

California State University, San Bernardino

CSUSB ScholarWorks

Theses Digitization Project

John M. Pfau Library

2006

A web accessible clinical patient information networked system

Andrew Yee Chang

Follow this and additional works at: <https://scholarworks.lib.csusb.edu/etd-project>



Part of the [Databases and Information Systems Commons](#)

Recommended Citation

Chang, Andrew Yee, "A web accessible clinical patient information networked system" (2006). *Theses Digitization Project*. 2980.

<https://scholarworks.lib.csusb.edu/etd-project/2980>

This Project is brought to you for free and open access by the John M. Pfau Library at CSUSB ScholarWorks. It has been accepted for inclusion in Theses Digitization Project by an authorized administrator of CSUSB ScholarWorks. For more information, please contact scholarworks@csusb.edu.

A WEB ACCESSIBLE CLINICAL PATIENT
INFORMATION NETWORKED SYSTEM

A Project
Presented to the
Faculty of
California State University,
San Bernardino

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
in
Computer Science

by
Andrew Yee Chang

June 2006

A WEB ACCESSIBLE CLINICAL PATIENT
INFORMATION NETWORKED SYSTEM

A Project
Presented to the
Faculty of
California State University,
San Bernardino

by
Andrew Yee Chang
June 2006


Approved by:




Dr. Ernesto Gomez, Chair, Computer Science

15 MAY 9 2006

Date



Dr. Richard Botting



Dr. George Georgiou

A WEB ACCESSIBLE CLINICAL PATIENT
INFORMATION NETWORKED SYSTEM

A Project
Presented to the
Faculty of
California State University,
San Bernardino

by
Andrew Yee Chang
June 2006

Approved by:

Dr. Ernesto Gomez, Chair, Computer Science

Date

Dr. Richard Botting

Dr. George Georgiou

ABSTRACT

The WACPINS is a software program designed for the outpatient clinical area. The project was envisioned because of the relatively inefficient methods currently used to document and store patient data. Computerizing the entire patient data storage method would lead to better efficiency in the outpatient clinical area.

The system stores all pertinent and relevant patient data such as lab results, patient history and X-ray images to name a few. The system is accessible via the internet as well as operable over a local area network.

This project was developed using Java Server Pages or JSP. This language basically integrates the Java programming language with Hyper Text Markup Language or HTML, to create dynamic web pages. The database used in this project was developed by Oracle and is using their latest version, Oracle 9i. The JSP pages communicate with the database and will display information dynamically depending on the contents of the database.

ACKNOWLEDGMENTS

We as a team would like to thank Dr. Mendoza first of all. Her input to the project was invaluable and although she was not part of the project committee, she was always willing to lend a helping hand.

Of course we would also like to thank our committee members which consist of Dr. Gomez, Dr. Botting, and Dr. Georgiou. Without their support, this paper would not have been possible.

Last but not least, we would like to thank Dr. Concepcion for his support as well. Many a times, we have just walked in on him in his office with no prior appointment but he was always willing to see us and answer our questions.

TABLE OF CONTENTS

ABSTRACT	iii
ACKNOWLEDGMENTS	iv
LIST OF FIGURES	vi
CHAPTER ONE: INTRODUCTION TO THE PROJECT	
1.1 Overview of the Project	1
1.2 Purpose of the Project	2
1.3 Scope	3
1.4 Significance of the Project	5
1.5 Limitations of the Project	6
1.6 Functional Requirements	8
1.7 Graphical User Interface	17
CHAPTER TWO: LITERATURE REVIEW	
2.1 Review of Related Literature	45
CHAPTER THREE: METHODOLOGY	
3.1 Methodology	48
CHAPTER FOUR: CONCLUSIONS AND RECOMMENDATIONS	
4.1 Summary	55
4.2 Conclusions of the Project	55
4.3 Recommendations	56
APPENDIX: SOURCE CODE	58
REFERENCES	95

LIST OF FIGURES

Figure 1.	Class Diagram	9
Figure 2.	Use-Case Diagram	10
Figure 3.	Deployment Diagram	12
Figure 4.	Entity Relation Diagram Part 1	13
Figure 5.	Entity Relation Diagram Part 2	14
Figure 6.	Entity Relation Diagram Part 3	15
Figure 7.	Entity Relation Diagram Part 4	16
Figure 8.	Entity Relation Diagram Part 5	17
Figure 9.	Desktop Icon of Startup.bat	19
Figure 10.	Screenshot of Successful Startup	20
Figure 11.	Icon of Enterprise Manager	21
Figure 12.	Enterprise Launch Screen	22
Figure 13.	Initial Logged-in Screen	23
Figure 14.	Login Screen to ANDREW Database	24
Figure 15.	System Database Administrators (SYSDBA) Login Screen	25
Figure 16.	Database ANDREW Schema	26
Figure 17.	Table Subtrees	27
Figure 18.	Table Attributes	29
Figure 19.	Table Editor Screen	30
Figure 20.	Localhost Intranet Address	32
Figure 21.	Welcome Page	33
Figure 22.	Web Accessible Clinical Patient Information Networked System (WACPINS) Login Page	34

Figure 23. Main Selection Screen	35
Figure 24. Error Page #1	36
Figure 25. Urinalysis Results for a Patient	37
Figure 26. Secondary Lab Selection Page	39
Figure 27. Error Page #2	40
Figure 28. X-Ray Image Page	41
Figure 29. Image of a Wrist Fracture	42
Figure 30. Shutdown.bat File	43
Figure 31. Shutdown Sequence	44

CHAPTER ONE.

INTRODUCTION TO THE PROJECT

1.1 Overview of the Project

The WACPINS is a web based, patient information system that is intended for the clinical outpatient area of healthcare. The user interfaces includes interfaces on the Internet for the Web accessible portion as well as the standard interfaces for the entering of patient information in the not online portion of the system. There is also an implementation of a login page where it requires users to enter a previously supplied user ID and password.

The hardware interface requirement is that it must run on the existing web server. The software interface is that it must support current versions of Netscape & Internet Explorer. The communications interface requires support for Hyper Text Transfer Protocol (HTTP) and Secure Socket Layer (SSL) The database will be implemented via Oracle version 9i. SSL will be used to provide the security for login and password purposes.

Operations requirements are that it must be available 24 hours a day, 7 days per week. All actions are user initiated. No separate backup and recovery or maintenance

functions are required as that is handled by system administration on the hosting server machine. The database shall be managed by the full time staff of CSUSB CSCI department. User interfaces for WACPINS is designed in HTML page using Java embedded forms. Other interfaces are also implemented in Java and other related languages was also be used. The following features are incorporated to produce a more descriptive representation of the interface.

1.2 Purpose of the Project

The purpose of this project was to develop a simple patient data storage system that will be accessible from the internet. WACPINS, the name of our project stands for "Web Accessible Clinical Patient Information Networked System." This project was envisioned primarily because of both of our undergraduate backgrounds. We both have worked in the health field for a few years and have noticed a deficiency in the clinical outpatient area where data was still stored using paper and filing cabinets. This project was undertaken to try and improve that area of healthcare where the single or group practice simply does not need the big products that big hospitals use. A lot of the functions that are included in those big patient storage

programs are simply not necessary since outpatient care is much less complex. This program was designed with the outpatient clinical area in mind and serves all the basic needs a general practitioner will have.

Another aspect of WACPINS is that it is internet accessible therefore making getting to patient data easier for the physicians who are not always on site but need to know important patient data. The system will also be able to function over a local area network so that a bigger private practice will be able to link all the computers together and share the information.

User requirements for the system were gathered by interviews with several local clinics in the Inland Empire and the general Los Angeles area. Staff interviewed included physicians, nurses, lab technicians and secretaries.

1.3 Scope

This project is a Clinical Patient Data Storage system that stores all relevant patient data. These data may include lab values, patient histories, insurance information, and MRI images among others. This data is stored in a database that was developed by Oracle Enterprises. The version of Oracle used was the latest

version available at the time of the project initiation which is version 9i. The program was coded using Java Server Pages (JSP) which utilizes both the Java programming language and aspects of HTML for the web portion.

The intended audience for this program is essentially the clinical staff. This would of course include the physician(s), nursing staff, secretarial staff, and any other authorized personnel deemed necessary to have access to sensitive patient information. Since patient data is very sensitive and is protected by law from unauthorized viewers, this program was not intended for the regular public, i.e. patients and their families. This project was not intended to let patients access their data from home since we cannot guarantee that the person accessing the sensitive information will be indeed the patient themselves and not somebody else.

This project was developed with the intention to make the patient data storage system in the clinical outpatient area more efficient. It was also designed with the intention to make the accessing of data for the clinic staff to be easier and more readily available. In the end, the patient is the true beneficiary which is the ultimate goal of this software program.

This system is developed with the clinical side of health care in mind. The clinics are mainly for outpatient care and as such, the patient population is radically different from those in a normal hospital environment. Such a system has yet to be developed for the outpatient areas of healthcare. Currently private clinics use a paper filing system that is relatively inefficient and takes up valuable storage space. By switching over to computerized documentation, their practice will be more streamlined and efficient. The WACPINS provides patient information such as their lab reports (which include many different kinds of tests), x-ray images, patient information, patient histories, pertinent allergies, insurance information and MRI/CT reports.

The WACPINS is also capable of giving this information over the World Wide Web. It is able to provide password access to the clinic staff so that they do not have to be at the clinic itself to get to pertinent patient information.

1.4 Significance of the Project

We feel the significance of the project is in its relative simplicity. There are products out there that do indeed store patient data but they are very expensive

programs. The typical outpatient clinic is a 2 or 3 physician practice that simply does not need such a complex program since the nature of what they do is less critical than say a regular hospital. This program is just a simple data storage system that uses a database to store and catalogue patient data. But after much research, we feel that this is all that outpatient clinical area requires.

Something that this program does that we have yet to see in any clinical patient data storage system is the ability to integrate the storage of images along with the storage of regular data. This program is designed to store X-ray, CT, and MRI images along with the regular laboratory data.

1.5 Limitations of the Project

Although there is theoretically an almost unlimited amount of time to finish this project, we as a team decided that it was acceptable to spend at most 3 quarters on the project. These 3 quarters also included the project approval process which unfortunately took longer than we expected. As such, although the program works as specified by the SRS, the aesthetics still leaves something to be desired. The interfaces are written in JSP which is based

off HTML and therefore the pages seem simplistic but are however fully functional.

One important part about our project that should be pointed out in this section will be our decision not to allow users to update the database over the web. This is done first of all due to security reasons but also has other merits as well. The simplicity of this project precluded us from taking into accounts such things as race conditions where users will access information before it has been updated or try to update the database while the information is already in use. Another consideration is also the problem with too many updates to the database which may overload the server.

Another aspect of the project that should be pointed out is that we are students with limited financial resources. Therefore, all the programs such as the database, web server program and OS are basically free. As such, we are sure that there are more professional programs out there that offer more versatility and security but we had to make do with what we had.

A further limitation of this project is the use of HTML versus the use of Cascading Style sheets. As will be explained in our methodology section, our use of HTML is mainly based upon the current standard as what is most

accepted by most browsers. This lead us to use HTML rather than CSS. The non-use of CSS did not really hamper our project in the area of scope or usability but rather in the area of aesthetics. That to us was not such a high consideration but rather the problem of compatibility prompted us to use HTML instead.

However, these are only temporary limitations that can be taken care of over time. This project can be taken over by other students who wish to improve and expand upon the original foundation. Other things that can be added are more security features, better looking graphic user interfaces, and maybe a more robust web server.

1.6 Functional Requirements

In this section, we will show the reader our class, user and deployment diagrams. We will also show our entity relations (ER) diagram to the reader as well.

The first figure we will show is our class diagram. When we looked through all the JSP classes and figured out which one will best suit our needs, we found one that pretty much did all that we needed to do for the project. Since the JSP was used mainly to display web pages and the all important database connection, we did not find a need to rewrite our own class. As such, we used an already

built class to do our project. We will give a description as to what each member function does and what variables we are using.

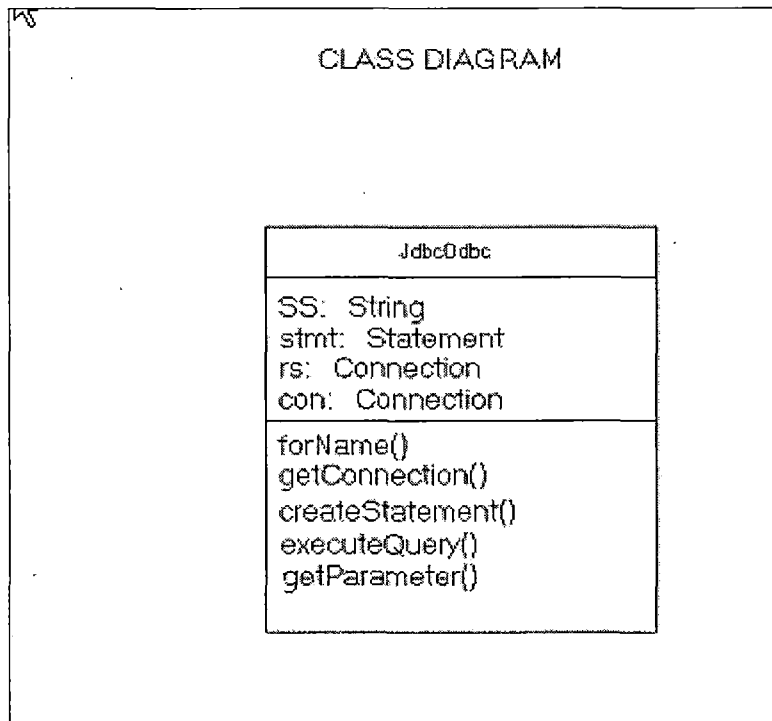


Figure 1. Class Diagram

The `forName()` function is just used to establish a connection to the database of a certain name. One has to know the database name and the login/password to establish a database connection. The next function, `getConnection()` establishes that connection to the database after having all the prerequisite information. The `createStatement()` function just lets us create an object of that function that we can use to set as a SQL function. The

getParameter() function gets the inputted SQL statement and sends it to the executeQuery() function where it will then query the database with the SQL statement and return a result.

The next diagram we will show will be our use-case diagram.

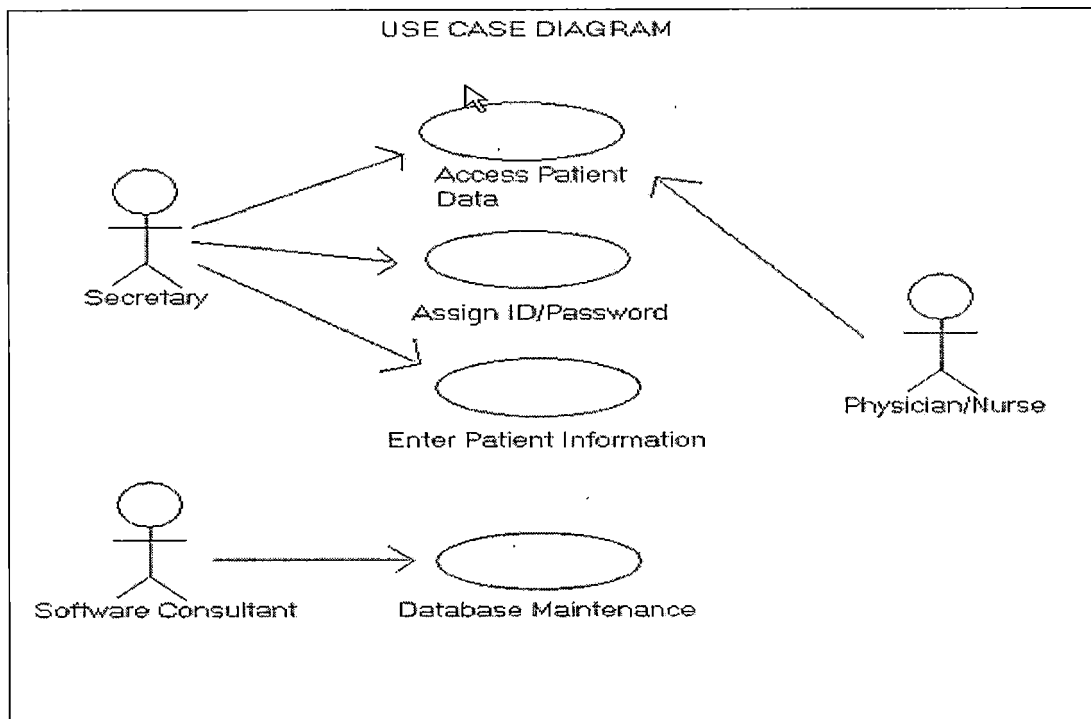


Figure 2. Use-Case Diagram

This diagram basically details who will use the system and in what way. The most important person in this diagram we daresay would be the secretary. Most importantly a well trained one because this system has a little learning curve but nothing we imagine anyone can't

handle. The secretary will be the one responsible for entering, updating, deleting patient information including all new lab values. Currently, we have no automated way for the information to be directly ported from any other infrastructure. That will definitely be a project for future days. The main "users" of this will be the physicians and nurses who take care of the patients and need access to their data. Software consultant(s) will be needed to do regular maintenance and updates to the system.

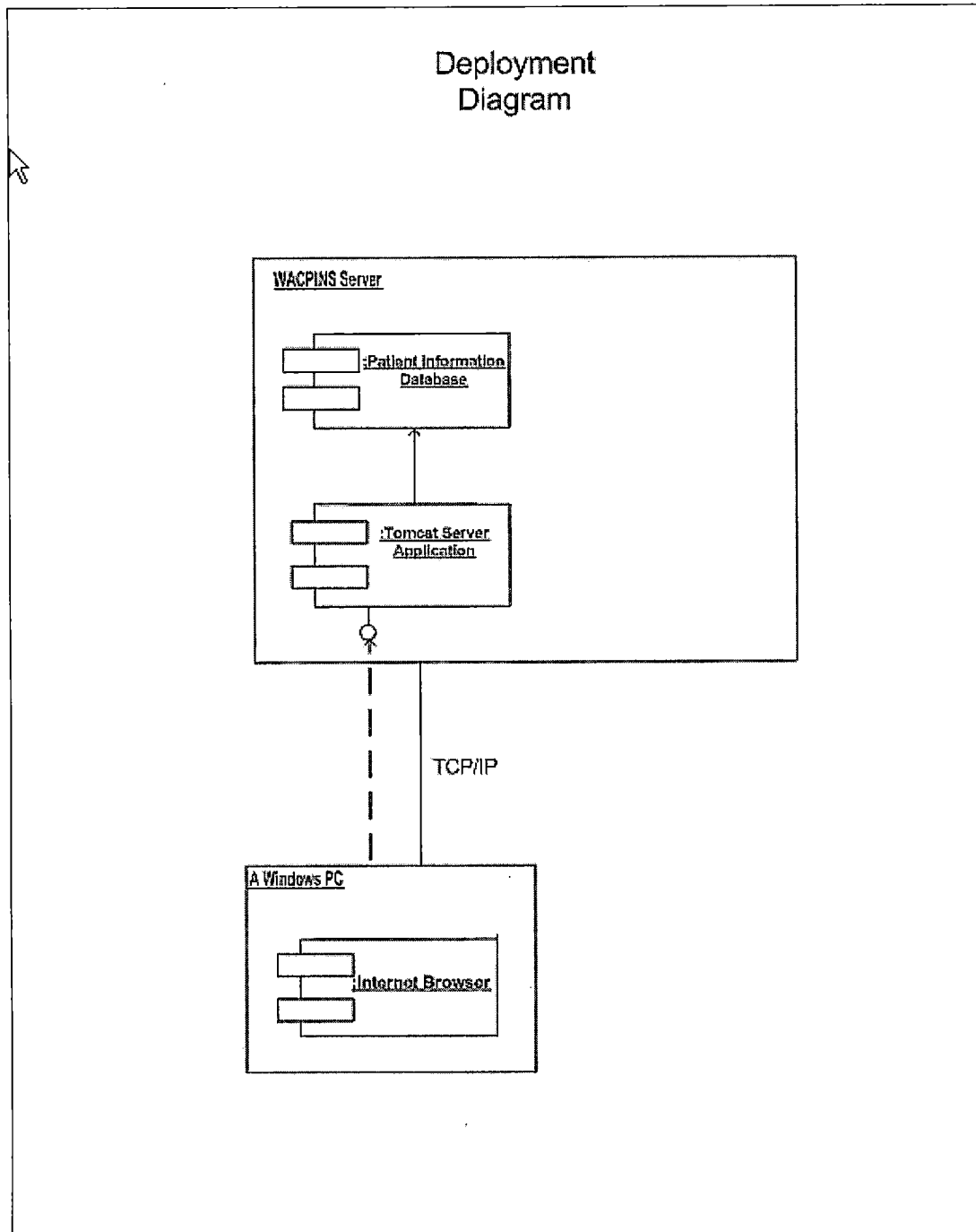


Figure 3. Deployment Diagram

We think the deployment diagram is pretty much self-explanatory therefore we will not go into a description.

Next we will present our entity relations diagrams. Since our database uses a good number of tables, our ER diagram ended up being pretty big. We broke it down into 5 separate pictures for easier viewing.

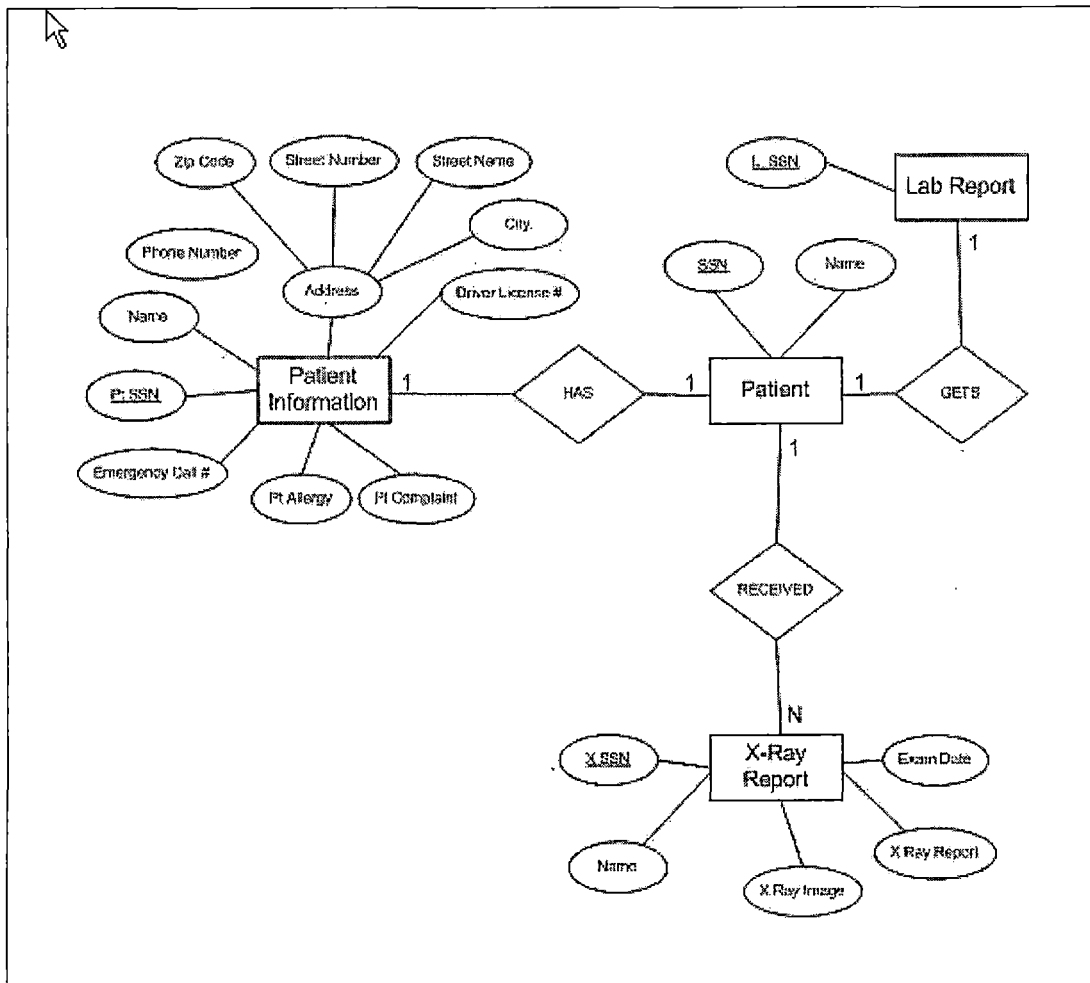


Figure 4. Entity Relation Diagram Part 1

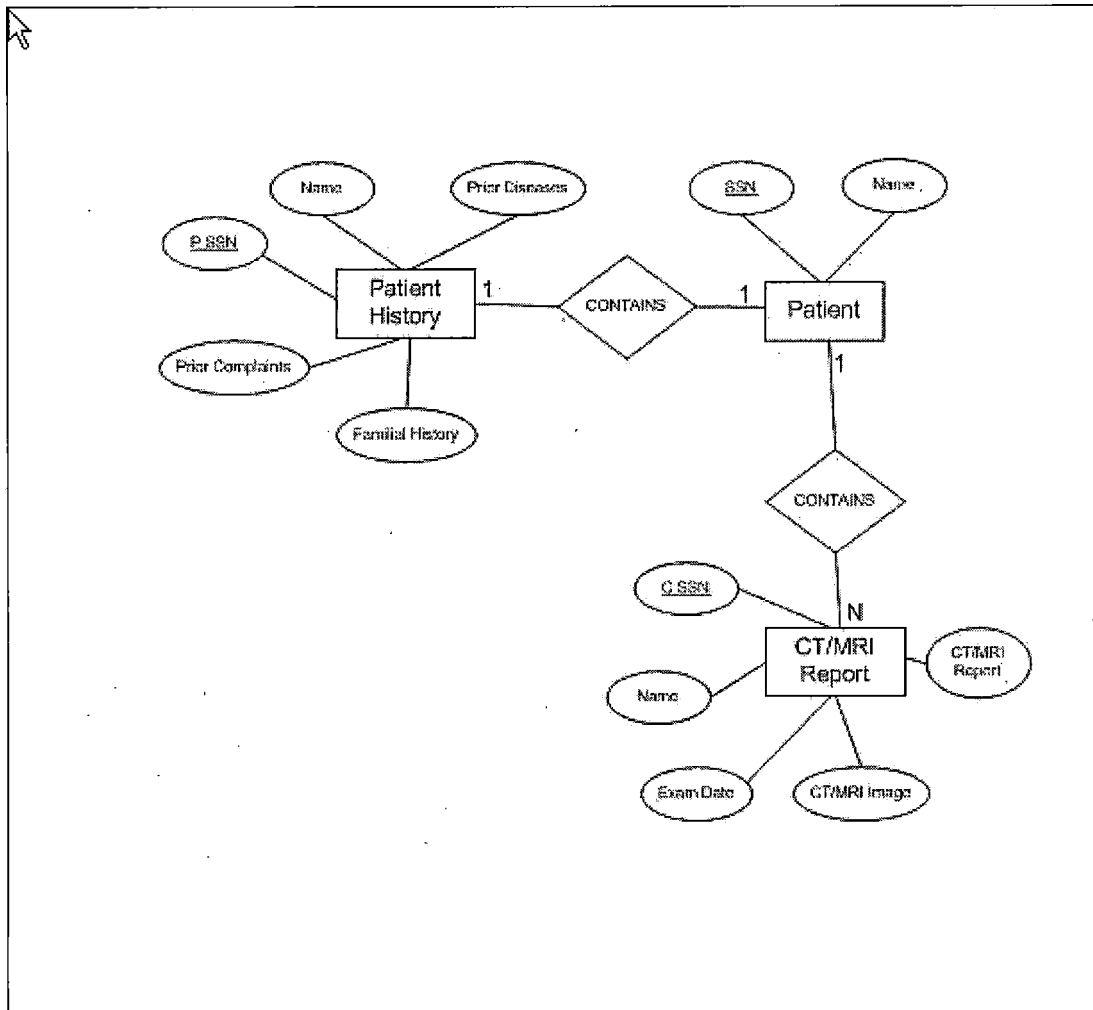


Figure 5. Entity Relation Diagram Part 2

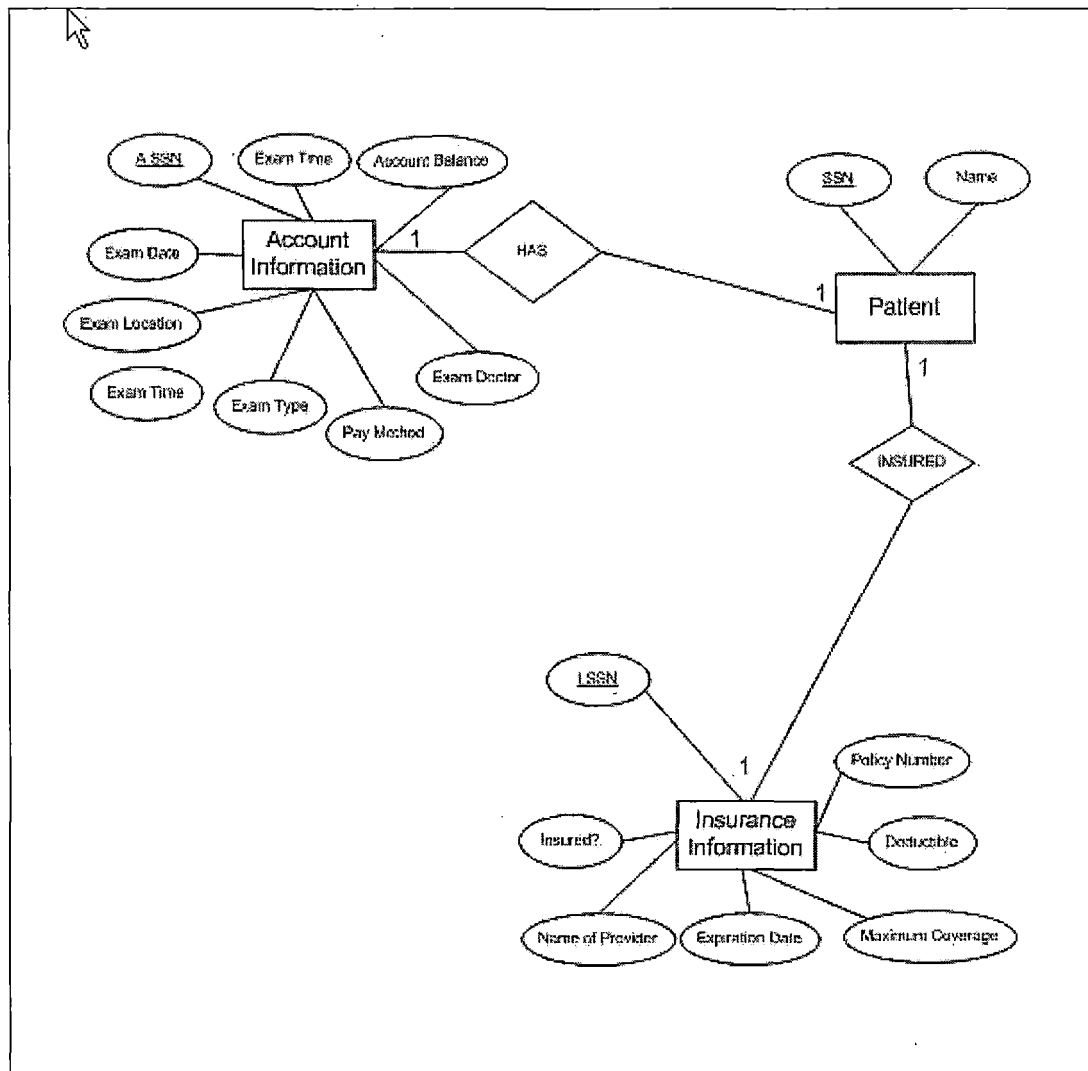


Figure 6. Entity Relation Diagram Part 3

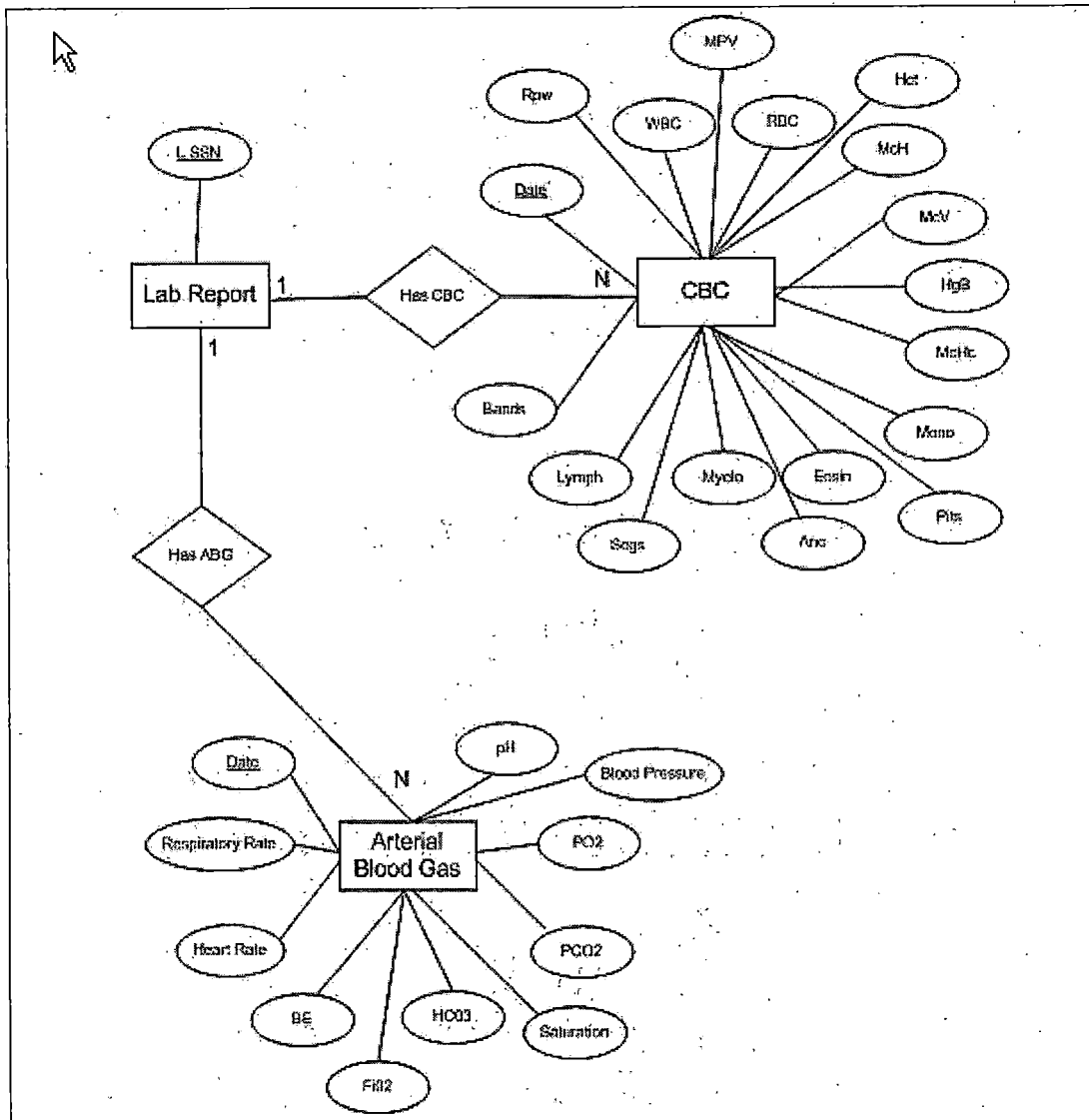


Figure 7. Entity Relation Diagram Part 4

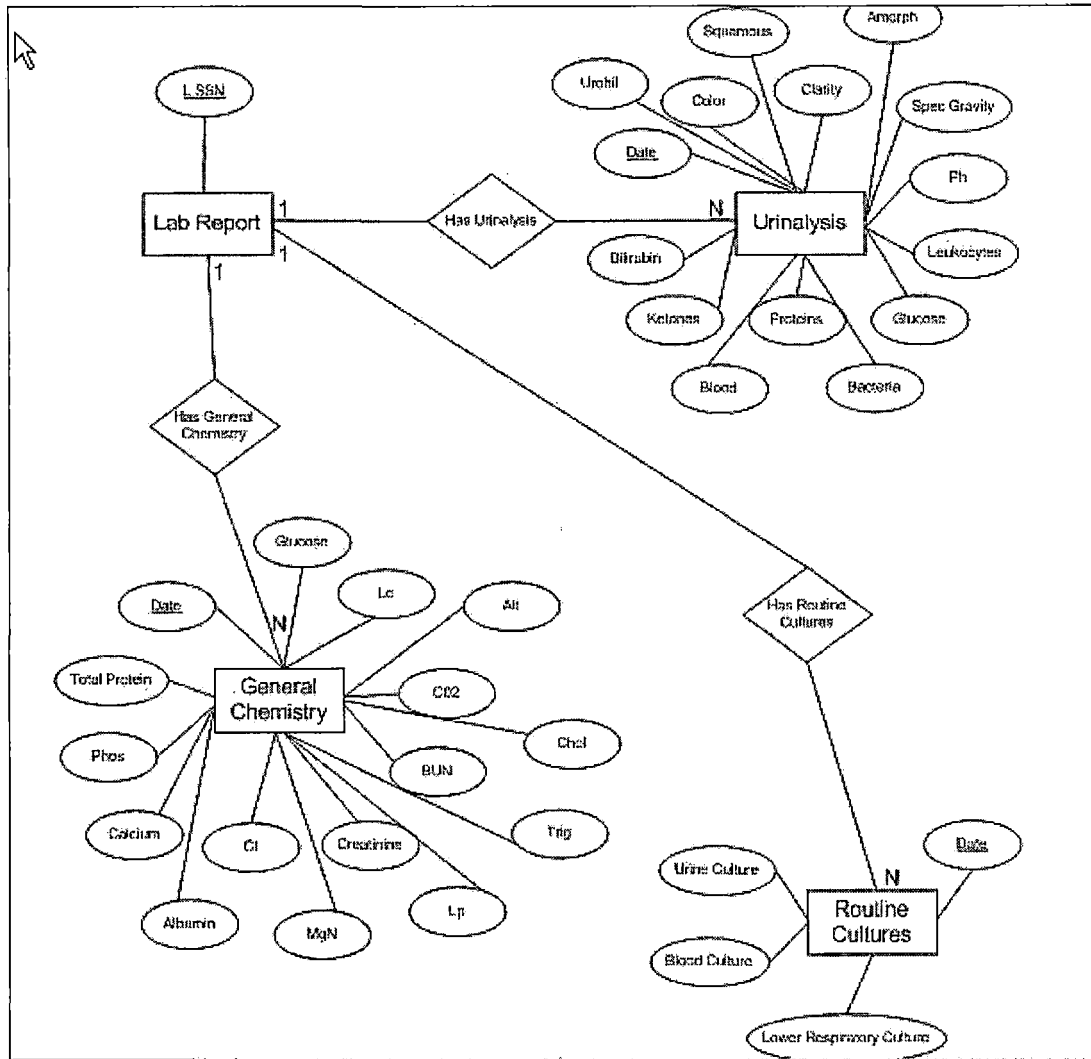


Figure 8. Entity Relation Diagram Part 5

1.7 Graphical User Interface

It is also useful for the reader of this report to be able to understand how to startup and operate the system without the help of somebody who actually developed the project. In this section of the report, we will present a "user guide" per say complete with screen shots and explanations so that the reader may better understand how

a user interfaces with a program and also how to manipulate and operate the system.

The first step of this system when starting up for the first time or just restarting after shutting down, (we don't anticipate the system ever being shutdown during normal operations as it will be continually accessible, except in the case of maintenance or upgrades) is to startup the Tomcat Server. We have on our desktop the shortcuts to the batch file that will startup the server as shown in figure 1.

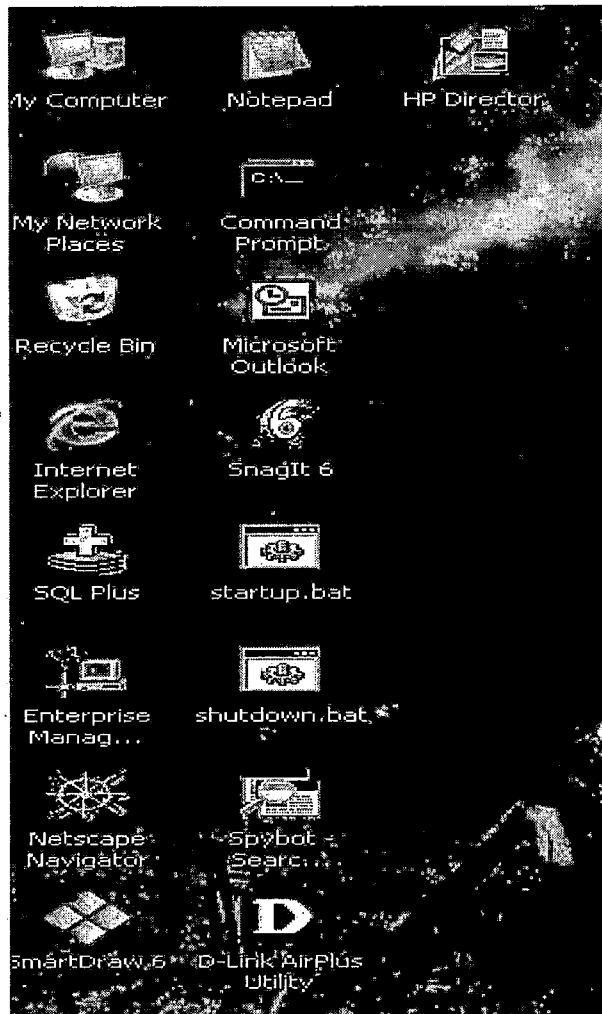


Figure 9. Desktop Icon of Startup.bat

By clicking on this icon, the Tomcat server will start to initialize. By doing this vital first step, the user will make their desktop computer into a server that will be able to accept and process web requests. It also will make it so that the desktop computer will be able to process Java Server Pages as well. Without the Tomcat Server up, the JSP code will not be compilable. The user will know if the Tomcat server is successfully initialized.

when they see this screen on their desktop as shown in Figure 2.

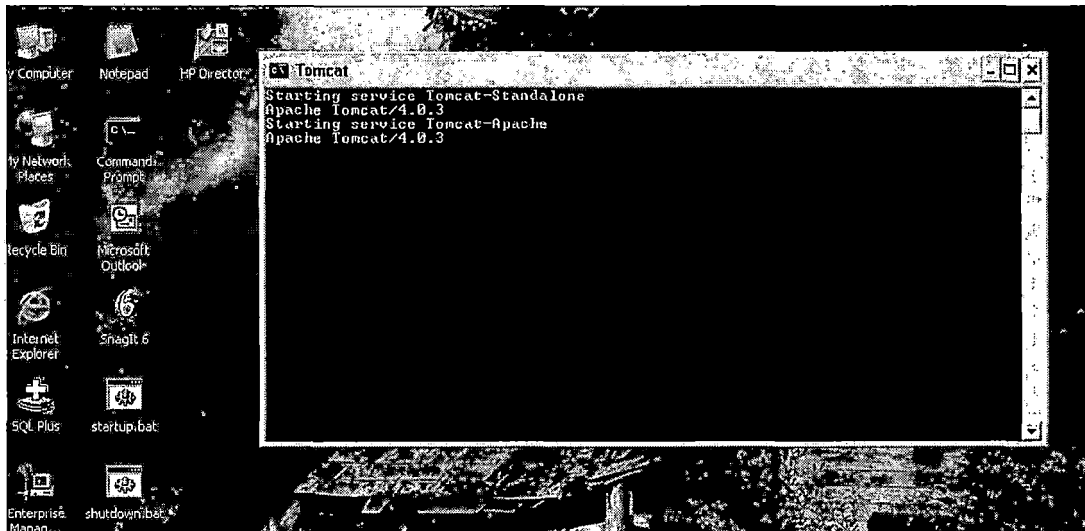


Figure 10. Screenshot of Successful Startup

The development team has received errors with this screen before where it showed that it was unable to find certain files or wrong path. In these situations, just make sure that the path specified by the Startup batch file is correctly pointing to the path in your directory.

The next step in the process is to startup the Enterprise Manager that comes with the Oracle 9i database. This Enterprise Manager is the main interface that the user will use to startup or shutdown the database itself. Since this system is used by people accessing it from the internet, this application is used only by those on the actual server/network the system is on and not accessible

via the web. This program is used to access and manipulate the database of patient information itself. We has database administrators (DBA) have full access to create/delete tables, shutdown or startup the database. However, when actually used out in public, certain users will only be granted ability to manipulate the data inside the tables but not create new tables and certainly not to shutdown/startup any databases. We will, in this guide, use the DBA access so that we may better show the reader how to use the system. The Enterprise Icon on the desktop is as follows.



Figure 11. Icon of Enterprise Manager

After double clicking on the icon, the user will see this screen.

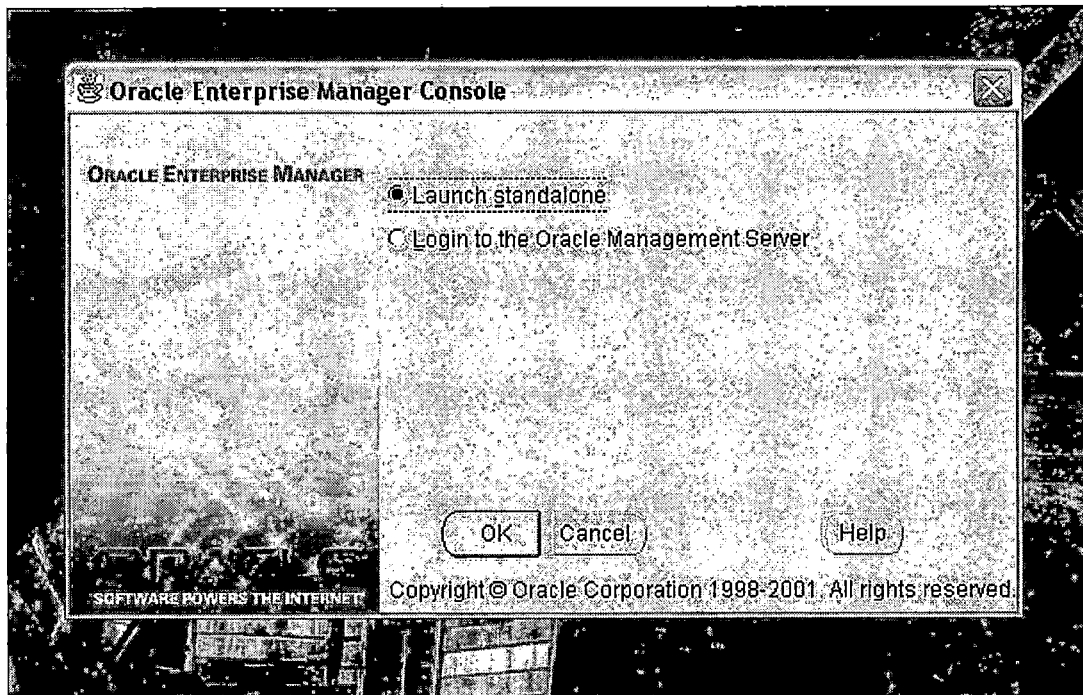


Figure 12. Enterprise Launch Screen

After seeing this screen, it is important for the user to choose the standalone version as logging into the other option would not lead to the correct results. By clicking on the "OK" button, the user will be then be allowed access into the Enterprise Manager utility. However, this does not mean that the user has access to manipulate the database. This screen just lets the user to see what databases are on the server at the time and also which ones are accessible by the user. Figure 5 shows the

screen the user will see when logging into the standalone version.

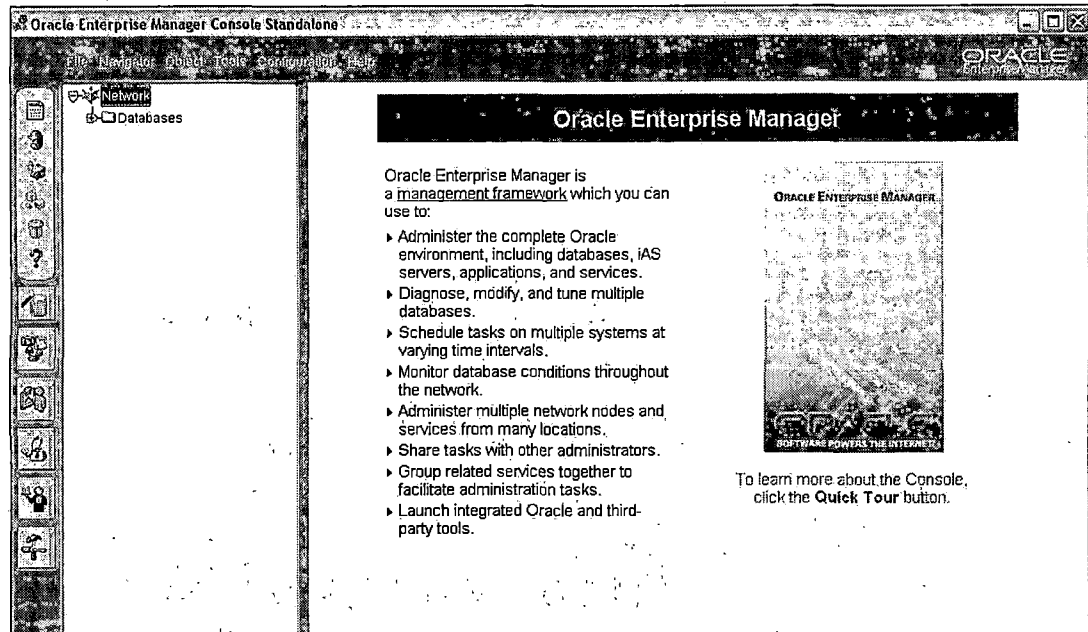


Figure 13. Initial Logged-in Screen

As the user can see from Figure 5, this screen gives a brief overview as to what functions this utility can do such as administer the Oracle environment including databases. It also allows us to diagnose, modify and tune databases for better performances. Although the buttons are accessible to the user at this point, when a user tries to actually do any of the aforementioned things (or any of the other functions for that matter) they will be prompted by the utility for their username/password. Since we do want to modify our database, this is our next step. In the upper left side of the Figure 5, the user should

see a hierarchical tree that just says databases. This interface is similar to any directory based utility that allows the user to explore the contents of like say a hard drive such as Windows Explorer. After clicking on the hard drive icon, the user will see all available databases on the server. As of now, we only have the one that we created named "ANDREW" as seen in figure 6. When the user clicks on the ANDREW database, the application will assume the user wants to access the DB and asks for the id/password.

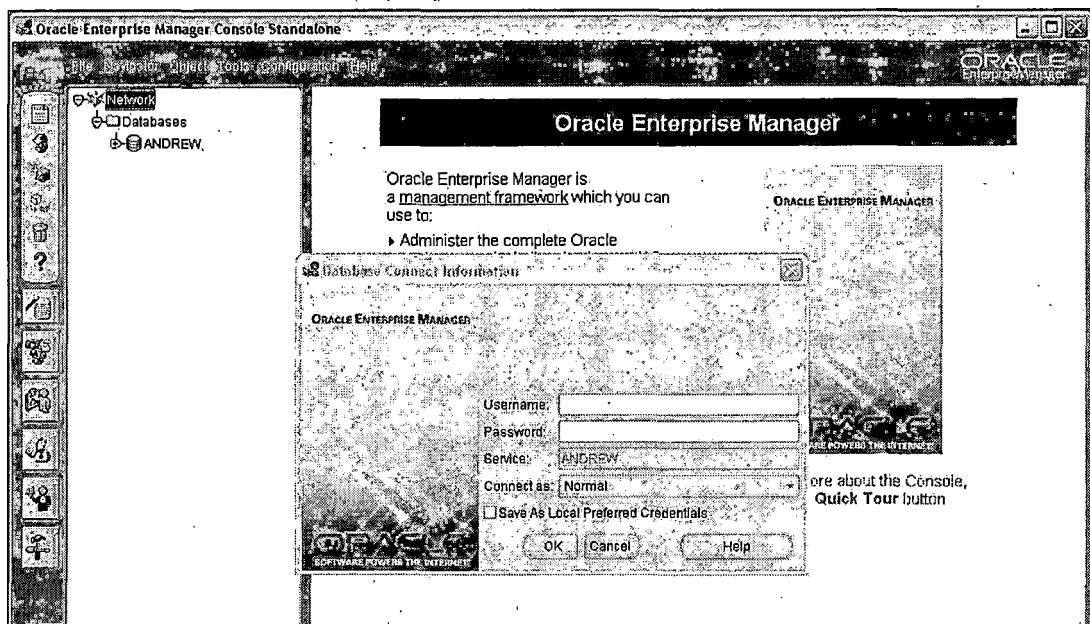


Figure 14. Login Screen to ANDREW Database

Notice at the bottom of the login screen has the option for the user to login as different roles. For our purposes, we want to choose the role of Database

Administrator (DBA) since we want full access. As mentioned before, roles can be assigned according to the need of the user. We would normally only allow DBA privileges to maintenance staff and not to normal clerical staff that would be normally entering and deleting patient information. So after filling out all the appropriate fields and choosing DBA as our role, the screen should look like this.

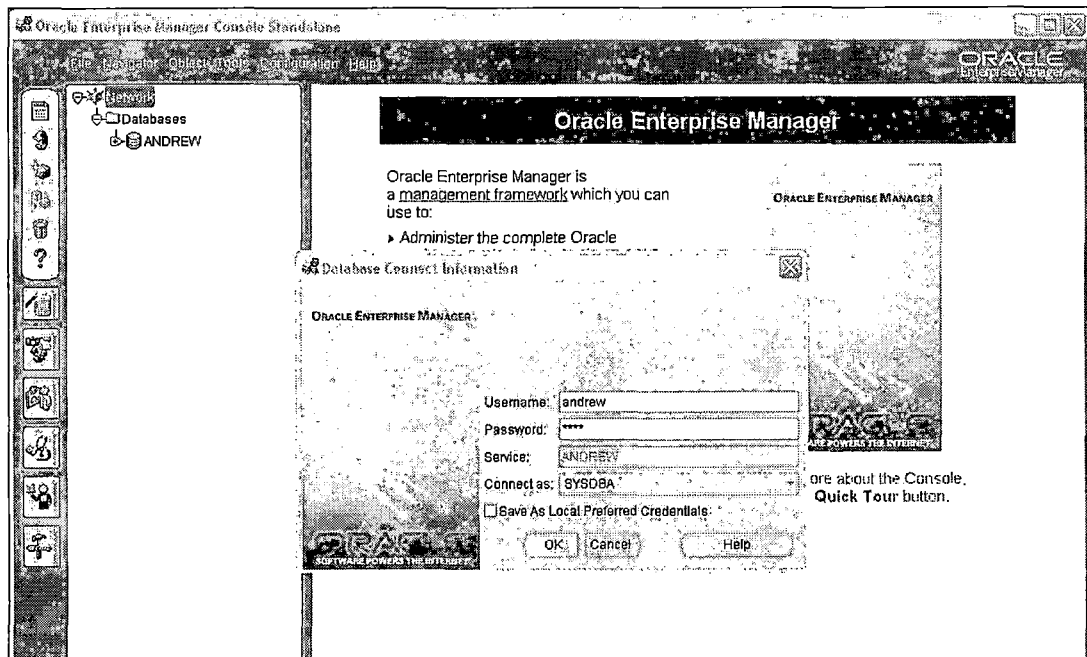


Figure 15. System Database Administrators (SYSDBA) Login Screen

Upon logging in, now the user has rights to delete/add tables along with pretty much else anything you want to do to the database including deleting it.

Now the user will see under the database ANDREW tree many subtrees. The subtree that we are most interested in is the "Schema" subtree. In this subtree, upon clicking it, we have access to the tables that are in our database. Oracle 9i database application came with many of its own tables including some examples used for its built-in tutorials and others that are actually vital to its operations. The screen will look like this as seen in Figure 8.

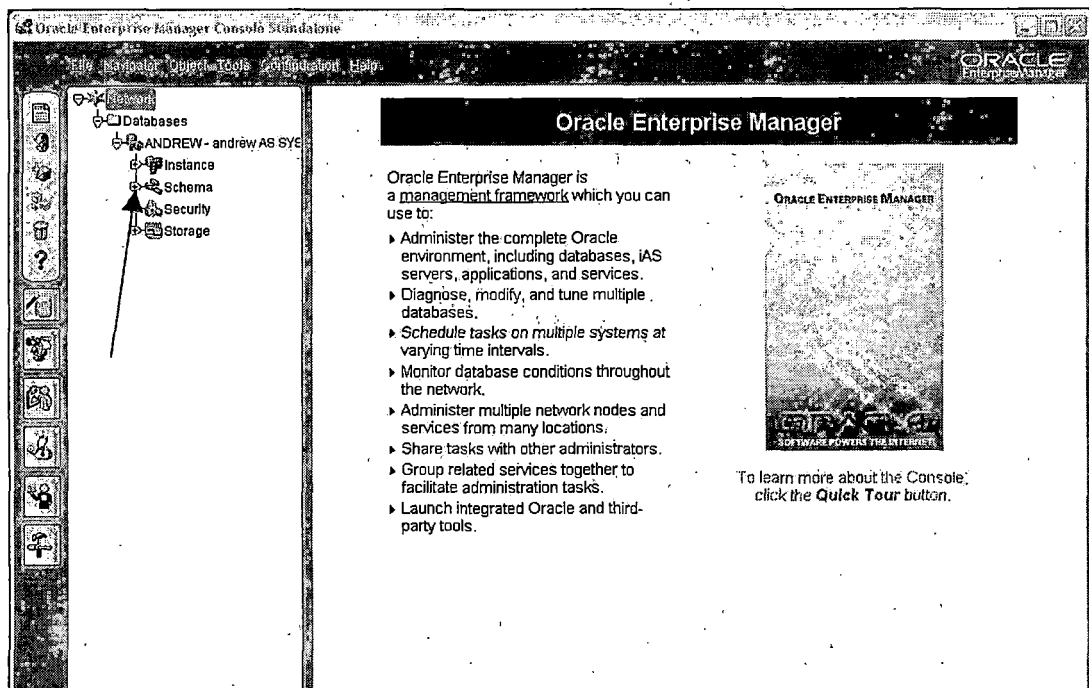


Figure 16. Database ANDREW Schema

To access our tables that we created for this database, we need to click on the schema button which will drop down a list of additional subtrees. We want to go to

the table subtree on the lower left hand side. After clicking on the Table subtree, the right hand pane should show a list of all the tables on the server sorted according to who created them and what database they belong to. In our case, we had a user named "LI" that we used to create and edit the tables.

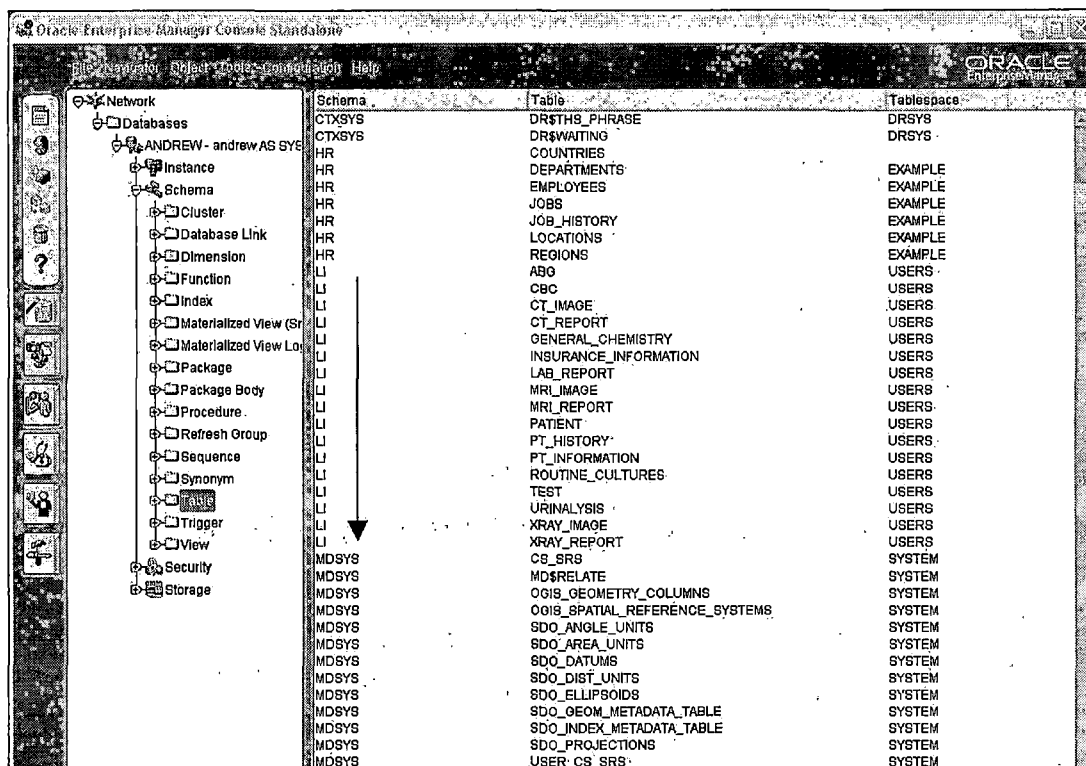


Figure 17. Table Subtrees

Notice how certain tables are created by the system while others are created as examples while still others such as the LI tables are created by users. From here we can access the tables that we created. Normally users will only be able to access the tables for the database that

they accessed. We are not sure, if the utility can further limit the users to only be able to access the tables that they themselves created within the database. Of course as SYSDBA we are currently allowed to access all the tables. Now a normal user will most likely want to access a certain table and either add, modify or delete patient information as the situation demands. To do this, we highlight any one of the LI tables and do a right click. It is important to note here they the user cannot access the table editor by double-clicking on the table as this will only bring the user into a screen that lists the characteristics of the table such as attributes as seen on this screen shot.

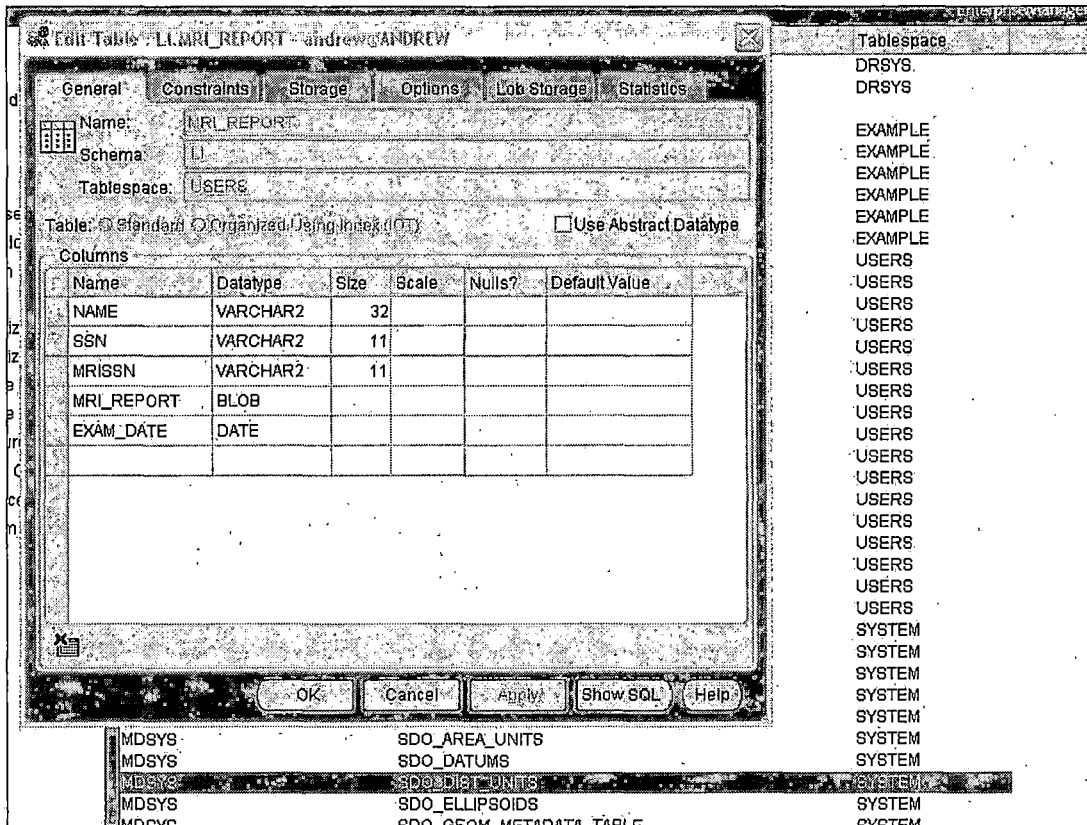


Figure 18. Table Attributes

By doing a right-click on the table and then choosing "Table Editor" we are able to access this screen. In this screen, we access the "Lab_Report" table using the table editor. On this screen, we see some lines already inputted. To add more data into the table, just click on the next blank line and add in new patient information. This negates the need for the user to know any SQL commands to add data to a table. This is ideal for a user such as clerical staff or a data entry clerk that does not

know SQL but needs to enter patient information into the database.

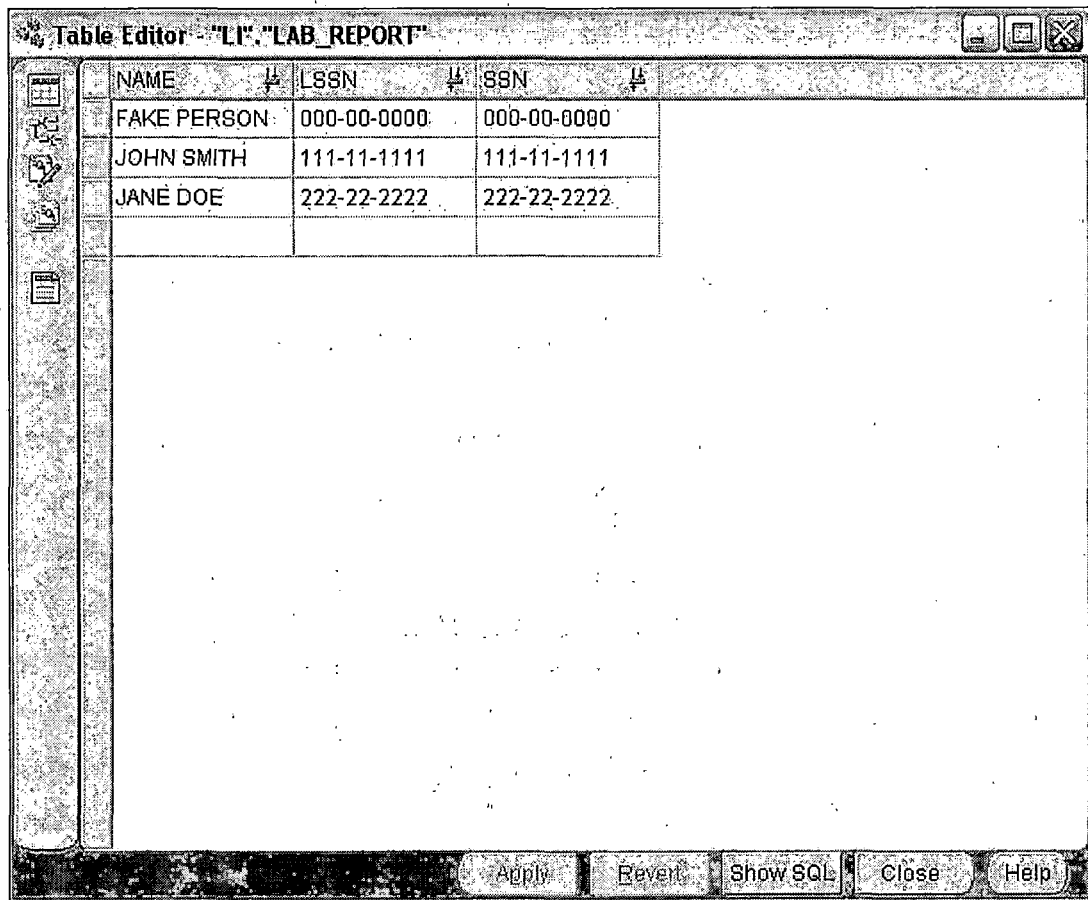


Figure 19. Table Editor Screen

After entering the desired information, the user needs to click the "Apply" button to ensure the data is saved into the database.

This is pretty much the extent a regular user will do with the Enterprise Manager. Of course, there are plenty of other functions the Enterprise Manager performs but tasks such as rollbacks, tuning and administrative tasks

are reserved for the maintenance team and should not be of concern to the lay user.

Now that the table accessing and data entry steps are explained, next comes the web end of the WACPINS. As stated before, WACPINS will be accessible via the internet. We have purchased an internet domain site named www.ptanywhere.com. It is just a faceless site that serves mainly as an access point that points to our server at home. The registrar that we bought the domain from provides rudimentary Domain Name Server (DNS) functions but they serve only as placeholders as they only point to themselves. We used a free service called ZoneEdit that allowed us to use their DNS servers to point to our IP address. A more detailed explanation of this process can be found in another part of this report.

The first step on the internet side of things is to access our system via the web. It will work with either Internet Explorer or Netscape Navigator but for some reason, it works more reliably with IE. We think it is that we have an older version of Netscape that is not compatible with more contemporary HTML code. Since we are at our home computer as we are writing this part of the project, we can access our own server in two different ways. One way, of course would be to go to the internet

and type in `www.ptanywhere.com` and access it like that. Another way would be to access it via the intranet by typing in "localhost" in the web browser. Depending on how the server was set up, sometimes you would need to type in `localhost/8080` where the "8080" is the port number used to access the system. For our system, we set it up so that we only have to type in "localhost" to access WACPINS. This is shown in the screenshot.

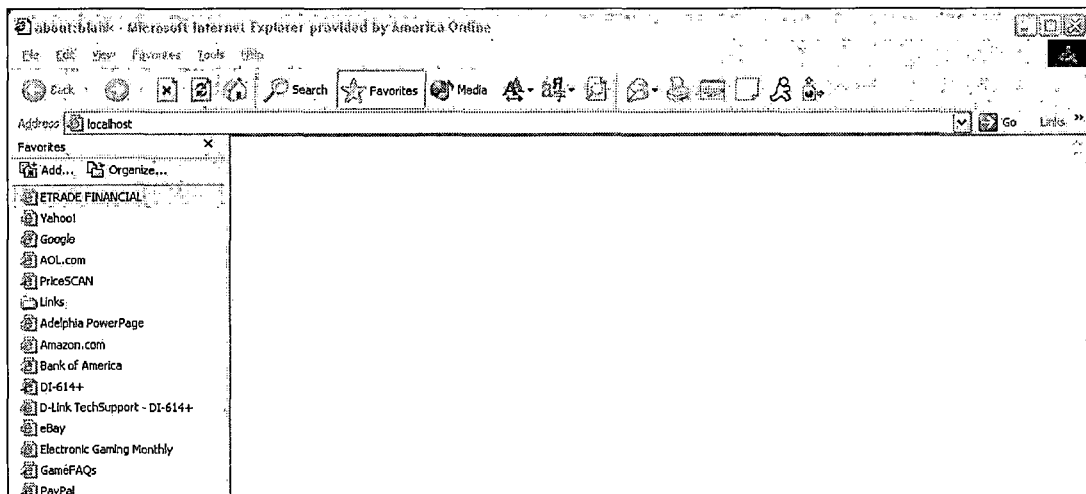


Figure 20. Localhost Intranet Address

After typing the address in, the browser should take you to the welcome page of the WACPINS system. You will know if it worked if you see this page.

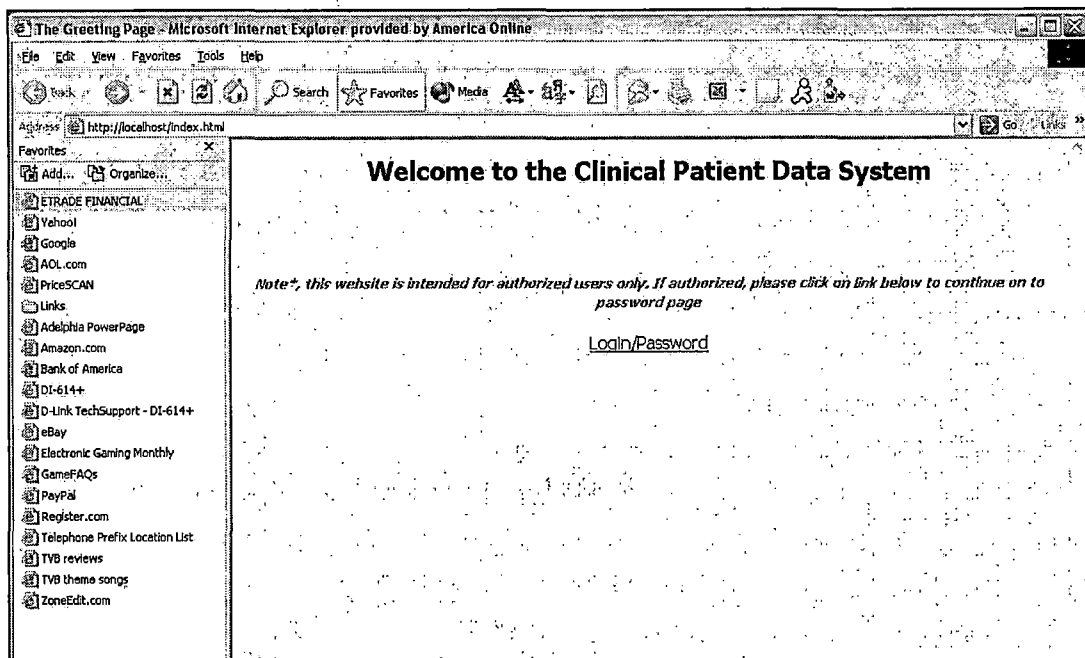


Figure 21. Welcome Page

On this page, you will view a welcome message along with a warning message to unauthorized users. There is also a link to take the user to a login page that will securely let them into the WACPINS system. As of this time, we do not have a way to set privilege levels according to who logs into the system via the web. Everyone is able to make queries and retrieve data from the database. No one is allowed to update, modify or delete information from the database via the web. These functions are only able to be performed from the Enterprise Manager by a qualified user at the actual site of the server. When the link to the login screen is clicked, the following page will appear to the user.

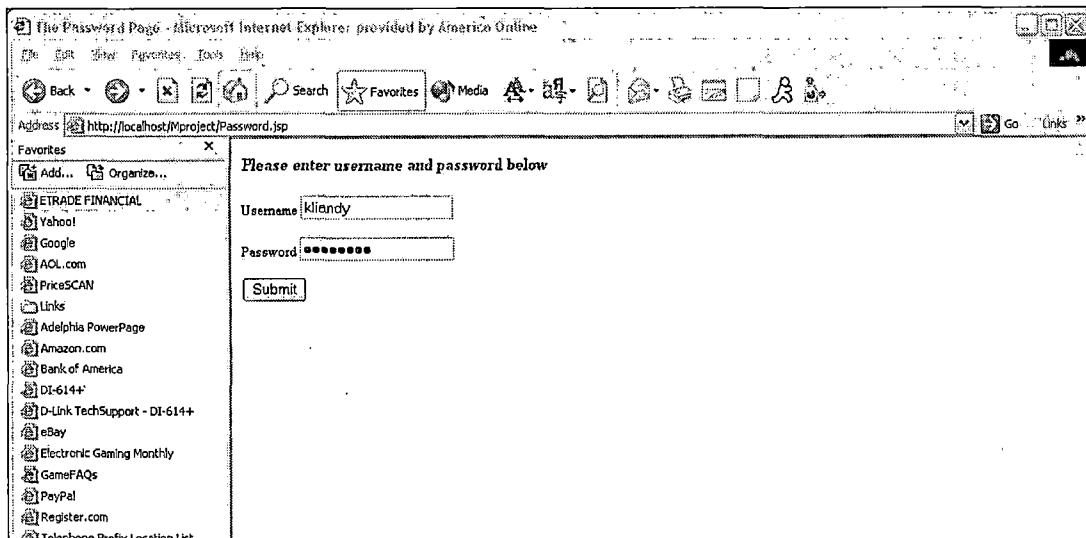


Figure 22. Web Accessible Clinical Patient Information Networked System (WACPINS) Login Page

This page is pretty straightforward in that all you have to do is enter a user ID, password and hit the submit button. Currently, the only way for new users to be added on is for them to make a request via e-mail to the DBA and the DBA will have to create an account for the user.

If successfully logged in, the user will be brought to the main selection screen. From this screen, the user will type in the patient's social security number as a means for identification and also select the kind of lab the user wishes to view. The updating of the database is done in real time so as soon as the data entry staff enters the information, it will be available to the doctor, nurse etc. accessing it from the internet. The main selection screen looks like this.

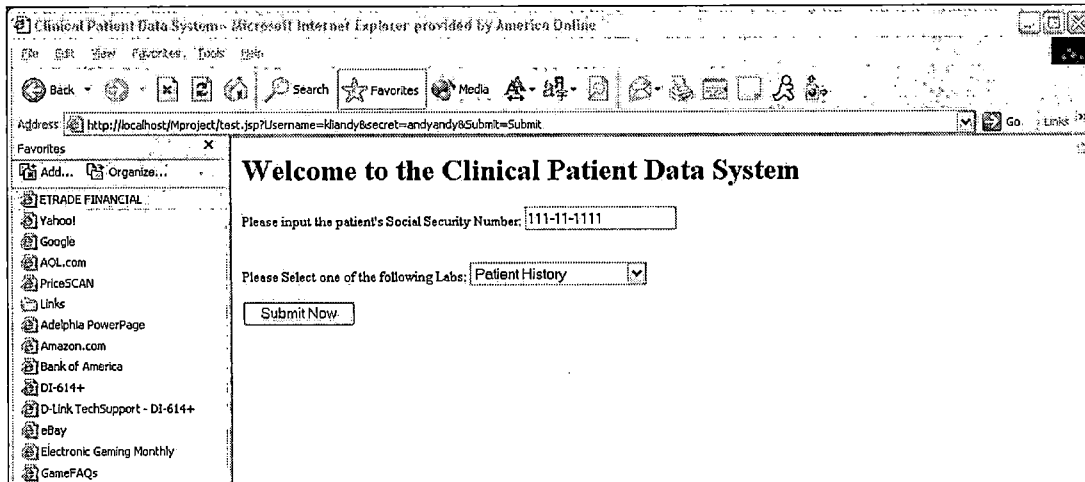


Figure 23. Main Selection Screen

The dropdown box provides a choice of all the labs that are available from the clinic. If a user chooses a lab that has not been performed on a patient, he/she will receive a blank screen in return. Also, the entering of the social security number must be precisely xxx-xx-xxxx to ensure that a patient's lab results are actually selected. If the social security number is entered erroneously, the database will return an error and go to the following error page.

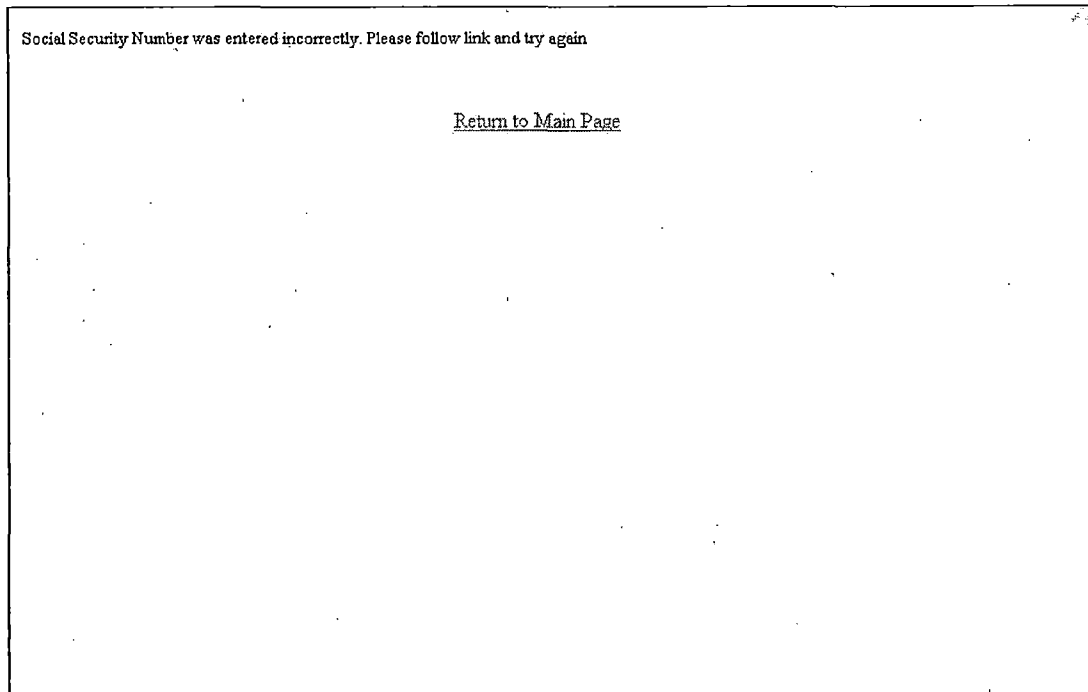


Figure 24. Error Page #1

This page provides a link that will take the user back to the main selection screen again or the user can simply hit "BACK" on their browser.

When a user successfully enters the correct Social Security number and the valid lab entry, a query will be made to the database that will return all results ever performed for that specific lab for that specific patient in the patient's history with that particular clinic. The results are sorted with the most current one on the bottom and then ascending in order of dates. This is done so that clinicians can easily see if there is a trend that they should be aware of. Often, patients present trends in

their lab values that can only be recognized upon repeated serial tests. Having the entire battery of results is also useful for determining a patients history of illness. A successful database query will return a page similar to this. For this example, we used the Urinalysis test for show.

Laboratory Results							
Test DATE	COLOR	CLARITY	SPEC_GRAV	PH	GLUCOSE	LEUKOCYTES	KETONES
1999-09-16 00:00:00	white	clear	8	7.15	130	16	8
1999-09-17 00:00:00	red	cloudy	11	6.54	100	18	7

Figure 25. Urinalysis Results for a Patient

It should be noted on Figure 17 that not all the lab values were shown on this screenshot. The amount of fields in that lab test exceeded a single page so the user would need to scroll to the right to view the other parameters.

There is a difference when it comes to images such as X-Rays and MRIs however. This difference is mainly how the tests are selected from the main lab selection page.

During development, the development team ran into a problem in which we had an extremely hard time figuring out how to integrate image files into Oracle. We finally found an alternate method in which we stored the images outside the database but still used JSP queries to locate the directory they are in and display them. Again, a more detailed explanation of the problem is found in the Methodology section of the report. Suffice to say in this section, we hope to find a more elegant solution to the image problem in the future that time constraints did not allow this time.

Upon selecting either MRI, X-Ray, or CT images/reports, the user is taken to another screen that will prompt them for the patients initials and last 4 numbers of the social security number. Here is how the page will look like.

Welcome to the XRAY Image page

Please enter the initials of the patient first name first. For example if patient name is John Smith then enter JS. Also please enter the last 4 digits of the patient's social security number. So JS0000 would be an example.

Patient Initials and last 4 digits of Social Security number:

Figure 26. Secondary Lab Selection Page

Again, an explanation of why this is required is explained in the methodology section. If the user enters the information incorrectly or no such patient exists, then the WACPINS system will return this page.

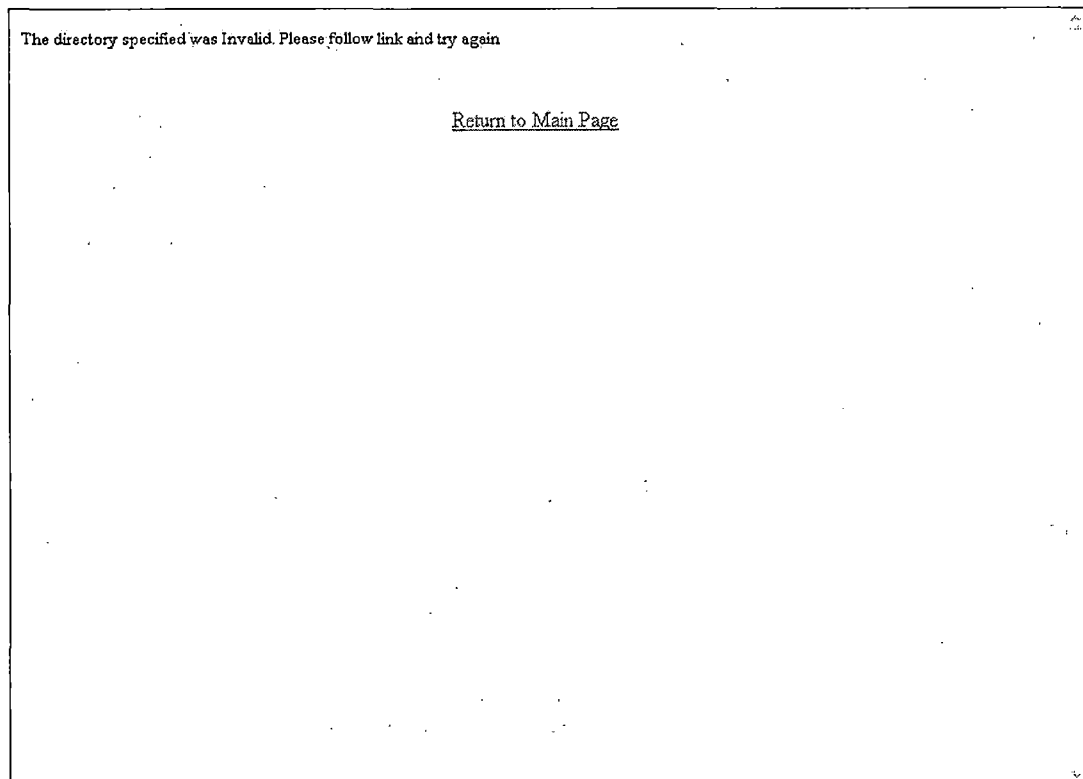


Figure 27. Error Page #2

As the reader can see, there is a brief explanation of the error and a link back to the main selection page. We thought about bringing the user back to the second selection page for image/reports only but we reasoned that they may have selected an invalid test for the patient and therefore may want to pick another test. However, upon more testing, we can easily change that if necessary.

As alluded to before, we have two separate sub-categories for each MRI, CT or X-Ray category. They are reports and images. Reports are like the radiologists verbal diagnosis that has been transcribed onto paper by a

secretary for records purposes. These include all prior reports that have been done on the patient since the patient has been with the clinic. This provides the trending aspect as mentioned before. Images are exactly those. Images scanned into the computer and then saved as .jpeg files. Here is an example screenshot of how a successfully retrieved X-Ray image page will look like.

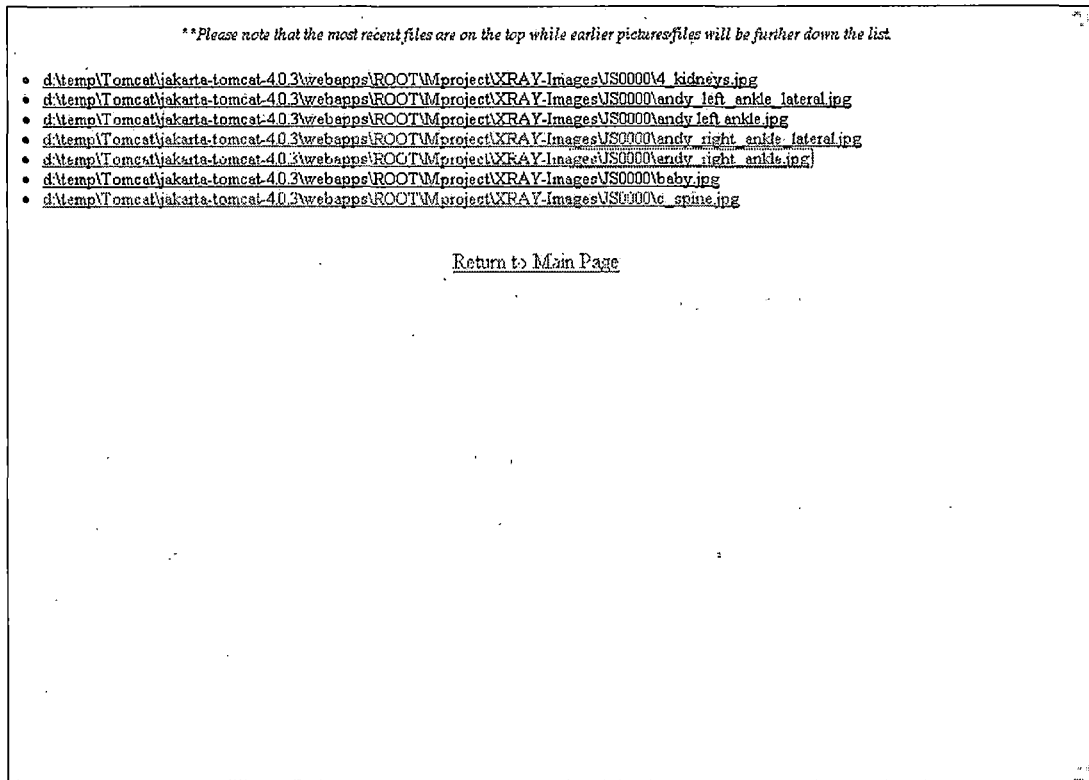


Figure 28. X-Ray Image Page

It is important that whoever is saving the images to the hard drive give a detailed explanation of the file so that it will be easier to locate and retrieve for the

user. The most recent files are located at the top and then descending in order of dates.

We used a high resolution scanner to scan real MRI, CT, and X-Ray images to the computer. These images are of actual patients courtesy of Li-Kwok Liang in his capacity as a CT/MRI specialist. Here is an example of one of the images that are contained within the system.

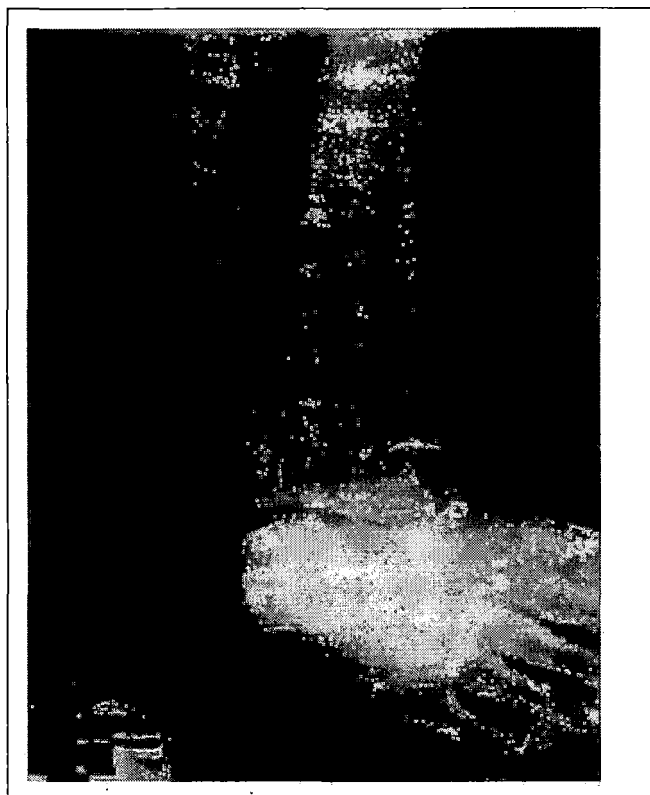


Figure 29. Image of a Wrist Fracture

The user is able, after retrieving the picture, manipulate it in anyway a .jpeg file can be played with including zooming in and out, adjusting contrast and make

notations onto the image for physicians who want to point out something important.

At this point, that pretty much brings the reader to the conclusion of this "user guide." The only step left to do would be to shutdown the server. It must be noted, however, that this step will not always be performed but there are times when it will be necessary to shutdown the server so we will also go over these steps as well. The shutdown sequence is very similar to the startup sequence in that there is a Shutdown.bat file on the desktop that will perform this operation.



Figure 30. Shutdown.bat File

After doing this, you will get the same window the Startup.bat used but the message will instead say something to the effect of "shutting down the Tomcat Server now." Once this is done, the computer is once again a regular computer.

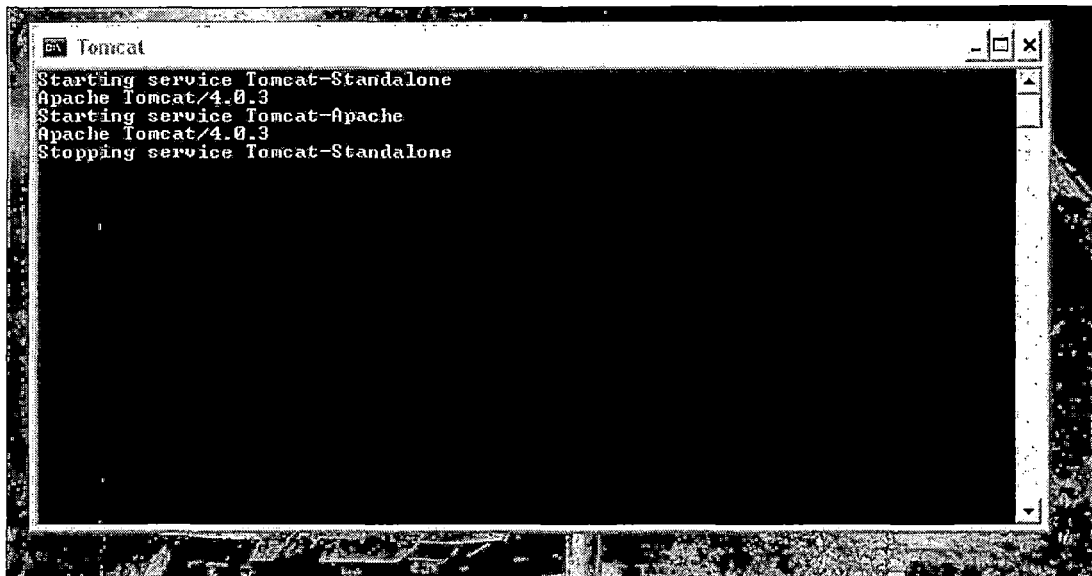


Figure 31. Shutdown Sequence

This concludes this section of the report. A more detailed explanation of the inner workings of the WACPINS system can be found in the Methodology section in Chapter 3.

CHAPTER TWO

LITERATURE REVIEW

2.1 Review of Related Literature

There was research done on this subject as to what products were out there in the clinical area. We both work at different hospitals. One of us works at Loma Linda University Medical Center and the other works at Desert Hospital. We both use the computer system there extensively and as such have a good idea of what is used. Upon further research on the internet, we found that there are other systems out there that also do what is similar to what we have at our respective hospitals. As of now, we have yet to see any program that integrates the storage of images along with regular text data.

All the products so far are very expensive and can hardly be justified for a private practice that simply will never use all the bells and whistles of those professional programs.

Another aspect of our research into related topics, we did a clinic to clinic survey of Inland Empire private practices many of whom are personal friends. We found in our survey that most if not all still use a paper filing system to store all their permanent records. These records

basically are all relevant patient data such as medical history, prior lab tests, insurance information etc. They indicated that if there was a program out there that would help them streamline their filing system and cut down on all the paper work, they would be interested.

The books that we used for our projects are listed in the "References" section. Here is a brief description of each and how they helped us in our project. The two volumes from Oracle University "Introduction to SQL/PLSQL" [2], [3] written by Kochar, Gravina, and Priya were used to construct the SQL statements needed to develop the database. The book "Java Server Pages" written by Whitehead [1] was a book that helped us mainly in developing the JSP code. It was extremely helpful when it came on how to manipulate the database from a web page. Another book that we used was "Pure JSP" written by J. Goodwill [4]. This book was not as helpful as the prior book but served as an ancillary book when more in-depth programming was needed. This book was not as easy to understand but provided more advanced techniques to program in JSP. "Oracle 8i on NT" [5] was basically just a brief guide and overview to Oracle 9i which we actually used. We bought this book mainly to have a reference to older techniques that was used on 8i but still applicable

to 9i. The last was actually a web site that we used called "Configuring and Using Apache Tomcat4" written by M. Etapi. This was obviously used to setup the server side to be able to process web and JSP pages.

CHAPTER THREE

METHODOLOGY

3.1 Methodology

The language used to produce this Clinical Patient Information System was Java Server Pages. Again, JSP is a language that combines the Java programming language with HTML. We chose JSP because it would be easily portable and also because we were somewhat familiar already with Java and HTML. However, we were unfamiliar with the method on how to combine these two programming languages to make a web page therefore we bought some material that gave us some pointers. The books described in the previous section were useful in helping us setup the web server although we used a website tutorial that was even more helpful as well.

The database was the first question I tackled. I had several choices for databases such as Microsoft's database program or MySQL which is free to all users. However, I was more inclined to use Oracle's database system since that was what I used for two of our database classes. However, I thought that I would have to buy the program since I didn't think Oracle was going to give it out for free. To our surprise, it was available for download at

the Oracle web site. In fact, they even had a home version of Oracle 9i available for download too. Although the download took a long time, the entire program worked fine after installation.

After choosing the database we wanted, the next step was to tackle the problem of how to connect the database to the internet. My partner Li-Kwok was responsible for this part. He took care of purchasing a web site from an online dealer which he set up to connect to our database via the web. He installed and set up the Tomcat Server program on our computer. He also installed the necessary JDK components so that our computer will be able to program in Java. The Tomcat Server enables our computer to understand and compile JSP code. All this can be referenced in Li-Kwok's paper.

The next step was the problem of how to get the website to "point" to our computer. Li-Kwok also took care of this part in which he used a free service called ZoneEdit that allowed us to use their DNS servers at no charge. With this service, we were able to customize the DNS server for our website to make it point to our IP address. After this, I then went on to the problem of which operating system to use.

The operating system I chose for this project turned out to be Windows XP Professional. I had discussed whether I should use XP since it is relatively new. I thought Windows 2000 Professional would be a better choice since it had been out in use for a longer period of time. However, I had been using XP Pro on our computers for a while now and I found it to be extremely stable with crashes almost nonexistent. With Windows 2000 Pro however, I found that crashes was not happening often but at a rate much higher than XP. I also chose XP because it turns out that after installing the Oracle database system on our server, it really bogged things down. If it wasn't for a relatively fast chip and a generous amount of RAM, things would not have been good. Therefore, I chose XP because it was benchmarked to be 20% faster than Windows 2000 in executing programs.

For the HTML coding portion of JSP, I used our prior experience in coding with HTML but also needed a bit more than that. I again purchased a book called "How to do Everything with HTML" by James H. Pence. In that book, I found out how to do more advanced things with HTML and also learned how to use Cascading Style Sheets (CSS) which, according to the book, is the next generation of web programming language that will eventually replace HTML. It

is much more versatile and robust but the only problem with CSS is that it is not supported by the majority of the browsers out there. Hence our decision to stick with HTML since it is the current standard and can be viewed on virtually all web browsers.

Another aspect that I thought was important was the error catching mechanisms that ought to be in place in case of errors. I looked over our whole system and decided that there were only a few select places where errors would come up in during usage. One of those places I decided was when the user entered their username and password to access the system. For this, I made sure that if the user did not enter the correct password or username, they were always sent back to the password page. There was also an error message generated that told the user that their password or username was invalid. However, I did not limit the amount of tries that the user can enter their username/password. Another of the areas in the system I felt the user might generate an error is when they are entering a social security number that is not in the correct format. For example, if the user entered a identification number that was too long or too short, an error message will be generated and the user will be sent back to the page to enter the social security number

again. However, if the user were to enter a social security number that was of the correct length but just invalid, the system would just return no results since no patients were found by that number.

As for other areas for error checking, I felt that it was not necessary at this point to implement them. I felt the only points that could possibly generate errors was where the user can interact with the system. So in other words, only in the areas where there is user input. It should also be noted that I did not feel it was a good idea to allow updates to the database over the internet. For one thing, I felt the updates should only occur at the doctor's office where the new labs, x-rays, etc. show up. We as a team decided that it was too risky and posed too much of a security threat to allow updates over the internet. Therefore I only allow queries to the database over the internet while all updating to the database will be taken care of by secretarial staff in the doctor's office. Such staff will have to be specially trained to use Oracle's Enterprise Manager to update specific patient's data. As I found the Enterprise Manager very easy to use, minimal orientation to the system is needed and anyone with an inkling of computer know how will be able to use it with little or no problem.

In this part of the Methodology section, we will describe the actual structure of the project. The project is again, a clinical patient information system. It is a database that stores patient information and which is also accessible over the internet. When the user first signs onto the program, he or she will be prompted for a username and a user password. These will only be given out to approved users such as physicians, nurses or other clinic staff. Those who get passwords will be determined by whomever purchases the program which will most likely be the physician. If the user enters an incorrect password or incorrect user name, the user will have 2 more opportunities to sign on. After an unsuccessful third try, the system will kick the user to an error screen with a message to see the program administrator for a password or account. Then after the user signs in, the user will be presented with a screen that will prompt for the patient ID (SSN) and the patient last name. If the user knows the correct information, he/she will be taken to the screen that will give a list of choices to the user such as what kind of labs to bring up or what kind of information the user wants to see. When the user clicks on a certain lab such as ABG for example, the program will search the local database for the appropriate patient information and

display it to the user. If the user enters an invalid patient ID, he or she will be given an error message and be taken back to the patient ID entry screen.

After the user gets to the lab results display screen, the user can read the information and then when finished, click on a link to bring them back to the lab selection screen. The user will also have the opportunity to sign off as well from the lab selection screen.

In the end however, it is the result to the end that matters most. Therefore, to test the final results, which is essentially that WACPINS does work over the internet and that the database is accessible, we set the system online and made it accessible via a personal web page we bought and created. We got a few friends from school to access WACPINS from their homes, at school and from wherever else they can. They were all able to access the online system with no problems. However, there were problems with speed as too many people accessing the system at any one time caused it to bog down due to a slower upload from our DSL connection.

CHAPTER FOUR

CONCLUSIONS AND RECOMMENDATIONS

4.1 Summary

In summary, this project is a clinical patient information system. It uses as its core a database developed by Oracle Enterprises to store all information. Information such as lab values, patient medical history and insurance information. It also stores images such as X-rays and MRI images.

The programming language used is Java Server Pages or JSP. It is essentially a combination of Java and HTML. The web server program used was TOMCAT which is essentially a derivative of Apache and is developed by the same folks.

4.2 Conclusions of the Project

The idea of this project was to develop a software program that serves the needs of private practices which are essentially outpatient clinics. These clinics often do not need the extensive bells and whistles used by hospitals and thus do not need to spend so much money. Our project was based off the research done by asking local clinics in the Inland Empire as to what their needs are when it comes to storing patient data. The result is the program you have now. This program was developed with very

basic tools. All the important things such as the database system and the web server program were all shareware that anybody with a little patience can download. The cost was reasonable enough for two students on a limited budget.

4.3 Recommendations

One of our recommendations before undertaking a master's project is to choose a subject that truly interests you. You basically live and breathe this subject for however long it takes for you to finish it. If the subject is not something that has a true interest to you, it is that much harder to find the motivation to work at it everyday. It was lucky that both of us had medical backgrounds and was thus truly interested in this subject.

Another recommendation that we have is to keep in touch with all your friends throughout the program. This may sound trivial but one cannot discount the amount of help and input a friend can give you when you are stuck and the advisor is not available. Your friends/classmates are a great source of information because they may have remembered some information that you have forgotten and vice-versa.

Our last recommendation is to not be intimidated by the unknown. There were some times when we thought that a

certain aspect of the project was too hard because we had never seen or dealt with such things before. But with a little research and some hard work, nothing is impossible. We found the main problem that we had was that we were constantly underestimating ourselves in terms of our ability to grasp new concepts on the fly and integrate them with our current knowledge. And also, Barnes and Nobles is a great place for resources.

APPENDIX
SOURCE CODE

```
<html>
<head>
<title> Clinical Patient Data System </title>
</head>

<body bgcolor ="floralwhite">

<h1>Welcome to the Clinical Patient Data System </h1>

<form action = "projectform.jsp" method = "get">
<form action = "DB_CBC.jsp" method = "get">
<form action = "DB_CTREPORT.jsp" method = "get">
<form action = "DB_CTIMAGE.jsp" method = "get">
<form action = "DB_MRIREPORT.jsp" method = "get">
<form action = "DB_MRIMAGE.jsp" method = "get">
<form action = "DB_GENERALCHEMISTRY.jsp" method = "get">
<form action = "DB_INSURANCEINFORMATION.jsp" method = "get">
<form action = "DB_LABREPORT.jsp" method = "get">
<form action = "DB_Patient.jsp" method = "get">
<form action = "DB_PTHISTORY.jsp" method = "get">
<form action = "DB_PTINFORMATION.jsp" method = "get">
<form action = "DB_ROUTINECULTURES.jsp" method = "get">
<form action = "DB_URINALYSIS.jsp" method = "get">
<form action = "DB_XRAYREPORT.jsp" method = "get">
<form action = "DB_XRAYIMAGE.jsp" method = "get">
<form action = "CTREPORTPROC.jsp" method = "get">
<form action = "CTIMAGEPROC.jsp" method = "get">
<form action = "MRIREPORTPROC.jsp" method = "get">
<form action = "MRIIMAGEPROC.jsp" method = "get">
<form action = "XRAYREPORTPROC.jsp" method = "get">
<form action = "XRAYIMAGEPROC.jsp" method = "get">
```

Please input the patient's Social Security Number:

```
<input type = "text" name = "Social Security"><br>
<br>
```

Please Select one of the following Labs:

```
<select name = "User Choice"><br>
<option value = "ABG"> ABG </option>
<option value = "CBC"> CBC </option>
```

```
        <option value = "General Chemistry"> General Chemistry
    </option>
        <option value = "Insurance Information"> Insurance
    Information</option>
        <option value = "Patient History"> Patient History</option>
        <option value = "Patient Information"> Patient
    Information</option>
        <option value = "Routine Cultures"> Routine Cultures</option>
        <option value = "Urinalysis"> Urinalysis</option>
        <option value = "X-Ray Report"> X-Ray Report (text
    only)</option>
        <option value = "X-Ray Image"> X-Ray Image</option>
        <option value = "MRI Report"> MRI Report (text only)</option>
        <option value = "MRI Image"> MRI Image</option>
        <option value = "CT Report"> CT Report (text only)</option>
        <option value = "CT Image"> CT Image</option>

</select>
```

```
<br>
```

```
<br>
```

```
<input type = "submit" name = "submit" value = "Submit Now">
```

```
</form>
```

```
</body>
```

```
</html>
```

```
<html>
```

```
<head>
```

```
<title> Processing Page </title>
```

```
</head>
```

```
<%@ page import="java.sql.*"%>
```

```
<body bgcolor = "floralwhite">
```

```
<h1>Welcome to the Processing Page </h1>
```

```
<hr>
```

```
<%
```

```
String SS = request.getParameter("Social Security");
```

```

        if(SS.length() != 11)
        {
%>
            <jsp:forward page = "Reject.jsp" />
<%
        }

%>

<br>
<%
String ID = request.getParameter("User Choice");
%>

<%
    if (ID.equals("ABG"))
    {
%>
        <jsp:forward page="DB_ABG.jsp" />
<%
    }

%>

<%
    if (ID.equals("CBC"))
    {
%>
        <jsp:forward page="DB_CBC.jsp" />
<%
    }

%>

<%
    if (ID.equals("General Chemistry"))
    {
%>
        <jsp:forward page="DB_GENERALCHEMISTRY.jsp" />
<%
    }

```

```

%>
<%
    if (ID.equals("Insurance Information"))
    {
%>
    <jsp:forward page="DB_INSURANCEINFORMATION.jsp" />
<%
    }
%>

<%
    if (ID.equals("Patient History"))
    {
%>
    <jsp:forward page="DB_PTHISTORY.jsp" />
<%
    }

%>
<%

    if (ID.equals("Patient Information"))
    {
%>
    <jsp:forward page="DB_PTINFORMATION.jsp" />
<%
    }
%>

<%
    if (ID.equals("Routine Cultures"))
    {
%>
    <jsp:forward page="DB_ROUTINECULTURES.jsp" />
<%
    }
%>

<%
    if (ID.equals("Urinalysis"))
    {
%>

```

```

    <jsp:forward page="DB_URINALYSIS.jsp" />
<%
}
%>

<%
if (ID.equals("X-Ray Report"))
{
%>
    <jsp:forward page="DB_XRAYREPORT.jsp" />
<%
}
%>

<%
if (ID.equals("X-Ray Image"))
{
%>
    <jsp:forward page="DB_XRAYIMAGE.jsp" />
<%
}
%>

<%
if (ID.equals("MRI Report"))
{
%>
    <jsp:forward page="DB_MRIREPORT.jsp" />
<%
}
%>

<%
if (ID.equals("MRI Image"))
{
%>
    <jsp:forward page="DB_MRIIIMAGE.jsp" />
<%
}
%>

<%
if (ID.equals("CT Report"))
{

```



```
%>
    <jsp:forward page="DB_CTREPORT.jsp" />
<%
    }
%>

<%
    if (ID.equals("CT Image"))
    {
%>
        <jsp:forward page="DB_CTIMAGE.jsp" />
<%
    }
%>
</hr>
</body>
</html>
```

```
<%@ page session = "false" %>
<html>
<head><title> The Password Page </title> </head>
<body bgcolor = "floralwhite">
```

```
<h1 "align = "center"> <font size = "4"> <b> Please enter username and
password below </b> </font> </h1>
```

```
<form action = "test.jsp" method = "get">
```

```
Username <input type = "text" name = "Username" maxlength = "20">
<br> <br>
```

```
Password <input type = "password" name = "secret" maxlength = "20">
```

```
<br> <br>
```

```
<input type = "submit" name = "Submit" value = "Submit">
```

```
</form>
</body>
</html>
```

```
<html>
```

```
<head>
<title> Reject Page </title>
</head>

<body bgcolor ="floralwhite">

<%
out.println("Social Security Number was entered incorrectly. Please follow link
and try again");

%>
<br>
<br>
<br>
<br>
<center><a href="IntroPage.jsp"><font face="arial" size = "4">Return to Main
Page</font></a></center>

</body>
</html>
```

```
<html>
<head>
<title> Reject Page </title>
</head>

<body bgcolor ="floralwhite">

<%
out.println("The directory specified was Invalid. Please follow link and try
again");

%>
<br>
<br>
<br>
<br>
<center><a href="IntroPage.jsp"><font face="arial" size = "4">Return to Main
Page</font></a></center>

</body>
</html>
```

```

<html>
<head>
<title> Test page </title>
</head>
<%@ page import="java.sql.*"%>
<body>
<hr>

<%
int i;
    for (i=0; i<2; i++)
    {
String ID = request.getParameter("Username");
String PW = request.getParameter("secret");
%>

<%

        if ( (ID.equals("kliandy")) && (PW.equals("andyandy")) )
        {
%>
                <jsp:forward page = "IntroPage.jsp" />
<%
        }

%>

                <jsp:forward page = "Warning.jsp" />
<%
    }
%>

</hr>
</body>
</html>

<html>
<head>
<title> Warning Page </title>
</head>

```

```

<%@ page import="java.sql.*"%>
<body>

<%
out.println("Invalid Password Entered. Please see System Administrator for
login name and password");
%>

<br><br><br>

<center> <a href = "/Mproject/Password.jsp"> Login/Password </a></center>

</body>
</html>

<html>
<head>
<title> XRAY Image Page </title>
</head>
<%@ page import="java.sql.*"%>
<%@ page import="java.io.*"%>
<body bgcolor = "floralwhite">

<h1 align = "center"> <font size = "2", face = "Arial" color = "red"> <i> **Please
note that the most recent files are on the top while earlier pictures/files will be
further down the list. </i> </font>

</h1>

<%
String initial = request.getParameter("Initials");
//out.println(initial);
String directory = "d:\\temp\\Tomcat\\jakarta-tomcat-
4.0.3\\webapps\\ROOT\\Mproject\\XRAY-Images\\" + initial;

File dataDir = new File(directory);
if(!(dataDir.exists()) && !(dataDir.isDirectory()))
{
%>
    <jsp:forward page = "Reject2.jsp" />
<%
}

```

```

File[] listing = dataDir.listFiles();

for(int x = 0; x < listing.length; x++)
{
    out.println("<li>" + " <a href = "+ listing[x] +" " + "</a>" + "</li>");
    out.println(listing[x]);
}

%>
<br>
<br>
<br>
<center><a href="IntroPage.jsp"><font face="arial" size = "4">Return to Main
Page</font></a></center>

</hr>
</body>
</html>

<html>
<head>
<title> XRAY Report Page </title>
</head>
<%@ page import="java.sql.*"%>
<%@ page import="java.io.*"%>
<body bgcolor = "floralwhite">

<h1 align = "center"> <font size = "2", face = "Arial" color = "red"> <i> **Please
note that the most recent files are on the top while earlier pictures/files will be
further down the list. </i> </font>

</h1>

<%
String initial = request.getParameter("Initials");
//out.println(initial);
String directory = "d:\\temp\\Tomcat\\jakarta-tomcat-
4.0.3\\webapps\\ROOT\\Mproject\\XRAY-REPORTS\\" + initial;

File dataDir = new File(directory);
if(!(dataDir.exists()) && !(dataDir.isDirectory()))
{
%>

```

```

        <jsp:forward page = "Reject2.jsp" />
    <%
    }

    File[] listing = dataDir.listFiles();

    for(int x = 0; x < listing.length; x++)
    {

        out.println("<li>" + " <a href = "+ listing[x] + " " + "</a>" + "</li>");
        out.println(listing[x]);
    }

    %>
    <br>
    <br>
    <br>
    <center><a href="IntroPage.jsp"><font face="arial" size = "4">Return to Main
    Page</font></a></center>

    </hr>
    </body>
    </html>

    <html>
    <head>
    <title> CT Image Page </title>
    </head>
    <%@ page import="java.sql.*"%>
    <%@ page import="java.io.*"%>
    <body bgcolor = "floralwhite">

    <h1 align = "center"> <font size = "2", face = "Arial" color = "red"> <i> **Please
    note that the most recent files are on the top while earlier pictures/files will be
    further down the list. </i> </font>

    </h1>

    <%
    String SS = request.getParameter("Social Security");

    String initial = request.getParameter("Initials");

```

```
String directory = "d:\\temp\\Tomcat\\jakarta-tomcat-4.0.3\\webapps\\ROOT\\Mproject\\CT-Images\\" + initial;
```

```
File dataDir = new File(directory);  
if(!(dataDir.exists()) && !(dataDir.isDirectory()))
```

```
{  
    %>
```

```
        <jsp:forward page = "Reject2.jsp" />
```

```
    <%  
    }
```

```
    File[] listing = dataDir.listFiles();  
    %>
```

```
    <%
```

```
    for(int x = 0; x < listing.length; x++)
```

```
    {
```

```
        out.println("<li>" + " <a href = \"" + listing[x] + "\" + "</a>" + "</li>");  
        out.println(listing[x]);
```

```
    }
```

```
    %>
```

```
    <br>
```

```
    <br>
```

```
    <br>
```

```
    <center><a href="IntroPage.jsp"><font face="arial" size = "4">Return to Main  
    Page</font></a></center>
```

```
    </hr>
```

```
    </body>
```

```
    </html>
```

```
    <html>
```

```
    <head>
```

```
    <title> CT Image Page </title>
```

```
    </head>
```

```
    <%@ page import="java.sql.*"%>
```

```
    <%@ page import="java.io.*"%>
```

```
    <body bgcolor = "floralwhite">
```

```
    <h1 align = "center"> <font size = "2", face = "Arial" color = "red"> <i> **Please  
    note that the most recent files are on the top while earlier pictures/files will be  
    further down the list. </i> </font>
```

</h1>

<%

String initial = request.getParameter("Initials");

//out.println(initial);

String directory = "d:\\temp\\Tomcat\\jakarta-tomcat-4.0.3\\webapps\\ROOT\\Mproject\\CT-Reports\\" + initial;

File dataDir = new File(directory);

if(!(dataDir.exists()) && !(dataDir.isDirectory()))

{

%>

<jsp:forward page = "Reject2.jsp" />

<%

}

File[] listing = dataDir.listFiles();

for(int x = 0; x < listing.length; x++)

{

out.println("" + " <a href = "+ listing[x] + " " + "" + "");

out.println(listing[x]);

}

%>

<center>Return to Main Page</center>

</hr>

</body>

</html>

<html>

<head>

<title> MRI Image Page </title>

</head>

<%@ page import="java.sql.*"%>


```

<%@ page import="java.io.*"%>
<body bgcolor = "floralwhite">

<h1 align = "center"> <font size = "2", face = "Arial" color = "red"> <i> **Please
note that the most recent files are on the top while earlier pictures/files will be
further down the list. </i> </font>

</h1>

<%
String initial = request.getParameter("Initials");
//out.println(initial);
String directory = "d:\\temp\\Tomcat\\jakarta-tomcat-
4.0.3\\webapps\\ROOT\\Mproject\\MRI-Images\\" + initial;

File dataDir = new File(directory);
if(!(dataDir.exists()) && !(dataDir.isDirectory()))
{
%>
    <jsp:forward page = "Reject2.jsp" />
<%
}

File[] listing = dataDir.listFiles();

for(int x = 0; x < listing.length; x++)
{
    out.println("<li>" + " <a href = "+ listing[x] + " " + "</a>" + "</li>");
    out.println(listing[x]);
}

%>
<br>
<br>
<br>
<center><a href="IntroPage.jsp"><font face="arial" size = "4">Return to Main
Page</font></a></center>

</hr>
</body>
</html>

```

```

<html>
<head>
<title> MRI Image Page </title>
</head>
<%@ page import="java.sql.*"%>
<%@ page import="java.io.*"%>
<body bgcolor ="floralwhite">

<h1 align = "center"> <font size = "2", face = "Arial" color = "red"> <i> **Please
note that the most recent files are on the top while earlier pictures/files will be
further down the list. </i> </font>

</h1>

<%
String initial = request.getParameter("Initials");
//out.println(initial);
String directory = "d:\\temp\\Tomcat\\jakarta-tomcat-
4.0.3\\webapps\\ROOT\\Mproject\\MRI-REPORTS\\" + initial;

File dataDir = new File(directory);
if(!(dataDir.exists()) && !(dataDir.isDirectory()))
{
%>
    <jsp:forward page = "Reject2.jsp" />
<%
}

File[] listing = dataDir.listFiles();

for(int x = 0; x < listing.length; x++)
{
    out.println("<li>" + " <a href = "+ listing[x] +" " + "</a>" + "</li>");
    out.println(listing[x]);
}

%>
<br>
<br>
<br>
<center><a href="IntroPage.jsp"><font face="arial" size = "4">Return to Main
Page</font></a></center>

```

```
</hr>
</body>
</html>
```

```
<html>
<head>
<title>DataBase Access</title>
</head>
<%@ page import="java.sql.*"%>
<body bgcolor="navajowhite">
```

```
<h2>Laboratory Results</h2>
<table border="1" cellpadding="20">
<caption> Arterial Blood Gas Results </caption>
<tr><th>Test
DATE<th>Ph<th>CO2<th>P02<th>HCO3<th>BE<th>FIO2<th>HR<th>RR<th
>SSN</tr>
```

```
<%
String SS = request.getParameter("Social Security");
//out.println(SS);
```

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection con =
DriverManager.getConnection("jdbc:odbc:Andy","system","pass");
```

```
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery("SELECT * FROM Li.ABG ORDER BY
TESTDATE");
```

```
while (rs.next())
{
```

```
String date = rs.getString("TESTDATE");
String PH = rs.getString("PH");
String CO2 = rs.getString("CO2");
String PO2 = rs.getString("PO2");
String HC03 = rs.getString("HC03");
String BE = rs.getString("BE");
String FIO2 = rs.getString("FIO2");
```

```
String HR = rs.getString("HR");
String RR = rs.getString("RR");
String SSN = rs.getString("LSSN");
```

```
if(SSN.equals(SS))
{
```

```
    out.print("<tr>");
    out.print("<td>" + date + "</td>");
    out.print("<td>" + PH + "</td>");
    out.print("<td>" + CO2 + "</td>");
    out.print("<td>" + PO2 + "</td>");
    out.print("<td>" + HC03 + "</td>");
    out.print("<td>" + BE + "</td>");
    out.print("<td>" + FIO2 + "</td>");
    out.print("<td>" + HR + "</td>");
    out.print("<td>" + RR + "</td>");
    out.print("<td>" + SSN + "</td>");
    out.print("</tr>");
```

```
}
```

```
}
```

```
con.close();
```

```
%>
```

```
<br>
```

```
<center><a href="IntroPage.jsp"><font face="arial" size = "4">Return to Main  
Page</font></a></center>
```

```
</body>
```

```
</html>
```

```
<html>
```

```
<head>
```

```
<title>DataBase Access</title>
```

```
</head>
```

```
<%@ page import="java.sql.*"%>
```

```
<body bgcolor="navajowhite">
```

```
<h2>Laboratory Results</h2>
```

```
<table border="1" cellpadding="20">
```

```

<caption> Cell Blood Count Results </caption>
<tr><th>Test
DATE<th>WBC<th>RBC<th>HGB<th>HCT<th>MCV<th>MCH<th>MCHC<th>
MPV<th>RPW<th>ANC<th>MONO<th>EOSIN<th>MYELO<th>LYMPH<th>
SEGS<th>BANDS<th>SSN</tr>

```

```

<%

```

```

String SS = request.getParameter("Social Security");

```

```

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

```

```

Connection con =

```

```

DriverManager.getConnection("jdbc:odbc:Andy","system","pass");

```

```

Statement stmt = con.createStatement();

```

```

ResultSet rs = stmt.executeQuery("SELECT * FROM Li.CBC ORDER BY
TESTDATE");

```

```

while (rs.next())

```

```

{

```

```

    String date = rs.getString("TESTDATE");

```

```

    String WBC = rs.getString("WBC");

```

```

    String RBC = rs.getString("RBC");

```

```

    String HGB = rs.getString("HGB");

```

```

    String HCT = rs.getString("HCT");

```

```

    String MCV = rs.getString("MCV");

```

```

    String MCH = rs.getString("MCH");

```

```

    String MCHC = rs.getString("MCHC");

```

```

    String MPV = rs.getString("MPV");

```

```

    String RPW = rs.getString("RPW");

```

```

    String ANC = rs.getString("ANC");

```

```

    String MONO = rs.getString("MONO");

```

```

    String EOSIN = rs.getString("EOSIN");

```

```

    String MYELO = rs.getString("MYELO");

```

```

    String LYMPH = rs.getString("LYMPH");

```

```

    String SEGS = rs.getString("SEGS");

```

```

    String BANDS = rs.getString("BANDS");

```

```

    String SSN = rs.getString("LSSN");

```

```

    if(SSN.equals(SS))

```

```

    {

```

```

        out.print("<tr>");

```

```

        out.print("<td>" + date + "</td>");

```

```

        out.print("<td>" + WBC + "</td>");

```

```

        out.print("<td>" + RBC + "</td>");

```

```

        out.print("<td>" + HGB + "</td>");
        out.print("<td>" + HCT + "</td>");
        out.print("<td>" + MCV + "</td>");
        out.print("<td>" + MCH + "</td>");
        out.print("<td>" + MCHC + "</td>");
        out.print("<td>" + MPV + "</td>");
        out.print("<td>" + RPW + "</td>");
        out.print("<td>" + ANC + "</td>");
        out.print("<td>" + MONO + "</td>");
        out.print("<td>" + EOSIN + "</td>");
        out.print("<td>" + MYELO + "</td>");
        out.print("<td>" + LYMPH + "</td>");
        out.print("<td>" + SEGS + "</td>");
        out.print("<td>" + BANDS + "</td>");
        out.print("<td>" + SSN + "</td>");
        out.print("</tr>");
    }
}

con.close();

%>
<br>
<center><a href="IntroPage.jsp"><font face="arial" size = "4">Return to Main
Page</font></a></center>
</body>
</html>

<html>
<head>
<title> CT Image Page </title>
</head>
<%@ page import="java.sql.*"%>
<%@ page import="java.io.*"%>
<body bgcolor = "floralwhite">

<h1>Welcome to the CT Image page </h1>
<hr>

<h2 align = "center"> <font size = "2", face = "Arial" color = "blue"> <i> Please
enter the initials of the patient first name first. For example if patient name is

```

John Smith then enter JS. Also please enter the last 4 digits of the patient's social security number. So JS0000 would be an example. </i>

</h2>

<form action = "CTIMAGEPROC.jsp" method = "get">

Patient Initials and last 4 digits of Social Security number <input type = "text" name = "Initials" maxlength = "20">

<input type = "submit" name = "submit" value = "Submit Now">

</hr>

</body>

</html>

<html>

<head>

<title> CT Report Page </title>

</head>

<%@ page import="java.sql.*"%>

<%@ page import="java.io.*"%>

<body bgcolor = "floralwhite">

<h1>Welcome to the CT Report page </h1>

<hr>

<h2 align = "center"> <i> Please enter the initials of the patient first name first. For example if patient name is John Smith then enter JS. Also please enter the last 4 digits of the patient's social security number. So JS0000 would be an example. </i>

</h2>

<form action = "CTREPORTPROC.jsp" method = "get">

Patient Initials and last 4 digits of Social Security number <input type = "text" name = "Initials" maxlength = "20">


```

<input type = "submit" name = "submit" value = "Submit Now">

<%
String SS = request.getParameter("Social Security");

%>

</hr>
</body>
</html>

<html>
<head>
<title>DataBase Access</title>
</head>
<%@ page import="java.sql.*"%>
<body bgcolor="navajowhite">

<h2>Laboratory Results</h2>
<table border="1" cellpadding="20">
<caption> General Chemistry Results </caption>
<tr><th>Test
DATE<th>NA<th>CO2<th>BUN<th>K<th>CREATININE<th>CL<th>CA<th>P
HOS<th>TOTAL_PROTEIN<th>ALBUMIN_PROTEIN<th>MG<th>LP<th>TRI
G<th>ALT<th>CHOL<th>AST<th>GLUCOSE<th>SSN</tr>

<%
String SS = request.getParameter("Social Security");

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection con =
DriverManager.getConnection("jdbc:odbc:Andy","system","pass");

Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery("SELECT * FROM
Li.GENERAL_CHEMISTRY ORDER BY TESTDATE");

while (rs.next())
{
    String date = rs.getString("TESTDATE");
    String NA = rs.getString("NA");
    String CO2 = rs.getString("CO2");
    String BUN = rs.getString("BUN");
    String K = rs.getString("K");

```



```

String CREATININE = rs.getString("CREATININE");
String CL = rs.getString("CL");
String CA = rs.getString("CA");
String PHOS = rs.getString("PHOS");
String TOTAL_PROTEIN = rs.getString("TOTAL_PROTEIN");
String ALBUMIN_PROTEIN = rs.getString("ALBUMIN_PROTEIN");
String MG = rs.getString("MG");
String LP = rs.getString("LP");
String TRIG = rs.getString("TRIG");
String ALT = rs.getString("ALT");
String CHOL = rs.getString("CHOL");
String AST = rs.getString("AST");
String GLUCOSE = rs.getString("GLUCOSE");
String SSN = rs.getString("LSSN");

```

```

if(SSN.equals(SS))

```

```

{
    out.print("<tr>");
    out.print("<td>" + date + "</td>");
    out.print("<td>" + NA + "</td>");
    out.print("<td>" + CO2 + "</td>");
    out.print("<td>" + BUN + "</td>");
    out.print("<td>" + K + "</td>");
    out.print("<td>" + CREATININE + "</td>");
    out.print("<td>" + CL + "</td>");
    out.print("<td>" + CA + "</td>");
    out.print("<td>" + PHOS + "</td>");
    out.print("<td>" + TOTAL_PROTEIN + "</td>");
    out.print("<td>" + ALBUMIN_PROTEIN + "</td>");
    out.print("<td>" + MG + "</td>");
    out.print("<td>" + LP + "</td>");
    out.print("<td>" + TRIG + "</td>");
    out.print("<td>" + ALT + "</td>");
    out.print("<td>" + CHOL + "</td>");
    out.print("<td>" + AST + "</td>");
    out.print("<td>" + GLUCOSE + "</td>");
    out.print("<td>" + SSN + "</td>");
    out.print("</tr>");
}

```

```

}

```

```

con.close();

%>
<br>
<center><a href="IntroPage.jsp"><font face="arial" size = "4">Return to Main
Page</font></a></center>
</body>
</html>

<html>
<head>
<title>DataBase Access</title>
</head>
<%@ page import="java.sql.*"%>
<body bgcolor="navajowhite">

<h2>Patient Data</h2>
<table border="1" cellpadding="20">
<caption> Insurance Information </caption>
<tr><th>SSN<th>INSURED<th>NAME_OF_INSURANCE<th>EXPIRED_DATE<th>POLICY_NUMBER<th>INSURANCE_PHONE_NUMBER<th>INSURANCE_STREET_NUMBER<th>INSURANCE_STREET_NAME<th>INSURANCE_CITY<th>INSURANCE_ZIPCODE<th>DEDUCTIBLE<th>MAXIMUM_COVERAGE<th>ACCOUNT_BALANCE<th>EXAM_DATE<th>EXAM_LOCATION<th>EXAM_DOCTOR<th>BILL<th>EXAM_NAME<th>EXAM_TIME<th>PAY_METHOD</tr>

<%
String SS = request.getParameter("Social Security");

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection con =
DriverManager.getConnection("jdbc:odbc:Andy","system","pass");

Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery("SELECT * FROM
Li.INSURANCE_INFORMATION");

while (rs.next())
{
    //String IISSN = rs.getString("IISSN");
    String SSN = rs.getString("SSN");
    String INSURED = rs.getString("INSURED");

```

```

String NAME_OF_INSURANCE =
rs.getString("NAME_OF_INSURANCE");
String EXPIRED_DATE = rs.getString("EXPIRED_DATE");
String POLICY_NUMBER = rs.getString("POLICY_NUMBER");
String INSURANCE_PHONE_NUMBER =
rs.getString("INSURANCE_PHONE_NUMBER");
String INSURANCE_STREET_NUMBER =
rs.getString("INSURANCE_STREET_NUMBER");
String INSURANCE_STREET_NAME =
rs.getString("INSURANCE_STREET_NAME");
String INSURANCE_CITY = rs.getString("INSURANCE_CITY");
String INSURANCE_ZIPCODE =
rs.getString("INSURANCE_ZIPCODE");
String DEDUCTIBLE = rs.getString("DEDUCTIBLE");
String MAXIMUM_COVERAGE =
rs.getString("MAXIMUM_COVERAGE");
String ACCOUNT_BALANCE = rs.getString("ACCOUNT_BALANCE");
String EXAM_DATE = rs.getString("EXAM_DATE");
String EXAM_LOCATION = rs.getString("EXAM_LOCATION");
String EXAM_DOCTOR = rs.getString("EXAM_DOCTOR");
String BILL_AMOUNT = rs.getString("BILL");
String EXAM_NAME = rs.getString("EXAM_NAME");
String EXAM_TIME = rs.getString("EXAM_TIME");
String PAY_METHOD = rs.getString("PAY_METHOD");

if(SSN.equals(SS))
{
    out.print("<tr>");
    //out.print("<td>" + IISN + "</td>");
    out.print("<td>" + SSN + "</td>");
    out.print("<td>" + INSURED + "</td>");
    out.print("<td>" + NAME_OF_INSURANCE + "</td>");
    out.print("<td>" + EXPIRED_DATE + "</td>");
    out.print("<td>" + POLICY_NUMBER + "</td>");
    out.print("<td>" + INSURANCE_PHONE_NUMBER + "</td>");
    out.print("<td>" + INSURANCE_STREET_NUMBER + "</td>");
    out.print("<td>" + INSURANCE_STREET_NAME + "</td>");
    out.print("<td>" + INSURANCE_CITY + "</td>");
    out.print("<td>" + INSURANCE_ZIPCODE + "</td>");
    out.print("<td>" + DEDUCTIBLE + "</td>");
    out.print("<td>" + MAXIMUM_COVERAGE + "</td>");
    out.print("<td>" + ACCOUNT_BALANCE + "</td>");
    out.print("<td>" + EXAM_DATE + "</td>");
    out.print("<td>" + EXAM_LOCATION + "</td>");
}

```

```

        out.print("<td>" + EXAM_DOCTOR + "</td>");
        out.print("<td>" + BILL_AMOUNT + "</td>");
        out.print("<td>" + EXAM_NAME + "</td>");
        out.print("<td>" + EXAM_TIME + "</td>");
        out.print("<td>" + PAY_METHOD + "</td>");
        out.print("</tr>");
    }
}

con.close();

%>
<br>
<center><a href="IntroPage.jsp"><font face="arial" size = "4">Return to Main
Page</font></a></center>

</body>
</html>

<html>
<head>
<title>DataBase Access</title>
</head>
<%@ page import="java.sql.*"%>
<body>

<table border="1" cellpadding="20">
<tr><th>NAME<th>SSN</tr>

<%
String SS = request.getParameter("Social Security");

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection con =
DriverManager.getConnection("jdbc:odbc:Andy","system","pass");

Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery("SELECT * FROM Li.LAB_REPORT");

while (rs.next())

```

```

{
    String NAME = rs.getString("NAME");
    //String LSSN = rs.getString("LSSN");
    String SSN = rs.getString("SSN");

    if(SSN.equals(SS))
    {
        out.print("<tr>");
        out.print("<td>" + NAME + "</td>");
        //out.print("<td>" + LSSN + "</td>");
        out.print("<td>" + SSN + "</td>");
        out.print("</tr>");
    }
}

```

```

con.close();

```

```

%>

```

```

<br>

```

```

<center><a href="IntroPage.jsp"><font face="arial" size = "4">Return to Main
Page</font></a></center>

```

```

</body>

```

```

</html>

```

```

<html>

```

```

<head>

```

```

<title> MRI Image Page </title>

```

```

</head>

```

```

<%@ page import="java.sql.*"%>

```

```

<%@ page import="java.io.*"%>

```

```

<body bgcolor = "floralwhite">

```

```

<h1>Welcome to the MRI Image page </h1>

```

```

<hr>

```

```

<h2 align = "center"> <font size = "2", face = "Arial" color = "blue"> <i> Please
enter the initials of the patient first name first. For example if patient name is
John Smith then enter JS. Also please enter the last 4 digits of the patient's
social security number. So JS0000 would be an example. </i> </font>

```

</h2>

<form action = "MRIIMAGEPROC.jsp" method = "get">

Patient Initials and last 4 digits of Social Security number <input type = "text"
name = "Initials" maxlength = "20">

<input type = "submit" name = "submit" value = "Submit Now">

<%

String SS = request.getParameter("Social Security");

%>

</hr>

</body>

</html>

<html>

<head>

<title> MRI Report Page </title>

</head>

<%@ page import="java.sql.*"%>

<%@ page import="java.io.*"%>

<body bgcolor = "floralwhite">

<h1>Welcome to the MRI Report page </h1>

<hr>

<h2 align = "center"> <i> Please
enter the initials of the patient first name first. For example if patient name is
John Smith then enter JS. Also please enter the last 4 digits of the patient's
social security number. So JS0000 would be an example. </i>

</h2>

<form action = "MRIREPORTPROC.jsp" method = "get">

Patient Initials and last 4 digits of Social Security number <input type = "text"
name = "Initials" maxlength = "20">

```

<br> <br>

<input type = "submit" name = "submit" value = "Submit Now">

<%
String SS = request.getParameter("Social Security");

%>

</hr>
</body>
</html>

<html>
<head>
<title>DataBase Access</title>
</head>
<%@ page import="java.sql.*"%>
<body bgcolor="navajowhite">

<h2>Patient Data</h2>
<table border="1" cellpadding="20">
<caption> Patient Information </caption>
<tr><th>Patient Name<th>SS Number</tr>

<%
String SS = request.getParameter("Social Security");

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection con =
DriverManager.getConnection("jdbc:odbc:Andy","system","pass");

Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery("SELECT * FROM LI.PATIENT");

while (rs.next())
{
    String name = rs.getString("NAME");
    String SSN = rs.getString("SSN");

    if(SSN.equals(SS))
    {
        out.print("<tr>");
        out.print("<td>" + name + "</td>");
    }
}
}

```

```

        out.print("<td>" + SSN + "</td>");
        out.print("</tr>");

    }
}

con.close();

%>
<br>
<center><a href="IntroPage.jsp"><font face="arial" size = "4">Return to Main
Page</font></a></center>
</body>
</html>

<html>
<head>
<title>DataBase Access</title>
</head>
<%@ page import="java.sql.*"%>
<body bgcolor="navajowhite">

<h2>Patient Data</h2>
<table border="1" cellpadding="20">
<caption> Patient History </caption>
<tr><th>NAME<th>SSN<th>PAST_MEDICAL_HISTORY</tr>

<%
String SS = request.getParameter("Social Security");

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection con =
DriverManager.getConnection("jdbc:odbc:Andy","system","pass");

Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery("SELECT * FROM Li.PT_HISTORY");

while (rs.next())
{
    String NAME = rs.getString("NAME");
    //String PSSN = rs.getString("PSSN");
    String SSN = rs.getString("SSN");

```



```
String PAST_MEDICAL_HISTORY =  
rs.getString("PAST_MEDICAL_HISTORY");
```

```
if(SSN.equals(SS))
```

```
{
```

```
    out.print("<tr>");
```

```
    out.print("<td>" + NAME + "</td>");
```

```
    //out.print("<td>" + PSSN + "</td>");
```

```
    out.print("<td>" + SSN + "</td>");
```

```
    out.print("<td>" + PAST_MEDICAL_HISTORY + "</td>");
```

```
    out.print("</tr>");
```

```
}
```

```
}
```

```
con.close();
```

```
%>
```

```
<br>
```

```
<center><a href="IntroPage.jsp"><font face="arial" size = "4">Return to Main  
Page</font></a></center>
```

```
</body>
```

```
</html>
```

```
<html>
```

```
<head>
```

```
<title>DataBase Access</title>
```

```
</head>
```

```
<%@ page import="java.sql.*"%>
```

```
<body bgcolor="navajowhite">
```

```
<h2>Patient Data</h2>
```

```
<table border="1" cellpadding="20">
```

```
<caption> Patient Information </caption>
```

```
<tr><th>NAME<th>SSN<th>PHONE_NUMBER<th>STREET_NUMBER<th>S
```

```
TREET_NAME<th>CITY<th>ZIPCODE<th>DRIVER_LICENSE<th>EMERGE
```

```
NCY_CALL_NO<th>PATIENT_COMPLAINT<th>PATIENT_ALLERGY</tr>
```

```
<%
```

```
String SS = request.getParameter("Social Security");
```

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

```

Connection con =
DriverManager.getConnection("jdbc:odbc:Andy","system","pass");

Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery("SELECT * FROM Li.PT_INFORMATION");

while (rs.next())
{
    String NAME = rs.getString("NAME");
    String SSN = rs.getString("SSN");
    //String PISSN = rs.getString("PISSN");
    String PHONE_NUMBER = rs.getString("PHONE_NUMBER");
    String STREET_NUMBER = rs.getString("STREET_NUMBER");
    String STREET_NAME = rs.getString("STREET_NAME");
    String CITY = rs.getString("CITY");
    String ZIPCODE = rs.getString("ZIPCODE");
    String DRIVER_LICENSE = rs.getString("DRIVER_LICENSE");
    String EMERGENCY_CALL_NO =
rs.getString("EMERGENCY_CALL_NO");
    String PATIENT_COMPLAINT =
rs.getString("PATIENT_COMPLAINT");
    String PATIENT_ALLERGY = rs.getString("PATIENT_ALLERGY");

    if(SSN.equals(SS))
    {
        out.print("<tr>");
        out.print("<td>" + NAME + "</td>");
        out.print("<td>" + SSN + "</td>");
        //out.print("<td>" + PISSN + "</td>");
        out.print("<td>" + PHONE_NUMBER + "</td>");
        out.print("<td>" + STREET_NUMBER + "</td>");
        out.print("<td>" + STREET_NAME + "</td>");
        out.print("<td>" + CITY + "</td>");
        out.print("<td>" + ZIPCODE + "</td>");
        out.print("<td>" + DRIVER_LICENSE + "</td>");
        out.print("<td>" + EMERGENCY_CALL_NO + "</td>");
        out.print("<td>" + PATIENT_COMPLAINT + "</td>");
        out.print("<td>" + PATIENT_ALLERGY + "</td>");
        out.print("</tr>");
    }
}
}

```

```

con.close();

%>
<br><center><a href="IntroPage.jsp"><font face="arial" size = "4">Return to
Main Page</font></a></center>
</body>
</html>

<html>
<head>
<title>DataBase Access</title>
</head>
<%@ page import="java.sql.*"%>
<body bgcolor = "navajowhite">

<h2>Laboratory Results</h2>
<table border="1" cellpadding="20">
<caption> Routine Cultures </caption>
<tr><th>TESTDATE<th>BLOOD_CULTURE<th>LOWER_RESP_CULTURE<
th>URINE_CULTURE<th>SSN</tr>

<%
String SS = request.getParameter("Social Security");

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection con =
DriverManager.getConnection("jdbc:odbc:Andy","system","pass");

Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery("SELECT * FROM
Li.ROUTINE_CULTURES ORDER BY TESTDATE");

while (rs.next())
{
    String TESTDATE = rs.getString("TESTDATE");
    String BLOOD_CULTURE = rs.getString("BLOOD_CULTURE");
    String LOWER_RESP_CULTURE =
rs.getString("LOWER_RESP_CULTURE");
    String URINE_CULTURE = rs.getString("URINE_CULTURE");
    String LSSN = rs.getString("LSSN");

    if(LSSN.equals(SS))
    {

```

```

        out.print("<tr>");
        out.print("<td>" + TESTDATE + "</td>");
        out.print("<td>" + BLOOD_CULTURE + "</td>");
        out.print("<td>" + LOWER_RESP_CULTURE + "</td>");
        out.print("<td>" + URINE_CULTURE + "</td>");
        out.print("<td>" + LSSN + "</td>");
        out.print("</tr>");
    }
}

con.close();

%>

<br>
<center><a href="IntroPage.jsp"><font face="arial" size = "4">Return to Main
Page</font></a></center>
</body>
</html>

<html>
<head>
<title>DataBase Access</title>
</head>
<%@ page import="java.sql.*"%>
<body bgcolor ="navajowhite">

<h2>Laboratory Results</h2>
<table border="1" cellpadding="20">
<caption> Urinalysis Results </caption>
<tr><th>Test
DATE<th>COLOR<th>CLARITY<th>SPEC_GRAV<th>PH<th>GLUCOSE<th>
LEUKOCYTES<th>KETONES<th>NITRITE<th>UROBIL<th>PROTEIN<th>
SQUAM<th>AMORPH<th>BACTERIA<th>BLOOD<th>BILI<th>SSN</tr>

<%
String SS = request.getParameter("Social Security");

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection con =
DriverManager.getConnection("jdbc:odbc:Andy","system","pass");

```

```
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery("SELECT * FROM Li.URINALYSIS ORDER
BY TESTDATE");
```

```
while (rs.next())
```

```
{
```

```
    String date = rs.getString("TESTDATE");
    String COLOR = rs.getString("COLOR");
    String CLARITY = rs.getString("CLARITY");
    String SPEC_GRAV = rs.getString("SPEC_GRAV");
    String PH = rs.getString("PH");
    String GLUCOSE = rs.getString("GLUCOSE");
    String LEUKOCYTES = rs.getString("LEUKOCYTES");
    String KETONES = rs.getString("KETONES");
    String NITRITE = rs.getString("NITRITE");
    String UROBIL = rs.getString("UROBIL");
    String PROTEIN = rs.getString("PROTEIN");
    String SQUAM = rs.getString("SQUAM");
    String AMORPH = rs.getString("AMORPH");
    String BACTERIA = rs.getString("BACTERIA");
    String BLOOD = rs.getString("BLOOD");
    String BILI = rs.getString("BILI");
    String SSN = rs.getString("LSSN");
```

```
    if(SSN.equals(SS))
```

```
{
```

```
        out.print("<tr>");
        out.print("<td>" + date + "</td>");
        out.print("<td>" + COLOR + "</td>");
        out.print("<td>" + CLARITY + "</td>");
        out.print("<td>" + SPEC_GRAV + "</td>");
        out.print("<td>" + PH + "</td>");
        out.print("<td>" + GLUCOSE + "</td>");
        out.print("<td>" + LEUKOCYTES + "</td>");
        out.print("<td>" + KETONES + "</td>");
        out.print("<td>" + NITRITE + "</td>");
        out.print("<td>" + UROBIL + "</td>");
        out.print("<td>" + PROTEIN + "</td>");
        out.print("<td>" + SQUAM + "</td>");
        out.print("<td>" + AMORPH + "</td>");
        out.print("<td>" + BACTERIA + "</td>");
        out.print("<td>" + BLOOD + "</td>");
        out.print("<td>" + BILI + "</td>");
```

```
out.print("<td>" + SSN + "</td>");
out.print("</tr>");
```

```
    }
}
```

```
con.close();
```

```
%>
```

```
<br>
```

```
<center><a href="IntroPage.jsp"><font face="arial" size = "4">Return to Main
Page</font></a></center>
```

```
</body>
```

```
</html>
```

```
<html>
```

```
<head>
```

```
<title> XRAY Image Page </title>
```

```
</head>
```

```
<%@ page import="java.sql.*"%>
```

```
<%@ page import="java.io.*"%>
```

```
<body bgcolor = "floralwhite">
```

```
<h1>Welcome to the XRAY Image page </h1>
```

```
<hr>
```

```
<h2 align = "center"> <font size = "2", face = "Arial" color = "blue"> <i> Please
enter the initials of the patient first name first. For example if patient name is
John Smith then enter JS. Also please enter the last 4 digits of the patient's
social security number. So JS0000 would be an example. </i> </font>
```

```
</h2>
```

```
<form action = "XRAYIMAGEPROC.jsp" method = "get">
```

```
Patient Initials and last 4 digits of Social Security number <input type = "text"
name = "Initials" maxlength = "20">
```

```
<br> <br>
```

```
<input type = "submit" name = "submit" value = "Submit Now">
```

```
<%
```

```
String SS = request.getParameter("Social Security");
```

```
%>
```

```
</hr>
```

```
</body>
```

```
</html>
```

```
<html>
```

```
<head>
```

```
<title> XRAY Report Page </title>
```

```
</head>
```

```
<%@ page import="java.sql.*"%>
```

```
<%@ page import="java.io.*"%>
```

```
<body bgcolor = "floralwhite">
```

```
<h1>Welcome to the XRAY Report page </h1>
```

```
<hr>
```

```
<h2 align = "center"> <font size = "2", face = "Arial" color = "blue"> <i> Please  
enter the initials of the patient first name first. For example if patient name is  
John Smith then enter JS. Also please enter the last 4 digits of the patient's  
social security number. So JS0000 would be an example. </i> </font>
```

```
</h2>
```

```
<form action = "XRAYREPORTPROC.jsp" method = "get">
```

```
Patient Initials and last 4 digits of Social Security number <input type = "text"  
name = "Initials" maxlength = "20">
```

```
<br> <br>
```

```
<input type = "submit" name = "submit" value = "Submit Now">
```

```
<%
```

```
String SS = request.getParameter("Social Security");
```

```
%>
```

```
</hr>
```

```
</body>
```

```
</html>
```

REFERENCES

- [1] P. Whitehead. JavaServer Pages. Hungry Minds, Inc. (NY): Marangraphics; 2001
- [2] Kochhar N, Gravina E, Priya N. Oracle University: Introduction to Oracle: SQL and PL/SQL Volume 1. Oracle; 1999
- [3] Kochhar N, Gravina E, Priya N. Oracle University: Introduction to Oracle, SQL and PL/SQL Volume 2. Oracle; 1999
- [4] J. Goodwill. Pure JSP: JavaServer Pages. SAMS Publishing, 2000
- [5] M. Thakkaar. Oracle 8i on Windows NT. SAMS Publishing, 1999
- [6] www.moreservlets.com/Using-Tomcat-4.html M. Etapi. "Configuring and Using Apache Tomcat 4" Mr. Marty Etapi is a employee of Sun Microsystems and writes tutorials for web sites as well.