A Web-Based Medical Image Archive System

Edward B. Suh[a], Steven Warach[b], Huey Cheung[a], Shaohua A Wang[a], Phanidral Tangiral[a], Marie Luby[b], and Robert L. Martino[a]

[a]Center for Information Technology, National Institutes of Health, Bethesda, MD 20892

[b]National Institute of Neurological Disorders and Stroke, National Institutes of Health, Bethesda, MD 20892

ABSTRACT

This paper presents a Web-based medical image archive system that has a three-tier, client-server architecture for the storage and retrieval of medical image data, as well as patient information and clinical data. The Web-based medical image archive system was designed to meet the need of the National Institute of Neurological Disorders and Stroke for a central image repository to address questions of stroke pathophysiology and imaging biomarkers in stroke clinical trials by analyzing images obtained from a large number of clinical trials conducted by government, academic and pharmaceutical industry researchers. In the database management-tier, we designed the image storage hierarchy to accommodate large binary image data files that the database software can access in parallel. In the middle-tier, a commercial Enterprise Java Bean server and secure Web server manages user access to the image database system. User-friendly Web-interfaces and applet tools are provided in the client-tier for easy access to the image archive system over the Internet. Benchmark test results show that our three-tier image archive system yields fast system response time for uploading, downloading, and querying the image database.

Keywords: Medical Image Archive, Web-based Image Archive, Image Database, Image Storage and Retrieval

1. INTRODUCTION

Biomedical research at the National Institutes of Health (NIH) involves the acquisition, storage, communication, display, and analysis of large numbers of medical images including many image modalities for both animal and human studies. NIH researchers use these medical images to discover biomarkers for disease states, determine the genetic origin of disease, evaluate the results of clinical trials, and develop therapeutic procedures. This accumulation of many medical images for a variety of research protocols and the desire to access and analyze them over many years generates the need for effective medical image archive systems, which allow researchers to readily access image and clinical data stored in a database via Web interfaces.

This paper presents a Web-based medical image archive system that we have developed for the National Institute of Neurological Disorders and Stroke (NINDS) at NIH. NINDS requires a central image repository to address questions of stroke pathophysiology and imaging biomarkers in stroke clinical trials by analyzing images obtained from a large number of clinical trials conducted by government, academic and pharmaceutical industry researchers. Our Web-based medical image archive system gives NINDS researchers and clinicians easy access to stroke MRI and CT images, as well as clinical trial data, for their research in developing experimental stroke therapies using imaging surrogate markers.

The Web-based medical image archive system employs a three-tier, client-server system architecture, incorporating modern database, secure Web, middleware software, network storage, and networking technologies. This three-tier system design gives users secure access to distributed system resources that appear as a single unified archive system from their desktop computer through user-friendly Web-interfaces over the Internet. Our design goal was to develop a scalable and upgradeable central image archive system, using commodity hardware and software, to store and disseminate medical image data through easy Web access over the Internet.

One of the essential features for the image archive system is quick delivery of image data in a consistent and reliable manner. We have chosen to implement the core of our Web-based medical image archive system using SUN Enterprise servers connected over 100 mega-bits per second Ethernet to ensure fast system response time and reliability. Benchmark results on our image archive system, consisting of a SUN 6500 database server, a SUN 450 application

server, and networked RAID storage systems, show consistently fast system response times as if uploads, queries, and downloads were performed at the user's local desktop computer.

2. SYSTEM ARCHITECTURE

Our three-tier, Web-based medical image archive system consists of a client-tier, middle-tier, and a database management-tier. This system is built on a distributed system component infrastructure, where system components are located over a wide geographic area. The system components are integrated with different interaction boundaries with the three level structures, appearing as a single archive system object to the user. A database server with a hierarchical storage system comprises the database management-tier. A secure Web server, Enterprise Java Bean (EJB) application server, and a firewall reside in the middle-tier as gateway objects, controlling access to the database management-tier. Desktop computers with a Web client in the client-tier give users transparent interfaces to data objects stored in the database management-tier.

2.1. Designing the System Architecture

We selected and combined the hardware and software components so that the system architecture provides the necessary image archive functions with the desired performance characteristics. As illustrated in Figure 1, we have designed a three-tier architecture to implement a flexible, scalable image archive system, using a commodity hardware and software products. This design includes a secure interface to support database query as well as image and textual data storage and retrieval functions. System security is essential for the protection of medical image, patient demographic, clinical and research data. The server-based, multi-tiered model provides a responsive, reliable and secure architecture for distributed transactional applications.

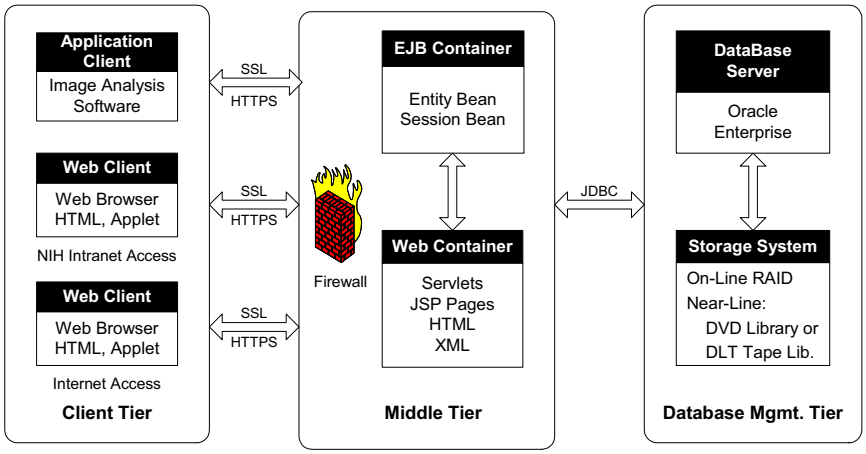

Figure 1. 3-Tier Web-Based Medical Image Archive System Architecture

The database management tier is implemented with a database server acting as a front-end to the online and near-line archive storage system where the image data is located. The archive storage is organized in a three-level hierarchy. We store the preprocessed meta-data and frequently accessed large data files in the first level of the hierarchy, which consist of high-performance disk storage units that provide rapid data access. The second level is implemented with RAID devices where recently obtained images are stored as a large object file. We store older images in near line storage units that form the final level of the hierarchy. The data on the high-performance disk storage units is partitioned and optimized for rapid parallel querying by multiple processors. The image database is also organized in a hierarchical structure and is indexed to facilitate the rapid searching for information.

The database hierarchy progresses from complete clinical studies to the patients within each study to the raw and processed image data. The database server interacts with textual data and preprocessed metadata abstractions to index the image database that is used for accessing image data files stored in the three levels of the storage hierarchy. The textual data and preprocessed metadata abstractions refer to descriptive information or attributes about the image data,

and are used in the form of keywords for the query and retrieval of the image data. Indexes are usually constructed manually by analyzing textual data and visual attributes of related images to find and organize important concepts for textual and visual queries. Other techniques such as content-based image retrieval (CBIR) could be used to offer the ability to retrieve images with similar etiology without first indexing the images, but they are typically based on simple features such as color, texture, shape or a combination of these.

The middle tier, the application server layer, consists of data rules to assist in resource sharing. Data rules are executed on the database server to manipulate the database. It manages the entire workflow process from data acquisition to data storage. It automates the loading, archiving, and subsequent display and delivery of medical images and metadata. Using a commercial EJB server as the middleware platform, clients can access the Web server through Web browsers using HTML, applets or dynamic HTML generated with JavaServer Pages (JSP).[1] Clients can also access the EJB server with stand-alone java language applications, such as image analysis programs. The web server and EJB server will then access the image database using Java Database Connectivity (JDBC).

The client tier provides a graphical user interface (GUI) and interactive software to grant the user access to the image archive. It presents users with patient data and related images for viewing, manipulation, and data entry. The two main types of user interface for this layer are the traditional standalone application and the Web-based application.

2.2. System Components

An Oracle 8i database management system is implemented on a Sun 6500 Enterprise server, which contains twenty-four 400MHz UltraSPARC processors, 24GB of memory, and 800 GB of fiber-channel disk array storage. The fiber channel disk storage uses the RAID 0+1 for high data transfer performance and reliability.

Web and EJB middleware services are implemented by running an IBM WebSphere application server on a Sun 450 Enterprise server to provide interfaces between the client-tier and the database management-tier. The SUN 450 server contains four 480MHz UltraSPARC processors, 4GB of memory, and 145.6 GB of disk storage.

Mapped to the database and application servers is the image storage system implemented on a Raidzone OpenNAS, Linux based RAID storage subsystem. This image storage system contains 1.03 TB disk storage and uses the common RAID 5 redundancy scheme configuration that protects against any single disk failure.

While the RAID storage system is very reliable, the regular backup of the database is essential to insure the integrity of the medical image and clinical data that is stored in the image archive system. For backup or near-line storage, a lower priced medium is used, but with the tradeoff of having a longer time delay for retrieval. A Sun StorEdge L1000 tape library robot with 36 GB/hour throughput was installed on the SUN 6500 Enterprise server as the backup system. This tape library robot contains two DLT7000 tape drives and holds twenty-five 35GB cartridges for a total of 875 GB of storage. We plan to install a DVD jukebox system as a near-line storage medium so that images that have not been accessed for a long time can be removed from the online RAID subsystem.

The system components described above are connected to a 100MB/s switched Ethernet network. This network connectivity allows a typical request for 50 Mbytes of image data to be transmitted in approximately four seconds or a larger request for 500 Mbytes of image data to be transmitted in approximately one minute, including overhead in both examples. We plan to upgrade the network connections to a Gigabit switched Ethernet, linking the archive system to the Internet.

2.3. System Security

To secure the image archive system from unauthorized access, the web server uses SSL web server certification with site authentication to insure the confirmation of the user's identity by validating the certificate and public ID that has been issued by NIH Secure Servers Certificate Authority (CA). We also implemented a strong 128-bit RC4 encryption algorithm, which encrypts patient information and user passwords, and a user login locking mechanism, which allows only three failed login attempts, thus providing a high degree of confidentiality for sensitive data and prevention of unauthorized system access.[2]

We implemented a two-layer access-authentication management scheme to give different users different access level privileges to the image archive database. This authentication management scheme controls all access and operations at the system and database levels, down to the individual data files stored in the archive database. To prevent unauthorized

identification of patient information associated with the image data, we developed a method that renders anonymous patient data and removes header information from the image data files.

A Lucent Brick firewall protects the archive system servers. This hardware-based firewall was implemented to prevent IP spoofing, allow packet filtering, and separate the network for the Web servers from the network for the database and storage system.

3. THE IMAGE DATABASE

We designed the image database system to store large binary image objects linked with clinical data and associated metadata. It supports queries across multiple imaging studies. An imaging study refers to a set of MRI and CT images collected for a single purpose on a single patient. Querying across collections of these studies allows for searching and retrieving multiple image data sets for common features. An example query would be to find the MRI images of all male patients within a certain range of ages who have been treated with a specific medical procedure.

3.1. Database Design

We divided the information stored in the image archive system into textual data and binary image data. The textual data, which is subject to indexing and querying, consists of metadata, patient data, and clinical data, such as examination technique, location of stroke, diagnosis, and treatment. The binary image data consists of MRI and CT images. The clinical data comprises over 200 attributes organized into ten groups as follows: Patient Information, Chief Complaints, Risk Factors, Studies Eligibility, Medication, Diagnosis, Stroke Location, Stroke Medical Complications, NIH Stroke Scale, and Thrombolytic Complications.

We primarily organized the textual data and the binary image data by patient. Each patient is assigned a unique subject ID number to avoid a situation where a patient is assigned multiple patient record numbers. This subject ID number is also used in database tables as a foreign key to ensure a deletion of all related data from the database when a patient record is deleted from the database. The patient and clinical data, including metadata, are stored on high-performance disk storage units attached to the database server to ensure rapid access. The image metadata includes the results derived from the original patient images, such as BWI B1000 Volume, Flair Volume, and ADC Mean of images, as well as the physical storage location of images in the external storage system. We partitioned the data according to its origin to speed up data access and improve parallel processing performance.

Figure 2 shows an entity-relationship (E-R) diagram capturing a subset of the full schema for this image archive system.

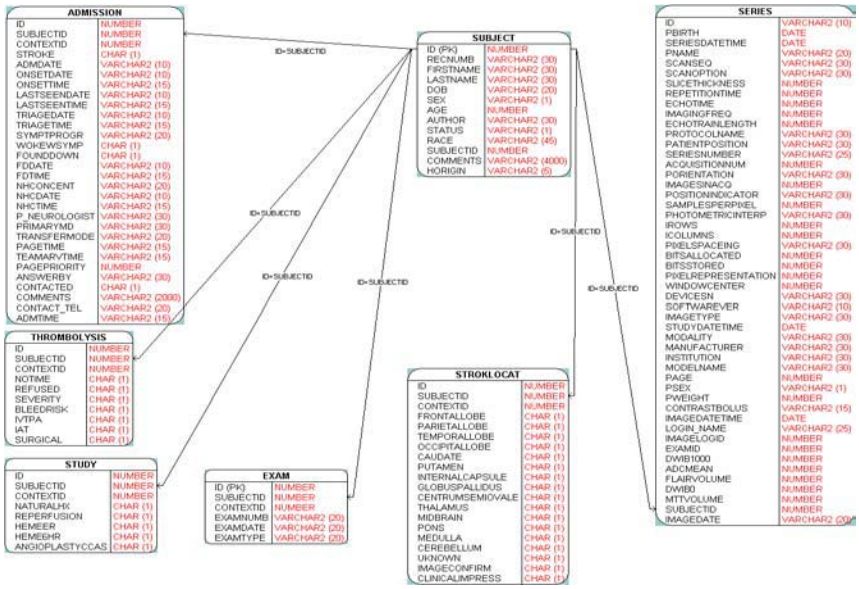

Figure 2. A Subset of Entity-Relationship Diagram of the Image Archive Database Schema

The binary images are stored in an external RAID storage system as BFILE data types. BFILE, a pointer to the operating system file directory system, is the data type in Oracle used to store large binary object files externally to the database. Since the images are stored outside the database, it is essential to enforce relational integrity between the image data and the textural data. We have implemented methods to enforce an automatic deletion of all image object files from the external storage system when a patient record is deleted.

3.2. Parallel Query Implementation

With the anticipation of rapid database growth, we have designed the database to use Oracle parallel database processing options. The Oracle parallel processing options allow the archive system to maintain query performance as the database grows. As the image database grows, two types of query parallelism can be exploited to speedup the query response time: first, inter-query parallelism and second, intra-query parallelism.

The inter-query parallelism is the ability to use multiple processors to execute several independent queries simultaneously. To reduce the startup time, multiple Oracle severs are spawned on multiple processors when the database is initiated. Each user is assigned an Oracle server when the user connects to the system. At any given time, an available Oracle sever can be connected when a user requests service. Since multiple processors service queries that are submitted simultaneously, the response time for each query is faster than it would be if a single processor executed query processing.

The intra-query parallelism is the ability to decompose a complex query statement composed of multiple sub-queries into these smaller sub-query units. Each sub-query unit can then be assigned to a unique server process for parallel query processing[3]. Since the overall query result of a complex query statement can be constructed by joining the results of each sub-query, the overall response time for the complex query would be about the same as the time required to complete the processing of a single sub-query unit. This assumes that the number of processors needed to complete each sub-query is available and does not include some minor overhead time. The overall response time for the complex query is much smaller using intra-query parallelism than if the complex query were processed by a single processor. By exploiting the intra-query parallelism, the overall response time for the complex query can be reduced significantly, the amount of the reduction depending on the number of processors used.

For Oracle, the SQL statement is given to a query coordinator, which is responsible for dividing the query among the processes and integrating the individual results into one result. The number of processes, also known as degree of parallelism; can be specified for each query using SQL hints. An example SQL statement that requests eight processes to run a query is as follows: SELECT /*+ FULL(all_images) PARALLEL(all_images, 8) */ sum(b1000_volum) FROM all_images.

3.3. Java Database Connectivity Implementation

Java Database Connectivity (JDBC) provides the capability of embedding dynamic SQL statements in Java programs to manipulate data stored in an Oracle database. Oracle Corporation provides four types of JDBC drivers: thin driver, OCI drivers, Sever-side thin driver, and Server-side internal driver. We implemented the JDBC thin driver in our Java client applications running on the application server to provide users transparent database connectivity through Web-interfaces. Oracle's thin JDBC driver is a one hundred percent Java compliant driver for client-side use and does not require an Oracle installation on the client-side. We use this thin JDBC driver for Web browser clients such as applets, servlets, EJBs, and Java Server Pages (JSP).

3.4. Future Development

To provide our users with the capability to search and retrieve images using content information (e.g., size, shape and color), we plan to develop and implement the capability of content-based visual information retrieval. Images can be stored in the database as the BLOB (Binary Large Object) data type, which can be used to perform analysis, data mining, and content-based visual information retrieval functions.[4] To allow the visual selection of images by a criteria that cannot be expressed in textual form[5], we also plan to implement the capability to thumbnail preview images stored in the archive database. When an image is loaded into the database, the system can generate a 40 by 40 pixel thumbnail of the image that can then be stored with the original image. Finally, as a sub-component to the image archive database, we plan to develop and implement a distributed database scheme that allows geographically distributed database systems

located at other research sites and containing additional research data to be connected to the central archive system over the Internet.[6]

4. WEB-BASED ARCHIVE SYSTEM

The secure Web server and the Enterprise Java Beans (EJB) application server in the middle-tier of the image archive system manage the entire workflow process from the user authentication to the data transaction. User authentication is handled by an encrypted password login to the Website via SSL. User passwords are encrypted using the RC4 algorithm and are stored in the archive database. The pre-authorized access level assigned to the user limits the operations a user can perform.

4.1. Enterprise Java Beans Application Server

The advantage of using EJB is that it separates the pre-built low-level details such as transaction handling, state management, resource pooling, etc. from the business logic, which make it easier to implement distributed transactional applications.[7,8] Our image archive system implements the EJB solution by providing only the business logic and using the container managed services.

The user interface is provided through a set of HTML and JSP pages for the storage, query, and retrieval of data in-and-out of the image archive system. The Web container hosts both presentation and business logic. When the user presents a request to access the image archive system, the JSP and/or servlet invoke the appropriate business logic, which will perform the service requested by the user.[9] The business logic is implemented as an EJB component hosted on the EJB container. The EJB components access the image archive database using JDBC to perform the requested service and returns results to the JSP and/or servlet, which will in turn present the results to the user.[10] The EJB server handles all the low-level details including database read and write operations. The server also employs high-performance, lightweight java applets for speedy image uploading and downloading.

4.2. Data Collection

We have implemented JavaServer Web pages, through which pre-authorized users can store and modify patient information, clinical data, and image files in the database by using a stateless Session Bean, NINDSDB, as shown in Figure 3. Typically, a session bean implements a business task and it invokes JDBC calls to execute the database reads and writes according to its business task.[11]

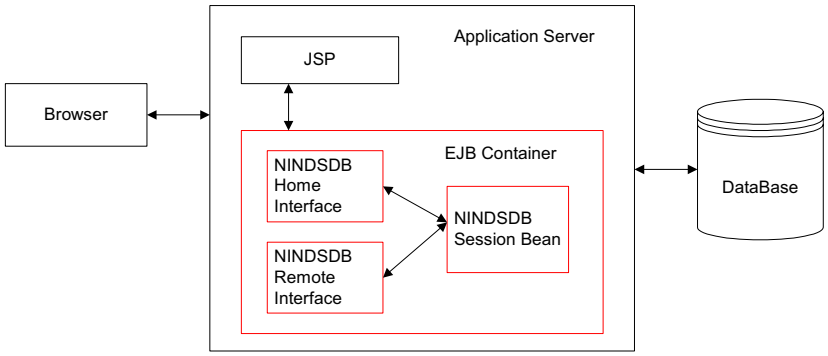

Figure 3. NINDSDB Stateless Session Bean

As shown in Figure 4, we have implemented an upload tool, a multi-threaded signed applet, to provide users the ability to display, navigate, and select image data files stored in the local computer so that the users can upload selected data files to the image archive system. As selected image data files are being uploaded, we display an uploading progress bar and a log window pop-up that shows uploading progress, status, and results. Uploaded image data files are stored in a temporary directory on the Web server to insure data integrity and perform authentication checks before they are inserted into the image database. We have implemented a Java stand-alone program, which is being executed as

CronJob, for the data integrity and authentication checks. After the data integrity and authentication checks, the Java program creates and inserts indexing and metadata in the image database and moves the image data files to the external image storage system.

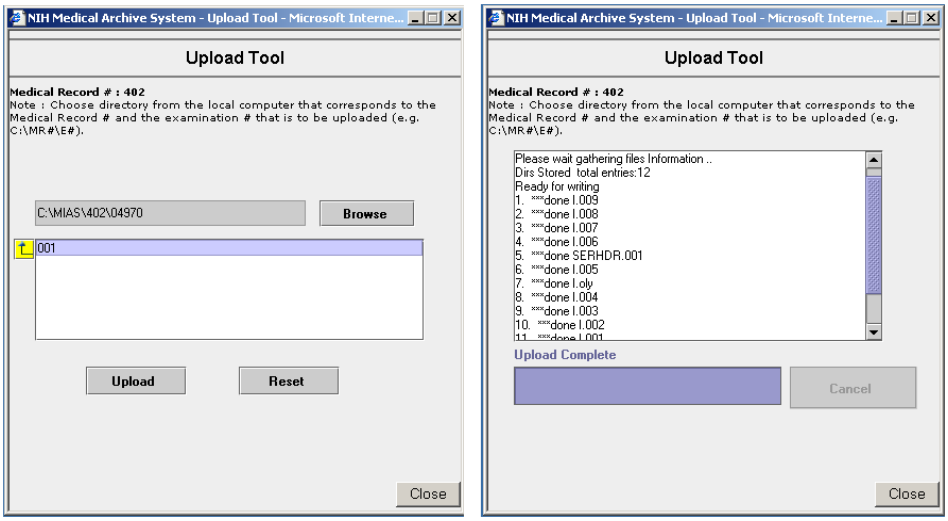

Figure 4. Image Upload Tool

4.3. Customized Database Query

Using provided Web interfaces, users can build a customized database query using over two hundred selectable query criteria including the following: select subject.id, recnumb, onsetdate, nihsstotal, nihscale.contextid from subject, mri, nihscale, study, medication, diagnosis, stroklocat, thromcomp where subject.id = admission.subjectid and subject.id = mri.subjectid and nihscale.subjected = subject.id and subject.id = risk.subjectid and subject.id = study.subjectid. The Web browser passes the customized database query to the NINDSDB stateless session bean in the EJB server, which then sends the query to the database server and obtains the query results for presentation to the user via the Web. All the queries are targeted to retrieve the medical images that meet the query criteria. A download tool, a multi-threaded signed applet, as shown in Figure 5 is provided for users to download image data files as well as patient information and clinical data. The download tool allows user to navigate the local computer hard drives to select a directory where the downloaded data can be saved.

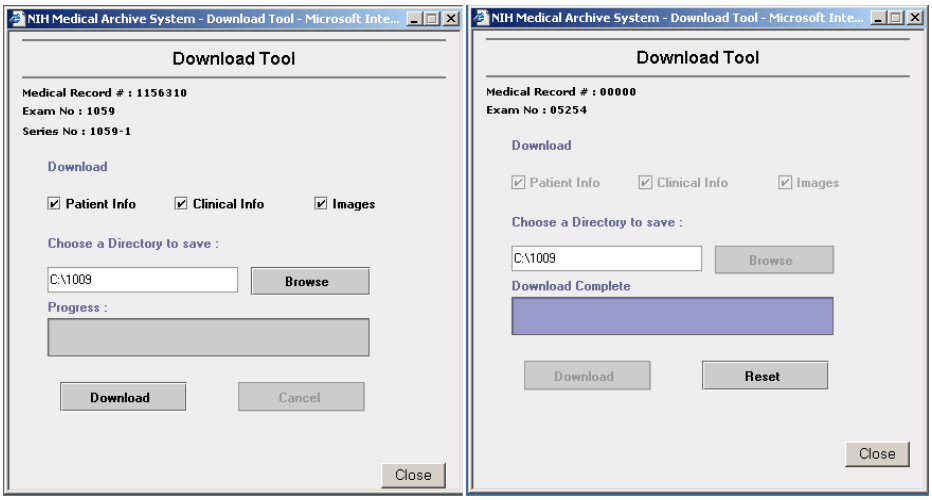

Figure 5. Image Download Tool

4.4. Archive System Administration

As shown in Figure 6, the user administration Web page gives the system administrator a user-friendly interface to add, delete, activate, and inactivate a user, as well as set the access privilege level. After a user requests a registration through a User Registration Web page, the system administrator can then activate the user and set an access privilege level for accessing the image archive system. The access privilege level is largely categorized as follows: first, different level of database access functionality (e.g., upload, query, and download); and second, limits on system visits (e.g., access counts and days).

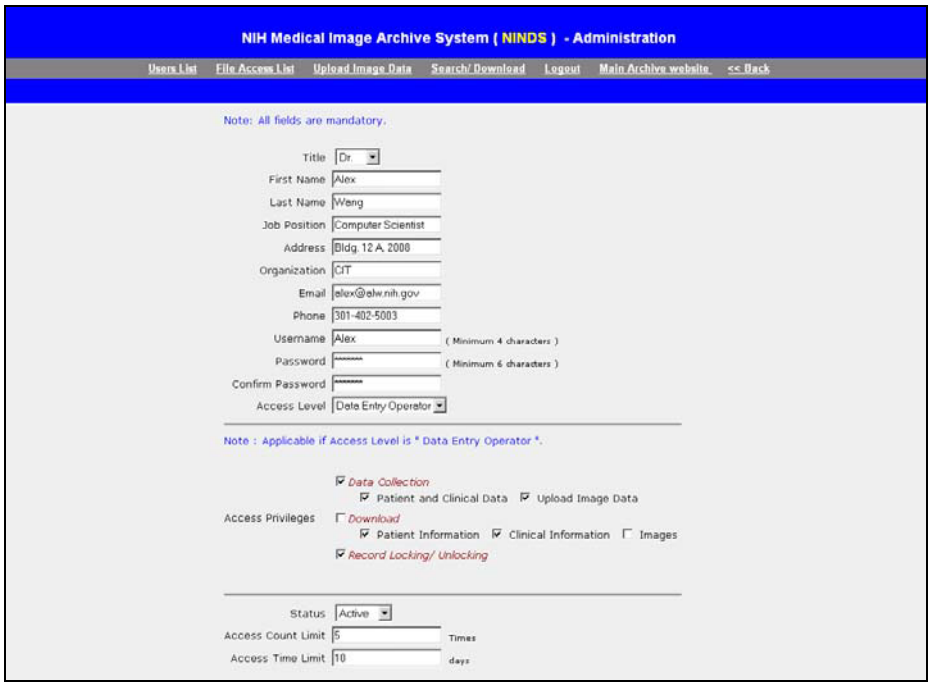

Figure 6. User Administration Page

4.5. Patient Information Anonymization

We developed a patient information anonymization program to prevent the identification of the patients associated with the images stored in the archive, and to blind researchers from unintentionally informing study participants or unconsciously biasing the evaluation of results. This program removes header information from the image data files and assigns random numbers for patient identification.

Using the GE Signa MR scanner that has the Signa 5 Genesis data format, NINDS scans Medical images into a directory file structure set by Patient Record Number, Examination Number, and Study Series Number. The anonymization program transforms this three-level directory structure into a single level directory. The randomization algorithm creates a single level, anonymous patient entry for this newly formed directory. Original image header information, such as patient ID, name, age, sex, history, and study description, are also removed from all the image data files.

The Processing Log pane, shown in Figure 7, logs all processing detail, such as creation and deletion of directories. Enabling the Reorder Directory button from the Task pull down menu can reverse the whole anonymization process. This will turn the single level directory structure back to the three levels directory structure. One can also interrupt the process by selecting the Stop radio button from the Task pull down menu. Once stopped, a dialog box will ask to see if the user wants to undo some or all of the work.

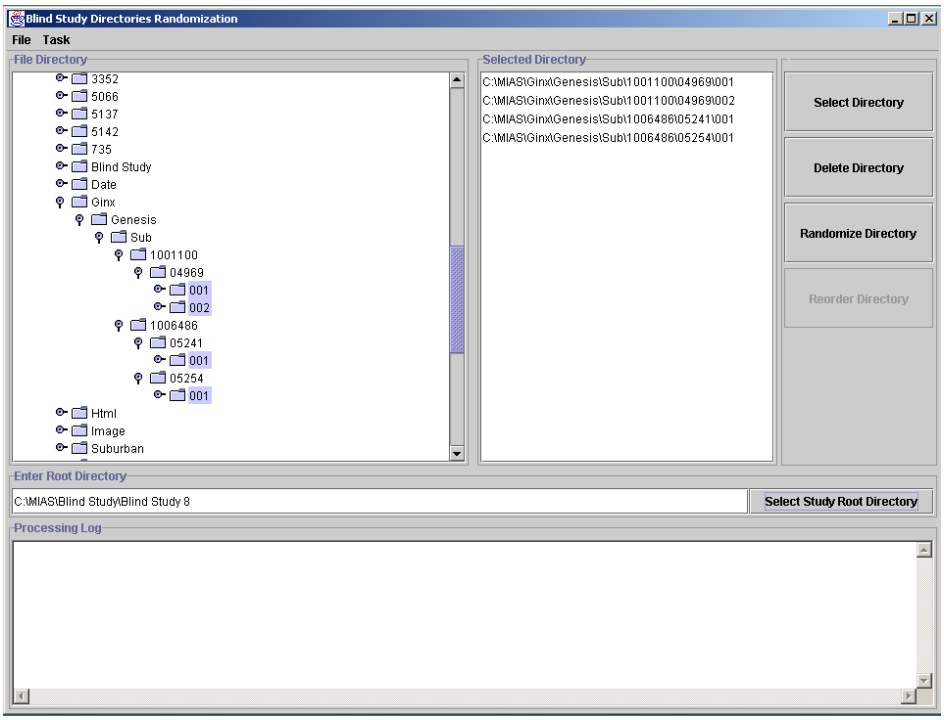

Figure 7. The Patient Information Anonymization Tool

5. PERFORMANCE BENCHMARK

We performed benchmark tests on two identical Sun 450 Enterprise servers to compare the performance of a servlet middleware implementation by Tomcat verses an EJB middleware implementation by the IBM WebSphere 3.5 application server.[12] The benchmark tests were performed during a regular business day with normal network traffic to measure the execution time for both the Tomcat and WebSphere implementations to perform database query, and image upload and image download.

In our benchmark tests, a simple query consists of about 20 data fields and a complex query consists of about 80 data fields. For each query type, 20 queries were submitted simultaneously. The test image database contains about 42,000 records and each search yielded about 1,000 records. Table 1 shows that WebSphere is faster than Tomcat and a simple query took much less time than a complex query.

	Query Execution Time	Record Reading Time	Measured Total Time	
WebSphere	27.567 sec	10.849 sec	39.854 sec	Simple Query
Tomcat	49.500 sec	54.197 sec	104.167 sec	
WebSphere	140.784 sec	4.216 sec	145.001 sec	Complex Query
Tomcat	207.804 sec	10.493 sec	218.560 sec	

Table 1. Query Comparison

For the upload and download benchmarks, four users submitted the upload and download requests simultaneously. We measured the upload time from reading the files from the HTTPInputStream to storing the files in a temporary directory on the web server. The download time was measured from reading the image files from the RAID image storage system to writing them to the HTTPOutputStream. Table 2 shows that the WebSphere significantly outperforms the Tomcat for image uploading. Table 3 shows that the Tomcat yields faster download times for a single file, whereas

the WebSphere yields faster download times for multiple files. When comparing the upload time with the download time of a single image file, the uploading is much faster than the downloading. This is mainly due to the time required to read the image file during a download operation from the RAID image storage system, which is made up of slower performance storage disks of 5000 rpm, verses the faster storage disks of 10,000 rpm used for uploading.

Image Upload	1.3MB (a single file)	11.3MB (a single file)
WebSphere	0.814 sec	4.985 sec
Tomcat	2.771 sec	19.039 sec

Table 2. Image Upload Time Comparison

Image Download	1.3MB (a single file)	11.3MB (a single file)
WebSphere	17.841 sec	103.266 sec
Tomcat	17.657 sec	78.985 sec
Image Download	**1.3MB (total of 43 files)**	**11.3MB (total of 101 files)**
WebSphere	11.455 sec	69.068 sec
Tomcat	15.666 sec	91.102 sec

Table 3. Image Download Time Comparison

6. CONCULSION

This paper presents a Web-based medical image archive system that has a three-tier, client-server architecture. We implemented this architecture by integrating distributed system components into a single image archive system for the storage and retrieval of medical image data files. We described the three-tier structure that has a database management-tier, a middle-tier, and a client-tier. This compartmentalized three-tier design allows us to accommodate an increase in the number of users and the amount required storage, upgrade the various components of the system as new commercial hardware and software products become available, and implement additional systems for different research environments. The database design accommodates large binary object files external to the database. The optimized parallel query capability ensures fast query and retrieval of image data files. Though the image database is designed to meet the specific NINDS stroke research needs, we feel that our design could easily be used for other image archive applications.

We described our implementation of user-friendly, configurable Web-interfaces, which provide users with easy access to the image archive system from their desktop computers over the Internet. The user administration Web page allows the system administrator the ability to control user level privileges for uploading data files, querying databases, and downloading data files remotely. We also developed applet tools and Java programs for image uploading and downloading, as well as stand alone Java programs for medical information anonymization. The benchmark test results demonstrate that our EJB middleware implementation using a commercial EJB application server, the IBM WebSphere, is efficient and robust.

The results presented in this paper form an initial report of our efforts. The detailed system design and benchmark test results will follow in a separate paper.

ACKNOWLEDGMENTS

The authors wish to thank Dr. Anthony Iano-Fletcher and Mr. Alexander Escobedo of the Division of Computational Bioscience, Center for Information Technology, NIH, for providing support for system administration and insightful suggestions on the configuration of the Web-based medical image archive system.

REFERENCES

1. Ken Ueno, Tom Alcott, Jeff Blight, Jeff Dekelver, Daniel Julin, Colin Pfannkuch, Tzueing Shieh, *WebSphere Scalability: WLM and Clustering using WebSphere Application Server Advanced Edition*, IBM Corp., 2000.
2. Scott Oaks, *Java Security, 2nd Edition*, O'Reilly & Associates, Inc., 2001.

3. Mahapatra, T. and Mishra, S. *Oracle Parallel Processing,* O'Reilly & Associates, Inc., 2000.
4. Zhu, X., Lee, K., Levin, D.L., Wong, S.T.C, Huang, H.K., Hoo, K.S., Gamsu, G. and Webb, W.R., "Temporal Image Database Design for Outcome Analysis of Lung Nodule," *Computerized Medical Imaging and Graphics,* 20(4): 347-356: 1996.
5. Stahl, J.N. and Kramann, B., "Customized Medical Image Databases: A Low-Cost Approach," *Computerized Medical Imaging and Graphics,* 21(6): 345-350: 1997.
6. Abdelguerfi, M. and Wong, K.F., *Parallel database Techniques,* IEEE Computer Society, 1998.
7. Bodoff S., Green D., Jendrock E., Pawolan M., Stearns B., *The J2EE Tutorial*, Sun Microsystems, Inc., 2001.
8. *The Java 2 Enterprise Edition Developer's Guide*, Sun Microsystems, Inc., 2000.
9. Marty Hall, *Core Servlets and JavaServer Pages*, Prentice Hall, 2000.
10. Richard Monson-Haefel, *Enterprise JavaBeans,* O'Reilly & Associates, Inc., 2000.
11. Chris Crenshaw, *Developer's Guide to Understanding Enterprise JavaBeans A Technical Primer to the Sun Microsystems™ EJB Specification,* Nova Labs.
12. *Introduction to WebSphere Application Server Version 3.5*, IBM Corp., June 2000.