans=scannerManager.addscanner(scanner);
        //function
        return ans;
    }

    @Override
    public boolean delescanner(int id) throws Exception {
        int idl=id;
        scannerManager scannerManager=new scannerManager();
        boolean ans=scannerManager.deletescanner(idl);
        //function
        return ans;
    }

    @Override
    public boolean updatscanner(int id, String motherb) throws Exception {
        int idl=id;
        String m=motherb; //model of scanner
        scannerManager scannerManager=new scannerManager();
        boolean ans=scannerManager.updatscanner(idl,m);
        //function
        return ans;
    }

    @Override
    public ArrayList<managment.model.TO.Scanner> selectscanner(int id) throws
Exception {
        int idl=id;
        scannerManager scannerManager=new scannerManager();
        ArrayList<managment.model.TO.Scanner> obj=scannerManager.selectscanner(idl);
        //function
        return null;
    }
}

```

## server view

```

package managment.view;

import javax.swing.*;
import java.awt.*;
import java.awt.geom.Line2D;

/**
 * Created by user on 5/25/2016.
 */
public class helpfourm extends JApplet{
    public static JFrame helppage=new JFrame("help page");
    public static JLabel line =new JLabel();
    public static JLabel text=new JLabel();
    public JScrollPane s=new JScrollPane();
    public static JLabel text1=new JLabel();
    public static JLabel text2=new JLabel();
    public JLabel tex3=new JLabel();
}

```



```

public JLabel text4=new JLabel();
public JLabel text5=new JLabel();
public JLabel text6=new JLabel();
public JLabel text7=new JLabel();
public JLabel text8=new JLabel();
public void drawline(Graphics2D g)
{
    Graphics2D g2=(Graphics2D)g;
    g2.draw(new Line2D.Double(50,150,250,30));
}

public void show1()
{
    JPanel jPanel=new JPanel(null);
    jPanel.setComponentOrientation(ComponentOrientation.RIGHT_TO_LEFT);
    Font titlefont = new Font("",Font.BOLD,25);
    helppage.setComponentOrientation(ComponentOrientation.RIGHT_TO_LEFT);
    text.setComponentOrientation(ComponentOrientation.RIGHT_TO_LEFT);
    helppage.setBackground(Color.white);
    helppage.setBounds(1500,200,800,1200);

    ///////////
    jPanel.add(text);
    jPanel.add(text1);
    jPanel.add(s);
    jPanel.add(text2);
    jPanel.add(line);
    jPanel.add(tex3);
    jPanel.add(text4);
    jPanel.add(text5);
    jPanel.add(text6);
    jPanel.add(text7);
    jPanel.add(text8);
    s.setVisible(true);
    //////////////////////////////////////////jPanel.add(text);////////////////////////////////////
    text.setText("( Help page )");
    text.setFont(titlefont);
    text.setBounds(320,10,200,30);
    text.setForeground(Color.blue);
    text1.setComponentOrientation(ComponentOrientation.RIGHT_TO_LEFT);
    text1.setBounds(0,80,700,50);
    text1.setText(": HELP");
    text1.setFont(titlefont);
    text1.setForeground(Color.BLACK);
    line.setBounds(0,50,770,20);
    line.setText("_____");

    line.setFont(new Font("",Font.BOLD,22));
    line.setBounds(25,50,750,30);
    text2.setComponentOrientation(ComponentOrientation.RIGHT_TO_LEFT);
    text2.setBounds(50,120,600,50);
    text2.setText("-شود می برده کار به دستورات از لیستی ی مشاهده برای -");
    Font f=new Font("tahoma",Font.ITALIC,20);
    text2.setFont(f);
    tex3.setBounds(0,200,685,50);
    tex3.setFont(titlefont);
    tex3.setComponentOrientation(ComponentOrientation.RIGHT_TO_LEFT);
    tex3.setText(": help");
    text4.setFont(f);
    text4.setComponentOrientation(ComponentOrientation.RIGHT_TO_LEFT);
    text4.setBounds(50,250,600,50);
    text4.setText("-شود می برده کار به دستورات از لیستی ی مشاهده برای -");
    text5.setFont(titlefont);
    text5.setComponentOrientation(ComponentOrientation.RIGHT_TO_LEFT);
    text5.setText(": Help");
    text5.setBounds(0,330,690,50);
    text6.setComponentOrientation(ComponentOrientation.RIGHT_TO_LEFT);

```

```

text6.setFont (f);
text6.setText ("-شود می برده کار به دستورات از لیستی مشاهده برای -");
text6.setBounds (50,400,600,50);
text7.setComponentOrientation(ComponentOrientation.RIGHT_TO_LEFT);
text7.setFont (titleFont);
text7.setText (": ؟ ");
text7.setBounds (0,450,690,50);
text8.setFont (f);
text8.setComponentOrientation(ComponentOrientation.RIGHT_TO_LEFT);
text8.setBounds (50,520,600,50);
text8.setText ("-شود می برده کار به دستورات از لیستی مشاهده برای -");
helpfourm g=new helpfourm();
helppage.add(g);
////////////////////////////////////
helppage.setComponentOrientation(ComponentOrientation.LEFT_TO_RIGHT);
text.setComponentOrientation(ComponentOrientation.LEFT_TO_RIGHT);
helppage.setBackground(Color.white);
helppage.setBounds (1500,200,800,1200);
helppage.add(text);
helppage.add(text1);
helppage.add(text2);
helppage.add(line);
helppage.add(tex3);
helppage.add(text4);
helppage.add(text5);
helppage.add(text6);
helppage.add(text7);
helppage.add(jPanel);
jPanel.add(new helpfourm() );
helppage.setResizable(false);
//helppage.setLocationRelativeTo(ServerPage.serverpage);
helppage.setDefaultCloseOperation (JFrame.HIDE_ON_CLOSE);
helppage.setVisible(true);

helppage.setLayout (null);

}
}

```

```

package managment.view;

import managment.controller.logincontroller.button1loginController;
import managment.controller.logincontroller.button2longController;
import managment.controller.logincontroller.loginController;

import javax.swing.*;
import java.awt.*;
import java.awt.Color;

/**
 * Created by saeedtavana on 5/8/16.
 */
public class login {

    public JFrame jFrame=new JFrame("Log in page");
    public JButton jButton1=new JButton("Enter");
    public JButton jButton2=new JButton("clear");
    public JLabel jLabel1=new JLabel ("passwprd :");
    public JLabel jLabel2=new JLabel ("username :");
    public JPasswordField jPasswordField=new JPasswordField();
    public JTextField jTextField=new JFormattedTextField();
    public JLabel jLabel3=new JLabel("Logged in username : ");
    public JTextField jTextField1=new JTextField();
    public GridBagConstraints gridBagConstraints=new GridBagConstraints();

```

```

public JPanel jPanel=new JPanel();
public loginController loginController=new loginController();
public button1loginController button1loginController=new button1loginController();
public button2longController button2longController=new button2longController();
public login() {
    super();
}

public void showLogin()
{
    Font f=new Font("",Font.BOLD,20);
    JFrame.setLayout(null);
    jPanel.add(jLabe3);
    jPanel.add(jTextField1);
    JFrame.add(jPanel);
    jPanel.setVisible(true);
    ///////////////////////////////////
    JFrame.setBounds(400,100,500,400);
    jLabe3.setBounds(20,20,200,30);
    jLabel2.setBounds(20,130,150,30);
    jLabel1.setBounds(20,180,150,30);
    jTextField1.setBounds(250,20,200,30);
    jTextField.setBounds(250,130,200,30);
    jPasswordField.setBounds(250,180,200,30);
    jButton1.setBounds(150,270,100,30);
    jButton2.setBounds(300,270,100,30);
    jButton1.setFont(f);
    jButton2.setFont(f);
    jLabel1.setFont(f);
    jLabel2.setFont(f);
    jLabe3.setFont(f);
    ///////////////////////////////////
    jButton2.addActionListener(loginController);
    jButton2.setActionCommand("clear button");
    jButton1.setActionCommand("enter button");
    jButton1.addActionListener(loginController);
    jButton1.addKeyListener(button1loginController);
    jButton2.addKeyListener(button2longController);
    ///////////////////////////////////
    JFrame.add(jLabe3);
    JFrame.add(jTextField1);
    JFrame.add(jLabel2);
    JFrame.add(jLabel1);
    JFrame.add(jPasswordField);
    JFrame.add(jTextField);
    JFrame.add(jButton1);
    JFrame.add(jButton2);

    ///////////////////////////////////
    jTextField1.setVisible(true);
    jLabe3.setVisible(true);
    jButton1.setVisible(true);
    jButton2.setVisible(true);
    jLabel1.setVisible(true);
    jLabel2.setVisible(true);
    jPasswordField.setVisible(true);
    jTextField.setVisible(true);
    ///////////////////////////////////
    JFrame.setBounds(400,100,500,400);
    JFrame.setLocationRelativeTo(null);
    JFrame.setComponentOrientation(ComponentOrientation.RIGHT_TO_LEFT);
    JFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    JFrame.getContentPane().setBackground(Color.LIGHT_GRAY);
    jTextField1.setEnabled(false);
    JFrame.setResizable(false);
    JFrame.setVisible(true);
    loginController.login1=this;
    button2longController.login2=this;
}

```

```

        buttonloginController.login3=this;

    }
}

```

```

package managment.view;

import com.sun.org.apache.bcel.internal.generic.NEW;
import com.sun.org.apache.xpath.internal.operations.String;
import com.sun.prism.paint.*;
import managment.controller.ServerConroler.exitserverbutton;
import managment.controller.ServerConroler.startserverbutton;
import managment.controller.ServerConroler.stopserverbutton;
import managment.controller.ServerConroler.sysfielscontroller;
import managment.controller.logincontroller.loginController;

import javax.swing.*;
import java.awt.*;
import java.awt.Color;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;

/**
 * Created by saeedtavana on 5/9/16.
 */
public class ServerPage {
    public static JFrame serverpage =new JFrame("Server page (main control)");
    public static JTextField loggedname=new JTextField();
    public static JButton startserver1=new JButton("start server");
    public static JButton endserver=new JButton("stop server");
    public static JLabel time=new JLabel("current date :");
    public static JTextPane INFO= new JTextPane();
    public static JLabel NAME=new JLabel("Logged user name :");
    public static JButton EXIT =new JButton("EXIT SERVER") ;
    public static JButton consol=new JButton("server information");
    public static JTextField time1=new JTextField();
    public JScrollPane js=new JScrollPane(INFO);
    public exitserverbutton exit=new exitserverbutton();
    public stopserverbutton stop=new stopserverbutton();
    public startserverbutton start=new startserverbutton();
    public static JTextField sys=new JTextField();
    public static JLabel command =new JLabel("command :");
    public static JTextField seconds=new JTextField();
    public Date date=new Date();
    public DateFormat format=new SimpleDateFormat("YYYY/MM/dd");
    public sysfielscontroller sysfielscontroller=new sysfielscontroller();
    public void showServerPage()
    {

        Font f=new Font("",Font.BOLD,23);
        serverpage.setLayout(new GridLayout(2,1));
        JPanel jPanellinfo=new JPanel(null);
        jPanellinfo.setLayout(null);
        JPanel console=new JPanel(new GridLayout());

        console.setBorder(BorderFactory.createEmptyBorder(30,30,30,30));
        jPanellinfo.setBorder(BorderFactory.createEmptyBorder(30,30,10,30));

        //////////////////////////////////////
        EXIT.setActionCommand("exitserver");
        EXIT.addActionListener(exit);
        EXIT.addKeyListener(exit);
        endserver.setActionCommand("stopserver");
        endserver.addActionListener(stop);
    }
}

```

```

        endserver.addKeyListener(stop);
        startserver1.setActionCommand("startserver");
        startserver1.addActionListener(start);
        startserver1.addKeyListener(start);
        NAME.setFont(new Font("",Font.BOLD,25));
        NAME.setBounds(80,60,250,40);
        loggedname.setBounds(400,60,400,60);

        loggedname.setEnabled(false);
        loggedname.setFont(f);
        //////////////////////////////////

        //////////////////////////////////

        startserver1.setBounds(600,300,300,150);
        startserver1.setBackground(Color.LIGHT_GRAY);
        endserver.setBounds(1200,300,300,150);
        endserver.setBackground(Color.PINK);
        endserver.setFont(f);
        startserver1.setFont(f);
        EXIT.setBounds(1800,300,300,150);
        EXIT.setFont(f);
        EXIT.setBackground(Color.red);
        consol.setBounds(35,635,2480,100);
        consol.setFont(new Font("",Font.BOLD,30));
        consol.setEnabled(true);
        consol.setForeground(Color.white);
        consol.setBackground(Color.LIGHT_GRAY);
        time.setBounds(1200,60,200,50);
        time.setFont(f);
        timel.setBounds(1400,60,350,50);
        timel.setEnabled(false);
        timel.setFont(f);
        timel.setText(format.format(date));
        sys.addActionListener(sysfielscontroler);
        sys.addKeyListener(sysfielscontroler);
        sys.setActionCommand("sys");
        sys.setFont(new Font("",Font.BOLD,20));
        sys.setBounds(2000,60,400,60);
        seconds.setBounds(2420,60,90,60);
        seconds.setFont(f);
        command.setFont(f);
        command.setBounds(1850,60,130,50);
        //////////////////////////////////////////
        jPanellinfo.add(NAME);
        jPanellinfo.add(time);
        jPanellinfo.add(timel);
        jPanellinfo.add(loggedname);
        jPanellinfo.add(endserver);
        jPanellinfo.add(startserver1);
        jPanellinfo.add(EXIT);
        jPanellinfo.add(sys);
        jPanellinfo.add(consol);
        jPanellinfo.add(command);
        jPanellinfo.add(seconds);
        jPanellinfo.setVisible(true);
        console.add(js);
        console.setVisible(true);
        //////////////////////////////////////////
        //////////////////////////////////////////

        INFO.setBackground(Color.BLACK);
        INFO.setForeground(Color.white);
        INFO.setFont(new Font("",Font.BOLD,25));
        serverpage.add(jPanellinfo);
        serverpage.add(console);
        serverpage.pack();

```

```

        loggedname.setVisible(true);
        serverpage.setVisible(true);
        serverpage.setResizable(true);
        serverpage.setExtendedState(Frame.MAXIMIZED_BOTH);
        serverpage.setLocationRelativeTo(null);

        serverpage.setDefaultCloseOperation(serverpage.EXIT_ON_CLOSE);

    }
}

```

## client

```

package start;

import classes.viewmanager.*;
import managment.RmiServer.Remote.PersonLoginAddconsol;

import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Scanner;

/**
 * Created by saeedtavana on 7/1/16.
 */
public class main {

    public static String try_again()
    {
        Scanner scanner=new Scanner(System.in);
        System.out.println(" do you want to try again [y/n] ? :");
        String ans=scanner.nextLine();
        return ans;
    }

    public Date d=new Date();
    public SimpleDateFormat simpleDateFormat=new SimpleDateFormat("YYYY/MM/dd");
    public static String username;

    public static void select_menu()
    {
        String s;
        int result;
        Scanner scanner=new Scanner(System.in);
        do {

            System.out.println("***** managment system
*****");
            System.out.println();
            System.out.println("[1] manage computer ");
            System.out.println("[2] manage monitor ");
            System.out.println("[3] manage printer ");
            System.out.println("[4] manage scanner ");
            System.out.println("[5] manage network unility ");
            System.out.println("[6] manage rooms ");
            System.out.println("[7] manage equipment maintenance ");
            System.out.println("[8] Exit ");
            System.out.print("yout asnwer : ");
            result = scanner.nextInt();

```

```

        switch (result) {
            case 1:
                Computer.computer_select_menu();

                break;

            case 2:
                Monitor.monitor_select_menu();
                break;

            case 3:
                Printer.printer_select_menu();
                break;

            case 4:
                Scanner.scanner_select_menu();
                break;

            case 5:
                NetworkUtility.networkutility_select_menu();
                break;

            case 6:
                Room.room_select_menu();
                break;

            case 7:
                Equipment.equipment_select_menu();
                break;

            case 8:
                try {
                    main m=new main();
                    String date5= m.simpleDateFormat.format(m.d)+" ";
                    PersonLoginAddconsol
per=(PersonLoginAddconsol) java.rmi.Naming.lookup("addconsole");
                    per.addconsole("the user wiht name : "+main.username+"
is logged off on : "+m.d.toString()+" ");
                    System.out.println("the user wiht name :
"+main.username+" is logged off on : "+ m.d.toString()+"");
                    System.exit(0);

                }
                catch (Exception e)
                {
                    System.out.println("the server is down ..");
                    System.exit(0);
                }

                break;
            default:
                System.out.println("wrong input");
                break;

        }

        s = try_again();

    }

    while (s.equals("y") || s.equals("Y")) ;

}

public static void main(String[] args) {

    int count=0;

```

```

do {

    Scanner scanner = new Scanner(System.in);
    System.out.print("please enter the user name : ");
    String uname = scanner.nextLine();
    main.username = uname;
    System.out.print("please enter the password :");
    String pass = scanner.nextLine();
    ///rmi function to check
    try {

        PersonLoginAddconsol per = (PersonLoginAddconsol)
java.rmi.Naming.lookup("addconsole");
        boolean b = per.checkuser(uname, pass);
        if (b == true) {
            main m = new main();
            String date5 = m.simpleDateFormat.format(m.d) + "";
            System.out.println();
            System.out.println();
            per.addconsole("the user with name : " + uname + " is logged
in on : " + m.d.toString() + "");
            System.out.println("the user name : " + uname + " is logged in
on : " + m.d.toString() + "");
            System.out.println();
            select_menu();
        } else if (b == false) {
            try {

                main m = new main();
                PersonLoginAddconsol perl = (PersonLoginAddconsol)
java.rmi.Naming.lookup("addconsole");
                perl.addconsole("the user entered wrong username or
password on : " + m.d.toString() + "");
                System.out.println("the username or password is
incorrect");
                count++;
            } catch (Exception e) {

            }

        }
    } catch (Exception e) {

        System.out.println("the server is down ...");
    }
    System.out.println();
    System.out.println();

}
while (count<3 && count>=0);
System.out.println("3 times you entered wrong username or password please
try again later ... ");
}
}

```

```

package managment;

import managment.model.TO.*;

/**
 * Created by saeedtavana on 7/3/16.
 */
public class factory {

```



```

public static device mackobj(String s )
{
    if (s.equals("computer"))
    {
        return new Computer();
    }
    else if (s.equals("consol"))
    {
        return new consol();
    }
    else if (s.equals("equipment"))
    {
        return new equipmentMainteneace();
    }
    else if (s.equals("monitor"))
    {
        return new Monitor();
    }
    else if (s.equals("network"))
    {
        return new NetworkUtility();
    }
    else if (s.equals("printer"))
    {
        return new Printer();
    }
    else if (s.equals("room"))
    {
        return new room();
    }
    else if (s.equals("scaner"))
    {
        return new Scanner();
    }
    else
    {
        return null;
    }
}
}

```

```

package managment.model.TO;

import java.io.Serializable;

/**
 * Created by saeedtavana on 6/5/16.
 */
public class Computer implements Serializable ,device{

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }
}

```

```

public String getUser() {
    return uuser;
}

public void setUser(String uuser) {
    this.uuser = uuser;
}

public int getRoomNumber() {
    return roomNumber;
}

public void setRoomNumber(int roomNumber) {
    this.roomNumber = roomNumber;
}

public String getSection() {
    return section;
}

public void setSection(String section) {
    this.section = section;
}

public String getMotherboardnumber() {
    return motherboardnumber;
}

public void setMotherboardnumber(String motherboardnumber) {
    this.motherboardnumber = motherboardnumber;
}

public String getDisknumber() {
    return disknumber;
}

public void setDisknumber(String disknumber) {
    this.disknumber = disknumber;
}

public int getNumberOfdisk() {
    return numberOfdisk;
}

public void setNumberOfdisk(int numberOfdisk) {
    this.numberOfdisk = numberOfdisk;
}

public int getCapacityOfDiskinG() {
    return capacityOfDiskinG;
}

public void setCapacityOfDiskinG(int capacityOfDiskinG) {
    this.capacityOfDiskinG = capacityOfDiskinG;
}

public int getRamInG() {
    return RamInG;
}

public void setRamInG(int ramInG) {
    RamInG = ramInG;
}

public String getCpu() {
    return cpu;
}

```

```

    public void setCpu(String cpu) {
        this.cpu = cpu;
    }

    public String getOs() {
        return Os;
    }

    public void setOs(String os) {
        Os = os;
    }

    public int getNumberOfUser() {
        return numberOfUser;
    }

    public void setNumberOfUser(int numberOfUser) {
        this.numberOfUser = numberOfUser;
    }

    public String getComments() {
        return Comments;
    }

    public void setComments(String comments) {
        Comments = comments;
    }

    private int id;
    private String uuser;
    private int roomNumber;
    private String section;
    private String motherboardnumber;
    private String disknumber;
    private int numberOfdisk;
    private int capacityOfDiskinG;
    private int RamInG;
    private String cpu;
    private String Os;
    private int numberOfUser;
    private String Comments;

    public String getDate() {
        return date;
    }

    public void setDate(String date) {
        this.date = date;
    }

    private String date;
}

```

```

package managment.model.TO;

import java.io.Serializable;

/**
 * Created by saeedtavana on 7/1/16.
 */
public class consol implements Serializable,device {
    public String getDate() {

```

```

        return date;
    }

    public void setDate(String date) {
        this.date = date;
    }

    public String getComments() {
        return comments;
    }

    public void setComments(String comments) {
        this.comments = comments;
    }

    private String date;
    private String comments;
}

```

```

package managment.model.TO;

/**
 * Created by saeedtavana on 7/1/16.
 */
public interface device {
}
package managment.model.TO;

import java.io.Serializable;
import java.util.Scanner;

/**
 * Created by saeedtavana on 7/3/16.
 */
public class equipmentMainteneace implements device,Serializable {
    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getTetx() {
        return tetx;
    }

    public void setTetx(String tetx) {
        this.tetx = tetx;
    }

    public String getDate() {
        return date;
    }

    public void setDate(String date) {
        this.date = date;
    }

    private String date;
    private int id;
    private String tetx;
}

```

```

}
package managment.model.TO;

import java.io.Serializable;

/**
 * Created by saeedtavana on 6/5/16.
 */
public class Monitor implements Serializable,device {
    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getUser() {
        return uuser;
    }

    public void setUser(String uuser) {
        this.uuser = uuser;
    }

    public int getRoomNumber() {
        return roomNumber;
    }

    public void setRoomNumber(int roomNumber) {
        this.roomNumber = roomNumber;
    }

    public String getSection() {
        return section;
    }

    public void setSection(String section) {
        this.section = section;
    }

    public String getDate() {
        return date;
    }

    public void setDate(String date) {
        this.date = date;
    }

    public String getModel() {
        return model;
    }

    public void setModel(String model) {
        this.model = model;
    }

    public String getBrand() {
        return brand;
    }

    public void setBrand(String brand) {
        this.brand = brand;
    }

    public String getType() {
        return type;
    }

```

```

    }

    public void setType(String type) {
        this.type = type;
    }

    public String getCommand() {
        return command;
    }

    public void setCommand(String command) {
        this.command = command;
    }

    private int id;
    private String uuser;
    private int roomNumber;
    private String section;
    private String date;
    private String model;
    private String brand;
    private String type;
    private String command;
}
package managment.model.TO;

import java.io.Serializable;

/**
 * Created by saeedtavana on 6/5/16.
 */
public class NetworkUtility implements Serializable, device {
    public String getBrand() {
        return brand;
    }

    public void setBrand(String brand) {
        this.brand = brand;
    }

    public String getModel() {
        return model;
    }

    public void setModel(String model) {
        this.model = model;
    }

    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }

    public String getDate() {
        return date;
    }

    public void setDate(String date) {
        this.date = date;
    }

    public String getComments() {
        return comments;
    }
}

```

```

    public void setComments(String comments) {
        this.comments = comments;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public int id;
    private String brand;
    private String model;
    private String type;
    private String date;
    private String comments;
}
package managment.model.TO;

import java.io.Serializable;

/**
 * Created by saeedtavana on 6/5/16.
 */
public class Printer implements Serializable,device {
    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getUser() {
        return uuser;
    }

    public void setUser(String uuser) {
        this.uuser = uuser;
    }

    public int getRoomNumber() {
        return roomNumber;
    }

    public void setRoomNumber(int roomNumber) {
        this.roomNumber = roomNumber;
    }

    public String getSection() {
        return section;
    }

    public void setSection(String section) {
        this.section = section;
    }

    public String getDate() {
        return date;
    }

    public void setDate(String date) {
        this.date = date;
    }

    public String getModel() {

```

```

        return model;
    }

    public void setModel(String model) {
        this.model = model;
    }

    public String getBrand() {
        return brand;
    }

    public void setBrand(String brand) {
        this.brand = brand;
    }

    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }

    public String getCommand() {
        return command;
    }

    public void setCommand(String command) {
        this.command = command;
    }

    private int id;
    private String uuser;
    private int roomNumber;
    private String section;
    private String date;
    private String model;
    private String brand;
    private String type;
    private String command;
}
package managment.model.TO;

import java.io.Serializable;

/**
 * Created by saeedtavana on 7/3/16.
 */
public class room implements device ,Serializable {
    public int getNumber() {
        return number;
    }

    public void setNumber(int number) {
        this.number = number;
    }

    public int getTelephone() {
        return telephone;
    }

    public void setTelephone(int telephone) {
        this.telephone = telephone;
    }

    private int number;
    private int telephone;
}

```



```

package managment.model.TO;

import java.io.Serializable;

/**
 * Created by saeedtavana on 6/5/16.
 */
public class Scanner implements Serializable,device {
    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getUser() {
        return uuser;
    }

    public void setUser(String uuser) {
        this.uuser = uuser;
    }

    public int getRoomNumber() {
        return roomNumber;
    }

    public void setRoomNumber(int roomNumber) {
        this.roomNumber = roomNumber;
    }

    public String getSection() {
        return section;
    }

    public void setSection(String section) {
        this.section = section;
    }

    public String getDate() {
        return date;
    }

    public void setDate(String date) {
        this.date = date;
    }

    public String getModel() {
        return model;
    }

    public void setModel(String model) {
        this.model = model;
    }

    public String getBrand() {
        return brand;
    }

    public void setBrand(String brand) {
        this.brand = brand;
    }

    public String getType() {
        return type;
    }
}

```

```

    public void setType(String type) {
        this.type = type;
    }

    public String getCommand() {
        return command;
    }

    public void setCommand(String command) {
        this.command = command;
    }

    private int id;
    private String uuser;
    private int roomNumber;
    private String section;
    private String date;
    private String model;
    private String brand;
    private String type;
    private String command;
}

package managment.RmiServer.Remote;

import java.rmi.Remote;
import java.util.ArrayList;

/**
 * Created by user on 5/12/2016.
 */
public interface Computer extends Remote {
    public boolean addComputer(managment.model.TO.Computer obj) throws Exception;
    public boolean delecomputer(int id) throws Exception;
    public boolean updatecomputer(int id, String motherb) throws Exception;
    public ArrayList<managment.model.TO.Computer> selectcomputer(int id) throws
Exception;
    public int show(int id) throws Exception;
}

package managment.RmiServer.Remote;

import managment.model.TO.equipmentMainteneace;

import java.rmi.Remote;
import java.util.ArrayList;

/**
 * Created by saeedtavana on 7/3/16.
 */
public interface equipmentMaintenace extends Remote{
    public boolean addmaintenance(equipmentMainteneace obj) throws Exception;
    public ArrayList<equipmentMainteneace> selectAllmaintenance() throws Exception;
    public int select_no_add_maintenance() throws Exception;
    public ArrayList<equipmentMainteneace> select_maintenance_of_device(int id)
throws Exception;
}

package managment.RmiServer.Remote;

import managment.model.TO.*;

import java.rmi.Remote;
import java.util.ArrayList;

/**
 * Created by user on 5/12/2016.
 */
public interface Monitor extends Remote {
    public boolean addmonitor(managment.model.TO.Monitor obj) throws Exception;

```

```

        public boolean delemonitor(int id)throws Exception;
        public boolean updatemonitor(int id, String model)throws Exception;
        public ArrayList<managment.model.TO.Monitor> selectmonitro(int id)throws
Exception;
    }
package managment.RmiServer.Remote;

import managment.model.TO.*;

import java.rmi.Remote;
import java.util.ArrayList;

/**
 * Created by user on 5/12/2016.
 */
public interface NetworkUtility extends Remote {
    public boolean addpnetwork(managment.model.TO.NetworkUtility obj) throws
Exception;
    public boolean delenetwork(int id)throws Exception;
    public boolean updatenetwork(int id, String motherb)throws Exception;
    public ArrayList<managment.model.TO.NetworkUtility> selectnetwork(int id)throws
Exception;
}
package managment.RmiServer.Remote;

import java.rmi.Remote;

/**
 * Created by user on 5/13/2016.
 */
public interface PersonLoginAddconsol extends Remote{
    public void addconsole(String s)throws Exception;
    public boolean checkuser(String user, String pass) throws Exception;

}
package managment.RmiServer.Remote;

import managment.model.TO.*;

import java.rmi.Remote;
import java.util.ArrayList;

/**
 * Created by user on 5/12/2016.
 */
public interface Printer extends Remote {
    public boolean addprinter(managment.model.TO.Printer obj) throws Exception;
    public boolean deleprinter(int id)throws Exception;
    public boolean updatprinter(int id, String motherb)throws Exception;
    public ArrayList<managment.model.TO.Printer> selectprinter(int id)throws
Exception;
}
package managment.RmiServer.Remote;

import java.rmi.Remote;
import java.util.ArrayList;

/**
 * Created by saeedtavana on 7/3/16.
 */
public interface room extends Remote {
    public boolean addroom(managment.model.TO.room obj) throws Exception;
    public boolean deleteroom(int id) throws Exception;
    public boolean updateroom(int id, int telephone)throws Exception;
    public ArrayList<managment.model.TO.room> selectroom(int id)throws Exception;
}

```

```

}
package managment.RmiServer.Remote;

import managment.model.TO.*;

import java.rmi.Remote;
import java.util.ArrayList;

/**
 * Created by user on 5/12/2016.
 */
public interface Scanner extends Remote {
    public boolean addscanneer(managment.model.TO.Scanner obj) throws Exception;
    public boolean delescanner(int id) throws Exception;
    public boolean updatscanner(int id, String motherb) throws Exception;
    public ArrayList<managment.model.TO.Scanner> selectscanner(int id) throws
Exception;
}

package classes.viewmanager;

import managment.RmiServer.Remote.PersonLoginAddconsol;
import managment.RmiServer.Remote.equipmentMaintenace;
import managment.factory;
import managment.model.TO.device;
import managment.model.TO.equipmentMainteneace;
import start.main;

import java.io.Serializable;
import java.text.DateFormat;
import java.text.FieldPosition;
import java.text.ParsePosition;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.Scanner;

/**
 * Created by saeedtavana on 6/5/16.
 */
public class Computer implements Serializable, deviceElectronic{

    public static String s;
    public Date d=new Date();
    public SimpleDateFormat simpleDateFormat=new SimpleDateFormat("YYYY/MM/dd");
    private static String try_again()
    {
        Scanner scanner=new Scanner(System.in);
        System.out.println(" do you want to try again [y/n] ? :");
        String ans=scanner.nextLine();
        return ans;
    }
    public Scanner scanner=new Scanner(System.in);
    @Override
    public void add() {
        /////////// create object and connect to rmi server

        managment.model.TO.Computer computer=new managment.model.TO.Computer();

        System.out.print("please enter the id : ");
        computer.setId(scanner.nextInt());
        if (computer.getId()==0)
        {
            System.out.println("id can not be empty or 0 ");
        }
        System.out.print("please enter the user : ");

```

```

computer.setUser(scanner.next());
System.out.print("please enter the room number : ");
computer.setRoomNumber(scanner.nextInt());
System.out.print("please enter the secction : ");
computer.setSection(scanner.next());
//
System.out.print("please enter the motherboard number : ");
computer.setMotherboardnumber(scanner.next());
System.out.print("please enter the disk number : ");
computer.setDisknumber(scanner.next());
System.out.print("please enter the number of disk : ");
computer.setNumberOfdisk(scanner.nextInt());
System.out.print("please enter the capacity of disk : ");
computer.setCapacityOfDiskinG(scanner.nextInt());
System.out.print("please enter the capacity of ram [GIG] : ");
computer.setRamInG(scanner.nextInt());
System.out.print("please enter the capacity of cpu : ");
computer.setCpu(scanner.next());
System.out.print("please enter the os : ");
computer.setOs(scanner.next());
System.out.print("please enter the number of user : ");
computer.setNumberOfUser(scanner.nextInt());
System.out.print("please enter the comment : ");
computer.setComments(scanner.next());
String date2=simpleDateFormat.format(d)+"";
computer.setDate(date2);
System.out.println();
Scanner scanner2=new Scanner(System.in);
System.out.println(" do you want to add [y/n] ? :");
String ans=scanner2.nextLine();

if (ans.equals("Y")||ans.equals("y")) {

    try {

        PersonLoginAddconsol
per=(PersonLoginAddconsol) java.rmi.Naming.lookup("addconsole");

        managment.RmiServer.Remote.Computer
computer1=(managment.RmiServer.Remote.Computer)
            java.rmi.Naming.lookup("computerremote");
        boolean b= computer1.addComputer(computer);
        if (b==true)
        {
            per.addconsole("the computer device with id
"+computer.getId()+" is added ");
            System.out.println("the device add sucseessfull");
        }
        else if (b==false)
        {
            System.out.println("the device not added ");
        }

    }
    catch (Exception e)
    {
        System.out.println("the server is down ...");
    }
}
else
{
}

}

```

```

@Override
public void delete() {
    ////////// create object and connect to rmi server
    System.out.println("please enter the id of computer you want to delete");
    int idd=scanner.nextInt();

    try {
        PersonLoginAddconsol
per=(PersonLoginAddconsol) java.rmi.Naming.lookup("addconsole");
        managment.RmiServer.Remote.Computer
computer1=(managment.RmiServer.Remote.Computer)
            java.rmi.Naming.lookup("computerremote");
        boolean bb=computer1.delecomputer(idd);
        if (bb==true)
        {
            per.addconsole("the computer device with id "+idd+" is deleted ");
            System.out.println("the device delete sucessfull");
        }
        else if (bb==false)
        {
            System.out.println("the device not deleted ");
        }
    }
    catch (Exception e)
    {
        System.out.println("the server is down ...");
    }
}

@Override
public void select() {
    ////////// create object and connect to rmi server
    System.out.print("please enter the id : ");
    int id=scanner.nextInt();
    System.out.println("*****");
    try {
        PersonLoginAddconsol
per=(PersonLoginAddconsol) java.rmi.Naming.lookup("addconsole");
        managment.RmiServer.Remote.Computer
computer1=(managment.RmiServer.Remote.Computer)
            java.rmi.Naming.lookup("computerremote");
        ArrayList<managment.model.TO.Computer> bb=computer1.selectcomputer(id);
        if (bb==null)
        {
            System.out.println("no computer device found ! ");
        }
        else
        {
            per.addconsole("select computer device");
            for (managment.model.TO.Computer c:bb)
            {
                System.out.println("
                System.out.print("id : ");
                System.out.println(c.getId());
                System.out.print("user : ");
                System.out.println(c.getUser());
                System.out.print("room number :");
                System.out.println(c.getRoomNumber());
                System.out.print("section :");
                System.out.println(c.getSection());
                System.out.print("motherboard no :");
                System.out.println(c.getMotherboardnumber());
                System.out.print("disk no :");

```

```

        System.out.println(c.getDisknumber());
        System.out.print("number of disk :");
        System.out.println(c.getNumberofdisk());

        System.out.print("number of disk :");
        System.out.println(c.getNumberofdisk());
        System.out.print("capacity of disk :");
        System.out.println(c.getCapacityOfDiskinG());
        System.out.print("RAM :");
        System.out.println(c.getRamInG());
        System.out.print("Cpu :");
        System.out.println(c.getCpu());
        System.out.print("Os :");
        System.out.println(c.getOs());
        System.out.print("number of user :");
        System.out.println(c.getNumberofUser());
        System.out.print("comments :");
        System.out.println(c.getComments());
        System.out.print("date :");
        System.out.println(c.getDate());
        System.out.println();
    }
}

}

}

catch (Exception e)
{
    System.out.println("the server is down ...");
}

}

@Override
public void update() {
    ////////// create object and connect to rmi server
    System.out.print("please enter the id of computer : ");
    int id5=scanner.nextInt();
    System.out.println("please enter the new motherboard number : ");
    String ss=scanner.next();
    try {
        PersonLoginAddconsol
per=(PersonLoginAddconsol) java.rmi.Naming.lookup("addconsole");
        managment.RmiServer.Remote.Computer
computer1=(managment.RmiServer.Remote.Computer)
            java.rmi.Naming.lookup("computerremote");
        boolean bb=computer1.updatecomputer(id5,ss);
        if (bb==true)
        {
            per.addconsole("the device with id "+id5+" is updated ");
            System.out.println("computer device update sucseessfully");
        }
        else if(bb==false)
        {
            System.out.println("computer not updated");
        }
    }
}

catch (Exception e)
{
    e.printStackTrace();
    System.out.println("server is down ...");
}

}

```

```

@Override
public void select_don_maintenance() {

    System.out.print("please enter the id of computer you want to see the
maintenance :");
    int id1=scanner.nextInt();
    try {

        PersonLoginAddconsole
per=(PersonLoginAddconsole) java.rmi.Naming.lookup("addconsole");
        equipmentMaintenace eq =(equipmentMaintenace)
java.rmi.Naming.lookup("equipmentremote");
        ArrayList<equipmentMainteneace> e=eq.select_maintenance_of_device(id1);
        if (e==null)
        {
            System.out.println("no maintenance for "+id1);
        }
        else
        {
            per.addconsole("select the maintenance of device "+id1);
            for (equipmentMainteneace eq1:e)
            {
                System.out.println("_____");
                System.out.print("id :");
                System.out.println(eq1.getId());
                System.out.print("date :");
                System.out.println(eq1.getDate());
                System.out.print("comment :");
                System.out.println(eq1.getTetx());
                System.out.println();
            }
        }
    }
    catch (Exception e)
    {
        e.printStackTrace();
        System.out.println("the server is down ...");
    }
}

public static void computer_select_menu()
{
    do {

        Computer computer = new Computer();
        Scanner scanner = new Scanner(System.in);
        System.out.println("_____ computer section
_____");
        System.out.println();
        System.out.println(" [1] add computer ");
        System.out.println(" [2] delete computer ");
        System.out.println(" [3] update computer ");
        System.out.println(" [4] select computer ");
        System.out.println(" [5] select don maintenance ");
        System.out.println(" [6] main page ");
        System.out.println("your answer : ");
        int result = scanner.nextInt();

        switch (result) {
            case 1:

                computer.add();

```



```

        break;

        case 2:
            computer.delete();
            break;

        case 3:
            computer.update();
            break;

        case 4:
            computer.select();
            break;
        case 5:
            computer.select_don_maintenance();
            break;
        case 6:
            main.select_menu();
            break;
        default:
            System.out.println("wrong input");
            break;
    }
    s = try_again();

    }
    while (s.equals("Y") || s.equals("y"));
    main.select_menu();

}

////////// view function //////////
}
package classes.viewmanager;

/**
 * Created by saeedtavana on 7/1/16.
 */
public interface deviceElectronic {

    public void add();
    public void delete();
    public void select();
    public void update();
    public void select_don_maintenance();

}

package classes.viewmanager;

import managment.RmiServer.Remote.PersonLoginAddconsol;
import managment.RmiServer.Remote.equipmentMaintenace;
import managment.model.TO.*;
import start.main;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.Scanner;

/**
 * Created by saeedtavana on 7/3/16.
 */
public class Equipment implements deviceElectronic {
    public Date d=new Date();
    public SimpleDateFormat simpleDateFormat=new SimpleDateFormat("YYYY/MM/dd");
    private static String try_again()
    {
        Scanner scanner=new Scanner(System.in);

```

```

        System.out.println(" do you want to try again [y/n] ? :");
        String ans=scanner.nextLine();
        return ans;

    }

    @Override
    public void add() {
        Scanner scanner=new Scanner(System.in);
        equipmentMainteneace eq=new equipmentMainteneace();
        System.out.print(" please enter the id of device you want to add
maintenance recorde :");
        eq.setId(scanner.nextInt());
        System.out.print("please enter your text :");
        eq.setTetx(scanner.next());
        String date=simpleDateFormat.format(d)+" ";
        eq.setDate(date);
        Scanner scanner2=new Scanner(System.in);
        System.out.println(" do you want to add [y/n] ? :");
        String ans=scanner2.nextLine();
        if (ans.equals("Y")||ans.equals("y")) {
            try {
                PersonLoginAddconsol
per=(PersonLoginAddconsol) java.rmi.Naming.lookup("addconsole");
                equipmentMaintenace equ=(equipmentMaintenace)
java.rmi.Naming.lookup("equipmentremote");
                boolean bb=equ.addmaintenance(eq);
                if (bb==true)
                {
                    per.addconsole("the maintenance is added");
                    System.out.println("the maintenance record is  added");

                }
                else if (bb==false)
                {
                    System.out.println("the maintenance record is failed to add");

                }

            }
            catch (Exception e)
            {
                System.out.println("the server is down ...");

            }

        }
        else
        {

        }

    }

    @Override
    public void delete() {

        // this must not working

    }

    @Override
    public void select() {
        //select all maintenance
        try {

            PersonLoginAddconsol
per=(PersonLoginAddconsol) java.rmi.Naming.lookup("addconsole");
            equipmentMaintenace equ = (equipmentMaintenace)

```

```

java.rmi.Naming.lookup("equipmentremote");
    ArrayList<equipmentMainteneace> eqobj = equ.selectAllmaintenance();
    if (eqobj==null) {
        System.out.println("there is no maintenance");
    }
    else
    {
        per.addconsole("select all maintenance");
        for (equipmentMainteneace e:eqobj) {
            System.out.println("_____");
            System.out.print("id : ");
            System.out.println(e.getId());
            System.out.print("date : ");
            System.out.println(e.getDate());
            System.out.print("text : ");
            System.out.println(e.getDate());
            System.out.println();
        }
    }
}
catch (Exception e)
{

}

}

@Override
public void update() {
    //this must not working
}

@Override
public void select_don_maintenance() {
    //this function must no work
}

public void select_no_of_maintenance()
{
    int no=0;
    try {
        PersonLoginAddconsol
per=(PersonLoginAddconsol) java.rmi.Naming.lookup("addconsole");
        equipmentMaintenace equ = (equipmentMaintenace)
java.rmi.Naming.lookup("equipmentremote");
        no = equ.select_no_add_maintenance();
        per.addconsole("select number of maintenance");
        System.out.println("there are [ "+no+" ] number of maintenance we have
in system");
    }
    catch (Exception e)
    {
        System.out.println("the server is down ....");
    }
}

public static void equipment_select_menu() {
    String s;
    do {

```

```

        Equipment equipment=new Equipment();
        Scanner scanner = new Scanner(System.in);
        System.out.println("_____ equipment section
        ");
        System.out.println();
        System.out.println(" [1] add maintenancee ");
        System.out.println(" [2] select all maintenancee ");
        System.out.println(" [3] select number of all maintenancee ");
        System.out.println(" [4] main page ");
        System.out.println("your answer : ");
        int result = scanner.nextInt();

        switch (result) {
            case 1:
                equipment.add();

                break;

            case 2:
                equipment.select();
                break;

            case 3:
                equipment.select no of maintenance();
                break;

            case 4:
                main.select_menu();
                break;
            default:
                System.out.println("wrong input");
                break;
        }
        s = try_again();

    }
    while (s.equals("y")|| s.equals("Y"));
    main.select_menu();

}

package classes.viewmanager;

import managment.RmiServer.Remote.PersonLoginAddconsol;
import managment.RmiServer.Remote.equipmentMaintenace;
import managment.model.TO.equipmentMainteneace;
import start.main;

import java.io.Serializable;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.Scanner;

/**
 * Created by saeedtavana on 6/5/16.
 */
public class Monitor implements Serializable, deviceElectronic {

    public Date d=new Date();
    public SimpleDateFormat simpleDateFormat=new SimpleDateFormat("YYYY/MM/dd");
    private static String try_again()
    {
        Scanner scanner=new Scanner(System.in);
        System.out.println(" do you want to try again [y/n] ? :");
        String ans=scanner.nextLine();
        return ans;
    }
}

```

```

@Override
public void add() {
    Scanner scanner1=new Scanner(System.in);
    managment.model.TO.Monitor scannerobj=new managment.model.TO.Monitor();
    System.out.print("please enter the id : ");
    scannerobj.setId(scanner1.nextInt());
    if (scannerobj.getId()==0)
    {
        System.out.println("id can not be empty or 0 ");
    }
    System.out.print("please enter the user : ");
    scannerobj.setUser(scanner1.next());
    System.out.print("please enter the room number : ");
    scannerobj.setRoomNumber(scanner1.nextInt());
    System.out.print("please enter the secction : ");
    scannerobj.setSection(scanner1.next());
    System.out.print("please enter the brand : ");
    scannerobj.setBrand(scanner1.next());
    System.out.print("please enter the modle : ");
    scannerobj.setModel(scanner1.next());
    System.out.print("please enter the type : ");
    scannerobj.setType(scanner1.next());
    System.out.print("please enter the comments : ");
    scannerobj.setCommand(scanner1.next());
    String date2=simpleDateFormat.format(d)+" ";
    scannerobj.setDate(date2);
    Scanner scanner2=new Scanner(System.in);
    System.out.println(" do you want to add [y/n] ? :");
    String ans=scanner2.nextLine();
    if (ans.equals("Y")||ans.equals("y")) {

        try {

            PersonLoginAddconsol
per=(PersonLoginAddconsol) java.rmi.Naming.lookup("addconsole");
            managment.RmiServer.Remote.Monitor
sc=(managment.RmiServer.Remote.Monitor)
                java.rmi.Naming.lookup("monotorremote");
            boolean bb=sc.addmonitor(scannerobj);
            if (bb==true)
            {
                per.addconsole("the monitor device is added");
                System.out.println("the device is added");
            }
            else if (bb==false)
            {
                System.out.println("the device is not added");
            }

        }
        catch (Exception e)
        {
            System.out.println("the serve is down ...");
        }

    }
    else
    {

    }

}

@Override

```

```

public void delete() {
    Scanner scanner1=new Scanner(System.in);
    System.out.print("please enter the id of monitor you want to delete :");
    int idd=scanner1.nextInt();
    if (idd==0)
    {
        System.out.println("the id must not be empty or 0 ");
    }
    else
    {
        try {
            PersonLoginAddconsol
per=(PersonLoginAddconsol) java.rmi.Naming.lookup("addconsole");
            managment.RmiServer.Remote.Monitor
sc=(managment.RmiServer.Remote.Monitor)
                java.rmi.Naming.lookup("monotorremote");
            boolean bb=sc.delemonitor(idd);
            if (bb==true)
            {
                per.addconsole("the monitor device is deleted");
                System.out.println("the device is deleted");
            }
            else if (bb==false)
            {
                System.out.println("the device is not deleted");
            }
        }
        catch (Exception e)
        {
            System.out.println("the server is down ...");
        }
    }
}

@Override
public void select() {
    Scanner scanner1=new Scanner(System.in);
    System.out.print("enter the id of monitor : ");
    int idd =scanner1.nextInt();
    if (idd==0)
    {
        System.out.println("the id must not emoty or 0");
    }
    else {
        try {
            PersonLoginAddconsol
per=(PersonLoginAddconsol) java.rmi.Naming.lookup("addconsole");
            managment.RmiServer.Remote.Monitor
sc=(managment.RmiServer.Remote.Monitor)
                java.rmi.Naming.lookup("monotorremote");

            ArrayList<managment.model.TO.Monitor> sobj=sc.selectmonitro(idd);
            if (sobj.equals(null))
            {
                System.out.println("there is no monitor found ");
            }
            else
            {
                per.addconsole("select monitor device");
                for (managment.model.TO.Monitor sca:sobj) {
                    System.out.println("

```

```

        System.out.print("id : ");
        System.out.println(sca.getId());
        System.out.print("user : ");
        System.out.println(sca.getUser());
        System.out.print("room number :");
        System.out.println(sca.getRoomNumber());
        System.out.print("section :");
        System.out.println(sca.getSection());
        System.out.print("model :");
        System.out.println(sca.getModel());
        System.out.print("brand :");
        System.out.println(sca.getBrand());
        System.out.print("type :");
        System.out.println(sca.getType());
        System.out.print("date :");
        System.out.println(sca.getDate());
        System.out.print("comments :");
        System.out.println(sca.getCommand());
    }
}

}

}
catch (Exception e)
{
    System.out.println("the server is down...");
}
}

}

@Override
public void update() {
    Scanner scanner1=new Scanner(System.in);
    System.out.println("please enter the id");
    int idd=scanner1.nextInt();
    System.out.println("please enter the new model ");
    String ss=scanner1.next();
    if (idd==0)
    {
        System.out.println("the id must not emoty or 0");
    }
    else
    {
        try {
            PersonLoginAddconsol
per=(PersonLoginAddconsol) java.rmi.Naming.lookup("addconsole");
            managment.RmiServer.Remote.Monitor
sc=(managment.RmiServer.Remote.Monitor)
                java.rmi.Naming.lookup("monotorremote");
            boolean bb=sc.updatemonitor(idd,ss);
            if (bb==true)
            {
                per.addconsole("the monitor device is updated");
                System.out.println("the device is updated");
            }
            else if (bb==false)
            {
                System.out.println("the device is not updated");
            }
        }
    }
    catch (Exception e)
    {
        System.out.println("the server is down ...");
    }
}

```

```

    }

    }

}

@Override
public void select_don_maintenance() {
    Scanner scanner=new Scanner(System.in);
    System.out.print("please enter the id of monitor you want to see the
maintenance :");
    int id1=scanner.nextInt();
    try {

        PersonLoginAddconsole
per=(PersonLoginAddconsole) java.rmi.Naming.lookup("addconsole");
        equipmentMaintenance eq =(equipmentMaintenance)
java.rmi.Naming.lookup("equipmentremote");
        ArrayList<equipmentMaintenance> e=eq.select maintenance of device(id1);
        if (e==null)
        {
            System.out.println("no maintenance for "+id1);
        }
        else
        {
            per.addconsole("select the maintenance of device "+id1);
            for (equipmentMaintenance eq1:e)
            {
                System.out.println("_____");
                System.out.print("id :");
                System.out.println(eq1.getId());
                System.out.print("date :");
                System.out.println(eq1.getDate());
                System.out.print("comment :");
                System.out.println(eq1.getTetx());
                System.out.println();
            }
        }
    }

}

}

catch (Exception e)
{
    e.printStackTrace();
    System.out.println("the server is down ...");
}

}

public static void monitor_select_menu()
{
    String s;
    do {

        Monitor monitor=new Monitor();
        Scanner scanner = new Scanner(System.in);
        System.out.println("_____ monitor section
_____");
        System.out.println();
        System.out.println(" [1] add monitor ");
        System.out.println(" [2] delete monitor ");
        System.out.println(" [3] update monitor ");
        System.out.println(" [4] select monitor ");
        System.out.println(" [5] select maintenance of monitors ");
        System.out.println(" [6] main page ");
        System.out.println("your answer : ");
    }
}

```



```

        int result = scanner.nextInt();

        switch (result) {
            case 1:

                monitor.add();
                break;

            case 2:
                monitor.delete();
                break;

            case 3:
                monitor.update();
                break;

            case 4:
                monitor.select();
                break;
            case 5:
                monitor.select_don_maintenance();
                break;
            case 6:
                main.select_menu();
                break;
            default:
                System.out.println("wrong input");
                break;
        }
        s=try_again();

    }
    while (s.equals("y") || s.equals("Y"));
    main.select_menu();
}

package classes.viewmanager;

import managment.RmiServer.Remote.PersonLoginAddconsol;
import managment.RmiServer.Remote.equipmentMaintenace;
import managment.model.TO.equipmentMainteneace;
import start.main;

import java.io.Serializable;
import java.rmi.Naming;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.Scanner;

/**
 * Created by saeedtavana on 6/5/16.
 */
public class NetworkUtility implements Serializable , deviceElectronic {
    public Date d=new Date();
    public SimpleDateFormat simpleDateFormat=new SimpleDateFormat("YYYY/MM/dd");
    private static String try_again()
    {
        Scanner scanner=new Scanner(System.in);
        System.out.println(" do you want to try again [y/n] ? :");
        String ans=scanner.nextLine();
        return ans;
    }
    @Override
    public void add() {
        Scanner scanner1=new Scanner(System.in);
        managment.model.TO.NetworkUtility scannerobj= new

```

```

managment.model.TO.NetworkUtility();
    System.out.print("please enter the id : ");
    scannerobj.setId(scanner1.nextInt());
    if (scannerobj.getId()==0)
    {
        System.out.println("id can not be empty or 0 ");
    }

    System.out.print("please enter the brand : ");
    scannerobj.setBrand(scanner1.next());
    System.out.print("please enter the modle : ");
    scannerobj.setModel(scanner1.next());
    System.out.print("please enter the type : ");
    scannerobj.setType(scanner1.next());
    System.out.print("please enter the comments : ");
    scannerobj.setComments(scanner1.next());
    String date2=simpleDateFormat.format(d)+"";
    scannerobj.setDate(date2);
    Scanner scanner2=new Scanner(System.in);
    System.out.println(" do you want to add [y/n] ? :");
    String ans=scanner2.nextLine();
    if (ans.equals("Y")||ans.equals("y")) {

        try {

            PersonLoginAddconsol
per=(PersonLoginAddconsol) java.rmi.Naming.lookup("addconsole");
            managment.RmiServer.Remote.NetworkUtility
sc=(managment.RmiServer.Remote.NetworkUtility)
                java.rmi.Naming.lookup("networkremote");
            boolean bb=sc.addpnetwork(scannerobj);
            if (bb==true)
            {
                per.addconsole("the network utility device is added");
                System.out.println("the device is added");
            }
            else if (bb==false)
            {
                System.out.println("the device is not added");
            }

        }
        catch (Exception e)
        {
            System.out.println("the serve is down ...");
        }

    }
    else
    {

    }

}

@Override
public void delete() {
    Scanner scanner1=new Scanner(System.in);
    System.out.print("please enter the id of network utility you want to delete
:");
    int idd=scanner1.nextInt();
    if (idd==0)
    {
        System.out.println("the id nust not be empty or 0 ");
    }
}

```

```

        else
        {
            try {

                PersonLoginAddconsole
per=(PersonLoginAddconsole) java.rmi.Naming.lookup("addconsole");
                managment.RmiServer.Remote.NetworkUtility
sc=(managment.RmiServer.Remote.NetworkUtility)
                java.rmi.Naming.lookup("networkremote");
                boolean bb=sc.delenetwork(idd);
                if (bb==true)
                {
                    per.addconsole("the network utility device is deleted");
                    System.out.println("the device is deleted");
                }
                else if (bb==false)
                {
                    System.out.println("the device is not deleted");
                }

            }

            catch (Exception e)
            {
                System.out.println("the server is down ...");
            }
        }
    }

    @Override
    public void select() {
        Scanner scanner1=new Scanner(System.in);
        System.out.print("enter the id of network utility : ");
        int idd =scanner1.nextInt();
        if (idd==0)
        {
            System.out.println("the id must not empty or 0");
        }
        else {

            try {

                PersonLoginAddconsole
per=(PersonLoginAddconsole) java.rmi.Naming.lookup("addconsole");
                managment.RmiServer.Remote.NetworkUtility
sc=(managment.RmiServer.Remote.NetworkUtility)
                java.rmi.Naming.lookup("networkremote");
                ArrayList<managment.model.TO.NetworkUtility>
sobj=sc.selectnetwork(idd);
                if (sobj==null)
                {
                    System.out.println("there is no scanner found ");
                }
                else
                {

                    per.addconsole("select network utility device");
                    for (managment.model.TO.NetworkUtility sca:sobj) {
                        System.out.println(" ");
                        System.out.print(" id : ");
                        System.out.println(sca.getId());
                        System.out.print("model  :");
                        System.out.println(sca.getModel());
                        System.out.print("brand  :");
                        System.out.println(sca.getBrand());
                        System.out.print("type  :");
                    }
                }
            }
        }
    }
}

```

```

        System.out.println(sca.getType());
        System.out.print("date :");
        System.out.println(sca.getDate());
        System.out.print("comments :");
        System.out.println(sca.getComments());
        System.out.println();
    }
}

}

}
catch (Exception e)
{
    System.out.println("the server is down ...");
}
}

}

@Override
public void update() {
    Scanner scanner1=new Scanner(System.in);
    System.out.println("please enter the id");
    int idd=scanner1.nextInt();
    System.out.println("please enter the new model ");
    String ss=scanner1.next();
    if (idd==0)
    {
        System.out.println("the id must not emoty or 0");
    }
    else
    {
        try {

            PersonLoginAddconsol
per=(PersonLoginAddconsol) java.rmi.Naming.lookup("addconsole");
            managment.RmiServer.Remote.NetworkUtility
sc=(managment.RmiServer.Remote.NetworkUtility)
                java.rmi.Naming.lookup("networkremote");
            boolean bb=sc.updatenetwork(idd,ss);
            if (bb==true)
            {
                per.addconsole("the network utility device is updated");
                System.out.println("the device is updated");
            }
            else if (bb==false)
            {
                System.out.println("the device is not updated");
            }

        }
        catch (Exception e)
        {

            System.out.println("the server is down ...");
        }
    }
}

}

@Override
public void select_don_maintenance() {
    Scanner scanner=new Scanner(System.in);
    System.out.print("please enter the id of network utility you want to see
the maintenance :");
}

```

```

        int id1=scanner.nextInt();
        try {

            PersonLoginAddconsole
            per=(PersonLoginAddconsole) java.rmi.Naming.lookup("addconsole");
            equipmentMaintenance eq =(equipmentMaintenance)
            java.rmi.Naming.lookup("equipmentremote");
            ArrayList<equipmentMaintenance> e=eq.select_maintenance_of_device(id1);
            if (e==null)
            {
                System.out.println("no maintenance for "+id1);
            }
            else
            {
                per.addconsole("select the maintenance of device "+id1);
                for (equipmentMaintenance eq1:e)
                {
                    System.out.println("_____");
                    System.out.print("id :");
                    System.out.println(eq1.getId());
                    System.out.print("date :");
                    System.out.println(eq1.getDate());
                    System.out.print("comment :");
                    System.out.println(eq1.getTetx());
                    System.out.println();
                }
            }
        }
        catch (Exception e)
        {
            System.out.println("the server is down ...");
        }
    }

    public static void networkutility_select_menu()
    {
        String s;
        do {

            NetworkUtility networkUtility=new NetworkUtility();
            Scanner scanner = new Scanner(System.in);
            System.out.println("_____ network utility section
            ");
            System.out.println();
            System.out.println(" [1] add network utility ");
            System.out.println(" [2] delete network utility ");
            System.out.println(" [3] update network utility ");
            System.out.println(" [4] select network utility ");
            System.out.println(" [5] select maintenance of network utility ");
            System.out.println(" [6] main page ");
            System.out.println("your answer : ");
            int result = scanner.nextInt();

            switch (result) {
                case 1:
                    networkUtility.add();
                    break;

                case 2:
                    networkUtility.delete();
                    break;
            }
        } while (s != "q");
    }
}

```

```

        case 3:
            networkUtility.update();
            break;

        case 4:
            networkUtility.select();
            break;

        case 5:
            networkUtility.select_don_maintenance();
            break;

        case 6:
            main.select_menu();
            break;

        default:
            System.out.println("wrong input");
            break;
    }
    s = try_again();

}
while (s.equals("y") || s.equals("Y"));
main.select_menu();
}
}
package classes.viewmanager;

import managment.RmiServer.Remote.PersonLoginAddconsol;
import managment.RmiServer.Remote.equipmentMaintenace;
import managment.model.TO.equipmentMainteneace;
import start.main;

import java.io.Serializable;
import java.rmi.Naming;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.Scanner;

/**
 * Created by saeedtavana on 6/5/16.
 */
public class Printer implements Serializable, deviceElectronic {

    public Date d=new Date();
    public SimpleDateFormat simpleDateFormat=new SimpleDateFormat("YYYY/MM/dd");
    private static String try_again() {
        Scanner scanner = new Scanner(System.in);
        System.out.println(" do you want to try again [y/n] ? :");
        String ans = scanner.nextLine();
        return ans;
    }

    @Override
    public void add() {

        Scanner scanner1=new Scanner(System.in);
        managment.model.TO.Printer scanerobj=new managment.model.TO.Printer();
        System.out.print("please enter the id : ");
        scanerobj.setId(scanner1.nextInt());
        if (scanerobj.getId()==0)
        {
            System.out.println("id can not be empty or 0 ");
        }
        System.out.print("please enter the user : ");
        scanerobj.setUser(scanner1.next());
        System.out.print("please enter the room number : ");
        scanerobj.setRoomNumber(scanner1.nextInt());
    }

```

```

        System.out.print("please enter the secction : ");
        scannerobj.setSection(scanner1.next());
        System.out.print("please enter the brand : ");
        scannerobj.setBrand(scanner1.next());
        System.out.print("please enter the modle : ");
        scannerobj.setModel(scanner1.next());
        System.out.print("please enter the type : ");
        scannerobj.setType(scanner1.next());
        System.out.print("please enter the comments : ");
        scannerobj.setCommand(scanner1.next());
        String date2=simpleDateFormat.format(d)+"";
        scannerobj.setDate(date2);
        Scanner scanner2=new Scanner(System.in);
        System.out.println(" do you want to add [y/n] ? :");
        String ans=scanner2.nextLine();
        if (ans.equals("Y")||ans.equals("y")) {

            try {

                PersonLoginAddconsol
per=(PersonLoginAddconsol) java.rmi.Naming.lookup("addconsole");
                managment.RmiServer.Remote.Printer
sc=(managment.RmiServer.Remote.Printer)
                    Naming.lookup("printerremote");
                boolean bb=sc.addprinter(scannerobj);
                if (bb==true)
                {
                    per.addconsole("the scanner device is added");
                    System.out.println("the device is added");
                }
                else if (bb==false)
                {
                    System.out.println("the device is not added");
                }

            }

            catch (Exception e)
            {
                System.out.println("the serve is down ...");
            }

        }
        else
        {

        }

    }

    @Override
    public void delete() {
        Scanner scanner1=new Scanner(System.in);
        System.out.print("please enter the id of printer you want to delete :");
        int idd=scanner1.nextInt();
        if (idd==0)
        {
            System.out.println("the id nust not be empty or 0 ");
        }
        else
        {
            try {
                PersonLoginAddconsol
per=(PersonLoginAddconsol) java.rmi.Naming.lookup("addconsole");
                managment.RmiServer.Remote.Printer
sc=(managment.RmiServer.Remote.Printer)
                    java.rmi.Naming.lookup("printerremote");
            }
        }
    }

```

```

        boolean bb=sc.deleprinter (idd) ;
        if (bb==true)
        {
            per.addconsole("the printer device is deleted");
            System.out.println("the device is deleted");
        }
        else if (bb==false)
        {
            System.out.println("the device is not deleted");
        }

    }

    }
    catch (Exception e)
    {
        System.out.println("the server is down ...");
    }
}

@Override
public void select() {
    Scanner scanner1=new Scanner(System.in);
    System.out.print("enter the id of printer : ");
    int idd =scanner1.nextInt();
    if (idd==0)
    {
        System.out.println("the id must not emoty or 0");
    }
    else {

        try {
            PersonLoginAddconsol
per=(PersonLoginAddconsol) java.rmi.Naming.lookup("addconsole");
            managment.RmiServer.Remote.Printer
sc=(managment.RmiServer.Remote.Printer)
                java.rmi.Naming.lookup("printerremote");
            ArrayList<managment.model.TO.Printer> sobj=sc.selectprinter (idd) ;
            if (sobj==null)
            {
                System.out.println("there is no printer found ");
            }
            else
            {

                per.addconsole("select scanner device");
                for (managment.model.TO.Printer sca:sobj) {
                    System.out.println("_____");
                    System.out.print(" id : ");
                    System.out.println(sca.getId());
                    System.out.print("user : ");
                    System.out.println(sca.getUuser());
                    System.out.print("room number :");
                    System.out.println(sca.getRoomNumber());
                    System.out.print("section  :");
                    System.out.println(sca.getSection());
                    System.out.print("model  :");
                    System.out.println(sca.getModel());
                    System.out.print("brand  :");
                    System.out.println(sca.getBrand());
                    System.out.print("type  :");
                    System.out.println(sca.getType());
                    System.out.print("date  :");
                    System.out.println(sca.getDate());
                    System.out.print("comments  :");
                    System.out.println(sca.getCommand());
                }
            }
        }
    }
}

```



```

        System.out.println();
    }
}

}
catch (Exception e)
{
    System.out.println("the server is down ...");
}
}

@Override
public void update() {
    Scanner scanner1=new Scanner(System.in);
    System.out.println("please enter the id");
    int idd=scanner1.nextInt();
    System.out.println("please enter the new model ");
    String ss=scanner1.next();
    if (idd==0)
    {
        System.out.println("the id must not emoty or 0");
    }
    else
    {
        try {
            PersonLoginAddconsol
per=(PersonLoginAddconsol) java.rmi.Naming.lookup("addconsole");
            managment.RmiServer.Remote.Printer
sc=(managment.RmiServer.Remote.Printer)
                java.rmi.Naming.lookup("printerremote");
            boolean bb=sc.updatprinter(idd,ss);
            if (bb==true)
            {
                per.addconsole("the printer device is updated");
                System.out.println("the device is updated");
            }
            else if (bb==false)
            {
                System.out.println("the device is not updated");
            }
        }
        catch (Exception e)
        {
            System.out.println("the server is down ...");
        }
    }
}

@Override
public void select_don_maintenance() {
    Scanner scanner=new Scanner(System.in);
    System.out.print("please enter the id of printer you want to see the
maintenance :");
    int idl=scanner.nextInt();
    try {
        PersonLoginAddconsol
per=(PersonLoginAddconsol) java.rmi.Naming.lookup("addconsole");
        equipmentMaintenance eq =(equipmentMaintenance)

```

```

java.rmi.Naming.lookup("equipmentremote");
    ArrayList<equipmentMainteneace> e=eq.select_maintenance_of_device(id1);
    if (e==null)
    {
        System.out.println("no maintenance for  "+id1);
    }
    else
    {
        per.addconsole("select the maintenance of device  "+id1);
        for (equipmentMainteneace eq1:e)
        {
            System.out.println("_____");
            System.out.print("id  :");
            System.out.println(eq1.getId());
            System.out.print("date  :");
            System.out.println(eq1.getDate());
            System.out.print("comment  :");
            System.out.println(eq1.getTetx());
            System.out.println();
        }
    }
}

catch (Exception e)
{
    e.printStackTrace();
    System.out.println("the server is down ...");
}

}

public static void printer_select_menu()
{
    String s;
    do {

        Printer printer=new Printer();
        Scanner scanner = new Scanner(System.in);
        System.out.println("_____ printer section
        _____");
        System.out.println();
        System.out.println(" [1] add printer ");
        System.out.println(" [2] delete printer ");
        System.out.println(" [3] update printer ");
        System.out.println(" [4] select printer ");
        System.out.println(" [5] select maintenance of printer ");
        System.out.println(" [6] main page ");
        System.out.println("your answer : ");
        int result = scanner.nextInt();

        switch (result) {
            case 1:

                printer.add();
                break;

            case 2:
                printer.delete();
                break;

            case 3:
                printer.update();
                break;

            case 4:

```

```

        printer.select();
        break;
    case 5:
        printer.select_don_maintenance();
        break;
    case 6:
        main.select_menu();
        break;
    default:
        System.out.println("wrong input");
        break;
    }
    s = try_again();

    }
    while (s.equals("y") || s.equals("Y"));
    main.select_menu();
}
}
package classes.viewmanager;

import managment.RmiServer.Remote.PersonLoginAddconsol;
import managment.model.TO.room;
import start.main;

import java.util.ArrayList;
import java.util.Scanner;

/**
 * Created by saeedtavana on 7/3/16.
 */
public class Room implements deviceElectronic {
    private static String try_again()
    {
        Scanner scanner=new Scanner(System.in);
        System.out.println(" do you want to try again [y/n] ? :");
        String ans=scanner.nextLine();
        return ans;
    }

    public static String getAnswer() {
        return answer;
    }

    public static void setAnswer(String answer) {
        Room.answer = answer;
    }

    public static String answer;
    @Override
    public void add() {

        Scanner scanner=new Scanner(System.in);
        room room=new room();

        System.out.print(" please enter the room number : ");
        room.setNumber(scanner.nextInt());
        System.out.print("please enter the telephone : ");
        room.setTelephone(scanner.nextInt());
        System.out.println();
        Scanner scanner2=new Scanner(System.in);
        System.out.println(" do you want to add [y/n] ? :");
        String ans=scanner2.nextLine();

        if (ans.equals("Y") || ans.equals("y")) {

```

```

        try {
            PersonLoginAddconsol
per=(PersonLoginAddconsol) java.rmi.Naming.lookup("addconsole");
            managment.RmiServer.Remote.room
room1=(managment.RmiServer.Remote.room)
                java.rmi.Naming.lookup("roomremote");
            boolean b=room1.addroom(room);
            if (b==true)
            {
                per.addconsole("the room is added");
                System.out.println("room sucesfully added");
            }
            else if (b==false)
            {
                System.out.println("room does not add");
            }
        }
        catch (Exception e)
        {
            System.out.println("the server is down ...");
        }

    }
    else
    {

    }

}

@Override
public void delete() {

    Scanner scanner=new Scanner(System.in);
    System.out.print("please enter the number of room you want to delete :");
    int roomid=scanner.nextInt();
    try {
        PersonLoginAddconsol
per=(PersonLoginAddconsol) java.rmi.Naming.lookup("addconsole");
        managment.RmiServer.Remote.room room1=(managment.RmiServer.Remote.room)
            java.rmi.Naming.lookup("roomremote");
        boolean b=room1.deleteroom(roomid);
        if (b==true)
        {
            per.addconsole("the room is deleted");
            System.out.println(" the room is deleted");
        }
        else if (b==false)
        {
            System.out.println("room does not deleted");
        }
    }
    catch (Exception e)
    {
        System.out.println("the server is down ... ");
    }

}

@Override
public void select() {

    Scanner scanner=new Scanner(System.in);
    System.out.print("please enter the room number : ");
    int roomid= scanner.nextInt();
    try {

```

```

        PersonLoginAddconsol
per=(PersonLoginAddconsol) java.rmi.Naming.lookup("addconsole");
managment.RmiServer.Remote.room room1=(managment.RmiServer.Remote.room)
        java.rmi.Naming.lookup("roomremote");
ArrayList<room> roomobj=room1.selectroom(roomid);
        if (roomobj==null)
        {
                System.out.println("there is no room !");
        }
        else
        {
                per.addconsole("select room with id "+roomid);
                for (room obj :roomobj)
                {
                        System.out.println("_____");
                        System.out.print("room number : ");
                        System.out.println(obj.getNumber());
                        System.out.print("telephone : ");
                        System.out.println(obj.getTelephone());
                        System.out.println();
                }
        }
    }
}
catch (Exception e)
{
    e.printStackTrace();
    System.out.println("the server is down ... ");
}

}

@Override
public void update() {
    Scanner scanner1 = new Scanner(System.in);

    System.out.print("please enter the room number");
    int idd = scanner1.nextInt();
    System.out.println();
    System.out.print("please enter the new telephone ");
    int ss = scanner1.nextInt();
    if (idd == 0) {
        System.out.println("the id must not emoty or 0");
    } else {
        try {
            PersonLoginAddconsol per = (PersonLoginAddconsol)
java.rmi.Naming.lookup("addconsole");
            managment.RmiServer.Remote.room sc =
(managment.RmiServer.Remote.room)
                java.rmi.Naming.lookup("roomremote");
            boolean bb = sc.updateroom(idd,ss);
            if (bb == true) {
                per.addconsole("the room is updated");
                System.out.println("the room is updated");
            } else if (bb == false) {
                System.out.println("the room is not updated");
            }
        }
    }
} catch (Exception e) {

    System.out.println("the server is down ...");
}
}

```

```

    }
}

@Override
public void select_don_maintenance() {
    //this must not work
}

public static void room_select_menu() {
    String s;
    do {

        Room room=new Room();
        Scanner scanner = new Scanner(System.in);
        System.out.println("_____ room section _____");
        System.out.println();
        System.out.println(" [1] add room ");
        System.out.println(" [2] delete room ");
        System.out.println(" [3] update room ");
        System.out.println(" [4] select room ");
        System.out.println(" [5] main page ");
        System.out.println("your answer : ");
        int result = scanner.nextInt();

        switch (result) {
            case 1:
                room.add();

                break;

            case 2:
                room.delete();
                break;

            case 3:
                room.update();
                break;

            case 4:
                room.select();
                break;
            case 5:
                main.select_menu();
                break;
            default:
                System.out.println("wrong input");
                break;
        }
        s = try_again();

    }
    while (s.equals("y") || s.equals("Y"));
    main.select_menu();
}

}

package classes.viewmanager;

import managment.RmiServer.Remote.PersonLoginAddconsol;
import managment.RmiServer.Remote.equipmentMaintenace;
import managment.model.TO.equipmentMainteneace;
import start.main;

import java.io.IOException;
import java.io.Serializable;
import java.text.SimpleDateFormat;
import java.util.ArrayList;

```

```

import java.util.Date;
import java.util.Scanner;

/**
 * Created by saeedtavana on 6/5/16.
 */
public class Scanner implements Serializable, deviceElectronic {

    public Date d=new Date();
    public SimpleDateFormat simpleDateFormat=new SimpleDateFormat("YYYY/MM/dd");
    public static String try_again()
    {
        Scanner scanner=new Scanner(System.in);
        System.out.println(" do you want to try again [y/n] ? :");
        String ans=scanner.nextLine();
        return ans;
    }
    @Override
    public void add() {

        Scanner scanner1=new Scanner(System.in);
        managment.model.TO.Scanner scannerobj=new managment.model.TO.Scanner();
        System.out.print("please enter the id : ");
        scannerobj.setId(scanner1.nextInt());
        if (scannerobj.getId()==0)
        {
            System.out.println("id can not be empty or 0 ");
        }
        System.out.print("please enter the user : ");
        scannerobj.setUser(scanner1.next());
        System.out.print("please enter the room number : ");
        scannerobj.setRoomNumber(scanner1.nextInt());
        System.out.print("please enter the secction : ");
        scannerobj.setSection(scanner1.next());
        System.out.print("please enter the brand : ");
        scannerobj.setBrand(scanner1.next());
        System.out.print("please enter the modle : ");
        scannerobj.setModel(scanner1.next());
        System.out.print("please enter the type : ");
        scannerobj.setType(scanner1.next());
        System.out.print("please enter the comments : ");
        scannerobj.setCommand(scanner1.next());
        String date2=simpleDateFormat.format(d)+" ";
        scannerobj.setDate(date2);
        Scanner scanner2=new Scanner(System.in);
        System.out.println(" do you want to add [y/n] ? :");
        String ans=scanner2.nextLine();
        if (ans.equals("Y")||ans.equals("y")) {

            try {

                PersonLoginAddconsol
per=(PersonLoginAddconsol) java.rmi.Naming.lookup("addconsole");
                managment.RmiServer.Remote.Scanner
sc=(managment.RmiServer.Remote.Scanner)
                java.rmi.Naming.lookup("scannerremote");
                boolean bb=sc.addscanneer(scannerobj);
                if (bb==true)
                {
                    per.addconsole("the scanner device is added");
                    System.out.println("the device is added");
                }
                else if (bb==false)
                {
                    System.out.println("the device is not added");
                }
            }
        }
    }
}

```

```

        }
        catch (Exception e)
        {
            System.out.println("the serve is down ...");
        }

    }
    else
    {

    }

}

@Override
public void delete() {
    Scanner scanner1=new Scanner(System.in);
    System.out.print("please enter the id of scanner you want to delete :");
    int idd=scanner1.nextInt();
    if (idd==0)
    {
        System.out.println("the id must not be empty or 0 ");
    }
    else
    {
        try {
            PersonLoginAddconsol
per=(PersonLoginAddconsol) java.rmi.Naming.lookup("addconsole");
            managment.RmiServer.Remote.Scaner
sc=(managment.RmiServer.Remote.Scaner)
                java.rmi.Naming.lookup("scannerremote");
            boolean bb=sc.delescanner(idd);
            if (bb==true)
            {
                per.addconsole("the scanner device is deleted");
                System.out.println("the device is deleted");
            }
            else if (bb==false)
            {
                System.out.println("the device is not deleted");
            }

        }

    }
    catch (Exception e)
    {
        System.out.println("the server is down ...");
    }

}

@Override
public void select() {
    Scanner scanner1=new Scanner(System.in);
    System.out.print("enter the id of scanner : ");
    int idd =scanner1.nextInt();
    if (idd==0)
    {
        System.out.println("the id must not emoty or 0");
    }
    else {

        try {

```



```

        PersonLoginAddconsol
per=(PersonLoginAddconsol) java.rmi.Naming.lookup("addconsole");
        managment.RmiServer.Remote.Scanner
sc=(managment.RmiServer.Remote.Scanner)
        java.rmi.Naming.lookup("scannerremote");
        ArrayList<managment.model.TO.Scanner> sobj=sc.selectscanner(idd);
        if (sobj==null)
        {
            System.out.println("there is no scanner found ");
        }
        else
        {
            per.addconsole("select scanner device");
            for (managment.model.TO.Scanner sca:sobj) {
                System.out.println("_____");
                System.out.print(" id : ");
                System.out.println(sca.getId());
                System.out.print("user : ");
                System.out.println(sca.getUuser());
                System.out.print("room number :");
                System.out.println(sca.getRoomNumber());
                System.out.print("section :");
                System.out.println(sca.getSection());
                System.out.print("model :");
                System.out.println(sca.getModel());
                System.out.print("brand :");
                System.out.println(sca.getBrand());
                System.out.print("type :");
                System.out.println(sca.getType());
                System.out.print("date :");
                System.out.println(sca.getDate());
                System.out.print("comments :");
                System.out.println(sca.getCommand());
                System.out.println();
            }
        }
    }
    catch (Exception e)
    {
        e.printStackTrace();
        System.out.println("the server is down ...");
    }
}

@Override
public void update() {
    Scanner scanner1=new Scanner(System.in);
    System.out.println("please enter the id");
    int idd=scanner1.nextInt();
    System.out.println("please enter the new model ");
    String ss=scanner1.next();
    if (idd==0)
    {
        System.out.println("the id must not emoty or 0");
    }
    else
    {
        try {
            PersonLoginAddconsol
per=(PersonLoginAddconsol) java.rmi.Naming.lookup("addconsole");

```

```

        managment.RmiServer.Remote.Scanner
sc=(managment.RmiServer.Remote.Scanner)
        java.rmi.Naming.lookup("scannerremote");
        boolean bb=sc.updatescanner(idd,ss);
        if (bb==true)
        {
            per.addconsole("the scanner device is updated");
            System.out.println("the device is updated");
        }
        else if (bb==false)
        {
            System.out.println("the device is not updated");
        }
    }
}
catch (Exception e)
{
    System.out.println("the server is down ...");
}
}

@Override
public void select_don_maintenance() {
    Scanner scanner=new Scanner(System.in);
    System.out.print("please enter the id of scanner you want to see the
maintenance :");
    int idl=scanner.nextInt();
    try {
        PersonLoginAddconsol
per=(PersonLoginAddconsol) java.rmi.Naming.lookup("addconsole");
        equipmentMaintenace eq =(equipmentMaintenace)
java.rmi.Naming.lookup("equipmentremote");
        ArrayList<equipmentMainteneace> e=eq.select_maintenance_of_device(idl);
        if (e==null)
        {
            System.out.println("no maintenance for "+idl);
        }
        else
        {
            per.addconsole("select the maintenance of device "+idl);
            for (equipmentMainteneace eq1:e)
            {
                System.out.println("_____");
                System.out.print("id :");
                System.out.println(eq1.getId());
                System.out.print("date :");
                System.out.println(eq1.getDate());
                System.out.print("comment :");
                System.out.println(eq1.getTetx());
                System.out.println();
            }
        }
    }
}
catch (Exception e)
{
    System.out.println("the server is down ...");
}
}

```

```

    }

    public static void scanner_select_menu() {
        String s;
        do {

            Scanner scannel=new Scanner();
            Scanner scanner = new Scanner(System.in);
            System.out.println("_____ scanner section
            _____");
            System.out.println();
            System.out.println(" [1] add scanner ");
            System.out.println(" [2] delete scanner ");
            System.out.println(" [3] update scanner ");
            System.out.println(" [4] select scanner ");
            System.out.println(" [5] select maintenance of scanner ");
            System.out.println(" [6] main page ");
            System.out.println("your answer : ");
            int result = scanner.nextInt();

            switch (result) {
                case 1:
                    scannel.add();

                    break;

                case 2:
                    scannel.delete();
                    break;

                case 3:
                    scannel.update();
                    break;

                case 4:
                    scannel.select();
                    break;
                case 5:
                    scannel.select_don_maintenance();
                    break;
                case 6:
                    main.select_menu();
                    break;
                default:
                    System.out.println("wrong input");
                    break;
            }
            s = try_again();

        }
        while (s.equals("Y") || s.equals("Y"));
        main.select_menu();
    }
}

```