Creating a Modern Electronic Medical Records (EMR) System
TJHSST Senior Research Project
Quarter 4
Computer Systems Lab 2009-2010

Jeremy Chaikind

June 10, 2010

Abstract

This project will attempt to create a functional, user-friendly medical management and medical records (EMR) system. Web-based programming languages, such as PHP, HTML, and CSS will be used with MySQL databases. Databases will be designed using the Relational Database Model and considering the ACID (*Atomicity*, *Consistency*, *Isolation*, and *Durability*) paradigm. The EMR design will ensure expandability, intuitive interface, and practicality for the end user.

Keywords: databases, HIPAA compliance, medical systems, electronic medical records (EMR), web applications

1 Introduction and Background

The business of medicine is a topic front and center for many Americans today. Beyond the question of health insurance reform, the United States government is in the process of changing the medical industry itself. Doctors have been given incentives to convert physical, paper charts to electronic ones in the near future. Soon after, physicians will be charged fees for using paper charts. [Butcher, 26] These changes present a difficult situation for doctors. Despite the exorbitant costs of many preexisting Electronic Medical Records (EMR) systems, some popular systems use older programming techniques and languages, and are as a result unintuitive to use. More problematic are the database designs used by many EMR systems, which do not allow physicians the flexibility they need to enter data with the precision they currently do for paper charts. This project plans to examine the feasibility of creating an EMR system designed in conjunction with physicians to ensure ease-of-use, using forward-thinking, more open, web-based languages, such as PHP, HTML, CSS, and MySQL and more modern database design paradigms to better approximate the convenience and power of using conventional paper charts.

2 Researcher Experience

Attempting a project of this scale is a difficult undertaking under any circumstances. The researcher's preexisting experience in programming and medical applications makes the task somewhat more reasonable. Prior to beginning this project, the researcher had a strong understanding of the PHP, HTML, and CSS, as well as basic experience with the MySQL and Javascript programming languages that will be used for this project. Within a few weeks, practical MySQL proficiency was cultivated through basic database work.

3 Development

3.1 Review of Literature

In order to meet medical security standards, the researcher examined HIPAA compliance for physicians and physicians's offices. Because this project primarily requires technological compliance with HIPAA regulations, an academic article specifically detailing security practices for HIPAA-compliant data transfer of EMR was studied. [Hristidis]

After an initial experimentation phase of this project was completed, the exact nature of security measures necessary for HIPAA-compliance was studied. An article about HIPAA-compliant digital storage measures was considered, recommending the implementation of meticulous documentation of actions and backup, as well as suggesting compliance with the National Institute of Standards and Technology (NIST) 800 series of documents, which detail general government guidelines for data storage and security. [Davis]

Further investigation in this area resulted in discussion with Russell McWey, M.D., a physician at the Virginia Hopspital Center in Arlington, VA who works closely with the hospital IT staff to manage its digital records. Based on a telephone interview with and subsequent typed letter from Dr. McWey, the researcher has determined that no encryption or additional system security is necesary for an intraoffice EMR system, negating the need for NIST compliance. However, an "audit trail" like that mentioned in the HIPAA-compliant storage article is necessary and will therefore be implemented in this project. [McWey]

The researcher also studied of modern practices for database management, including the ACID paradigm for database design and the Relational Database model. Initially, the topic was studied by informal work on design with another student of the Computer Systems Lab (Jason Koenig). Exposure to the ACID paradigm continued by studying an article specifically about database design and management [Haerder]. The Relational Database model was further studied in the context of an article about creating a general database system using both the Relational and Object-Oriented models to integrate media in an SQL database.

In keeping with the third quarter goal to further user interface development, an article about the creation of a robust, active-content-compatible WYSIWYG (What-You-See-Is-What-You-Get) was reviewed. Implementing the CRUD paradigm, this system offered the end user to **C**reate, **R**ead, **U**pdate, and **D**elete data, which are features of an active document. [Karger, 257] The system created in the article expands upon this idea by creating a WYSIWYG for an intermediate user to easily modify the types of data to which CRUD could be applied. [Karger, 258-259] While the ideas in this article have not directly influenced work done to date, the projected future of this project will include user-customizable templates for data entry. For this reason, work on templates this quarter was done in a way to ensure automated, function-based template design which may be implemented in a WYSIWYG for CRUD content creation such as this in the future.

3.2 Theory

To ensure the durability and utility of this EMR system, a server using Linux, Apache, MySQL, and PHP (LAMP) will be used. Unlike many other medical management systems that use older, closed Microsoft database technologies, this EMR will utilize the a more open database model so that the system will be applicable in the future.

The ACID paradigm will also be implemented for this system. Implementation of ACID, an abreviation for *Atomicity*, *Consistency*, *Isolation*, and *Durability*, ensures that information retrieved from a database is always correct.

Atomicity specifies that specific database functions must be performed in total or not at all. [Haerder, 289] For example, if a function calls for a database entry to be deleted in one table and added to another, neither database action will occur until both are requested. In this way, should the transaction be interrupted, the entry cannot be deleted in one place without being added to the other. Atomicity prevents database corruption that could provide incorrect information with disasterous results.

Consistency states that at all times actions called on the database (assuming Atomicity) leave the database in a correct state. [Haerder, 289-290] While a database that fails to practice Atomicity may crash, allowing the database to fall into an incorrect state, a database that fails to practice Consistency can leave the database in an incorrect state after functioning correctly. As a result, a correctly-functioning Consistent database will never write incorrect data to the database.

Isolation demands that all database processes run without knowledge of other functions running concurrently. [Haerder, 290] In a database without Isolation implemented, a user accessing one part of the database could see incorrect data from an intermediate step of an ongoing database process. For example, if one user accessed a patient record in order to call him/her while another user was in the process of changing the patient's phone number, the first user may see the old phone number, the new phone number, or no phone number (if the second user accessed the record while the database transaction was in progress). By implementing the principle of Isolation, no two users could ever access the same record, preventing this problem.

Durability ensures the integrity of all data by requiring database data to survive any malfunction. This could be accomplished with relative ease by instituting measures of redundancy and requiring multiple sources to match in order to display information. To prevent data loss on a larger scale, database backup is a necessity. If Durability exists, one can be certain that correctly-entered data will never become corrupted.

The Relational Database model is also important in a modern database design. The Relational model uses a single, global, unique record ID to associate various data to the same individual record. For example, an individual John Doe may have a global ID of 123. Data relating to John Doe will be organized into smaller, more focused tables. In the phone numbers table, there might be three records for global ID 123, representing John's home, cell, and office numbers. Two records for global ID 123 might be present in the addresses table for John's Manhattan penthouse and his country residence. In the medical field, the accuracy of data can literally determine life-and-death situations, so a physician should never be hampered by a strict program design for his/her EMR system. With this in mind, the Relational model will be implemented in this EMR system to ensure the physician has the most flexibility possible in data entry and organization.

4 Expected Procedure and Methodology

To program this EMR system, web-based languages, such as HTML, CSS, and Javascript (for the user interface) and PHP and MySQL (for database and other active-web functions) will be used for almost all aspects of the project. Initially, files will be located on a personal remote web server. However, the program will be transferred to a physical server as soon as possible in order to permit security testing to begin. To enhance the user experience, aspects of the AJAX (Asynchronous Javascript and XML) varient AJAJ (Asynchronous Javascript and JSON [Javascript Object Notation]) will be used to create such elements as database-based autocomplete form inputs. As a result, beginning to type certain common phrases (such as medication names) into a given template field will produce a list of options from which the end user may choose (this will be further explained in the Third Quarter Results section).

In order to test the EMR system, false data will initially be used for alpha testing by the researcher. This type of testing will be adequate for evaluating basic functionality of the program. For the program to be effectively tested for intuitive interface design, additional feature requests, and utility for large amounts of data, actual patient data must be used in the context of a physician's office. The researcher plans to test the system in the office of Pediatric Ophthalmologist Melissa Kern, M.D. at the Virginia Hospital Center complex in Arlington, VA. The EMR system functionality and user interface will be designed to best fit the needs of this office. Further testing and application may result in work with Barry Byer, M.D. at the Virginia Hospital Center, a

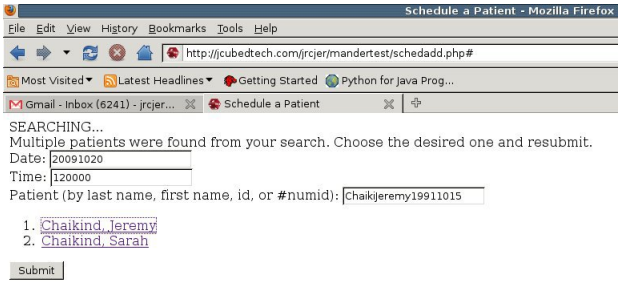

Figure 1: A screenshot of the first quarter Schedule Patient screen.

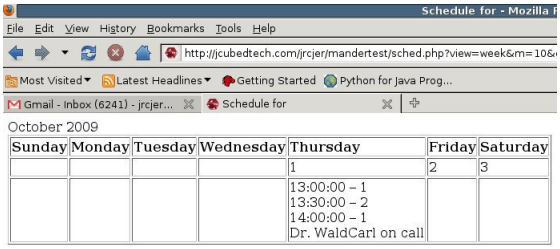

Figure 2: A screenshot of the first quarter Physician Schedule (week view) screen.

physician who frequently performs mission work in third-world countries. Although his mission would not necessarily benefit from an EMR system, he is in the process of contacting affiliated medical clinics and hospitals in these countries to see if such a system might benefit them.

5 Results

5.1 First Quarter

First quarter work on the EMR system was largely confined to exploratory work with PHP/MySQL setups. Basic tasks for EMR systems, including adding a patient, searching for a patient, scheduling a patient (Figure 1), and viewing a schedule by month or by week (Figure 2), were implemented. While little code from this experimental phase was used in the final EMR design, implementing EMR *screens* fostered MySQL fluency and understanding while prototypes for the final *screens* were considered. Because these constructions of EMR tasks were not designed to be integrated into the final project itself, unstylized HTML forms were used for the practice *screens*.

5.2 Second Quarter

Second quarter work on the EMR system was spent in the design and early implementation phases of the final, "Mander" EMR system. First, a personal web server was created for this project, implementing the LAMP (Linux, Apache, MySQL, and PHP) system

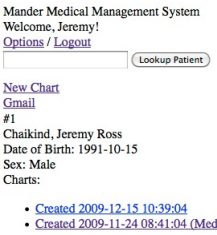

Figure 3: A patient's "Facesheet," displaying all of his/her identifying information and any *Chart*s he/she might have

to support the languages used in earlier stages of this project. Next, the final database design for Mander was created based on the relational model, keyed on a global user ID that would link all patients, physicians, and users to a *Names* table, in which crucial information for patient identification would be stored. This key would be associated to any other table in which a patient may have data. For example, a user would be associated to a username and password in the *Users* table, or a patient would be associated to the various levels of his/her virtual chart. The virtual chart is organized in a hierarchical manner. In this construct, a patient may have multiple *Chart*s (with all related medical information for a patient), which might have many *Sheet*s (each representing a specific interaction with a patient), each of which might contain a few *Record*s (which reflect only a single type of data [i.e. templated data, plain text data, or associated file data). This database structure was tested throughout the second quarter using a Create New Chart page, which used PHP functions to create rows in the *Name* and *Chart* tables, search functionality that accessed these tables to look-up a patient by name, a *Facesheet* page that displayed all of a patient's *Chart*s, and a *Chart View* page that listed all the *Sheet*s in a *Chart*. Initial work began on a templating system, exploring both WYSIWYG-creation and RTF-parsing approaches. However, the difficulties in implementing such a system quickly became too numerous to address as a small feature of a year-long project, and the automated templating system was postponed. Finally, basic user interface design began for the Mander EMR, including development of a navigation bar, page layout, and other tweaks, all of which were tested seperately from the rest of the project.

5.3 Third Quarter

The first two quarters of this project were spent largely in design phases. The third quarter was entirely different, focusing on high-level implementation of the database design paradigm. Work began with basic template creation. Although the researcher realized the infeasibility of using a WYSIWYG for easy, user-based creation of templates, the idea of using graphical templates for end-user data entry was still a crucial tenet of the project. Initial attempts to digitize the paper template began with hand-coded HTML. Due to the design of the paper template being replicated, nested tables were needed to properly implement a digital fabrication of the paper template. Despite the confusing nature of table-based positioning, such a system was the only reasonable way to implement Dr. Kern's template. The template was successfully coded in HTML and CSS, reproducing the design of of Dr. Kern's paper template without form inputs. While hand-coding the template was the simplest way to begin the template design process, the code it produced was difficult to understand and debug, and the inevitable task of adding text inputs to the template would be difficult, tedious, and mistake-prone. To avoid these issues, a library of PHP functions was created to generate appropriately-formatted text inputs and organizational elements. Functions were created to generate text, textarea, checkbox, autocomplete, date inputs, as well as to generate code for the overall page layout (including an invaluable function enabling the generation of an HTML table from a 2D array in PHP) were created. Alongside development of the library, Dr. Kern's template was redeveloped using the new functions, and later a paper template by neurologist Faye Rosenbaum, M.D. was coded using the functions.

Next, research began into the use of Javascript to enhance the user interface of the templates. An initial experiment to allow multiple text inputs for a single form field (i.e. multiple allergies for a patient could be listed in this way). Javascript enabled inputs to be added or removed by clicking a plus or minus button next to the field in question. While this experiment was not used in the final version of Mander, it served as an introduction to the Javascript language. More useful was the creation of an autocompletion input, which would allow the end user to automate the entry of commonly-used phrases (i.e. medications, allergies, and other words and phrases a physician might have to use often). Initially, simple AJAX (*A*synchronous *J*avascript *a*nd *X*ML) transmitted plain-text results via the PHP echo command. Discussions with Jason Koenig led to the adoption of an AJAJ- (*A*synchronous *J*avascript

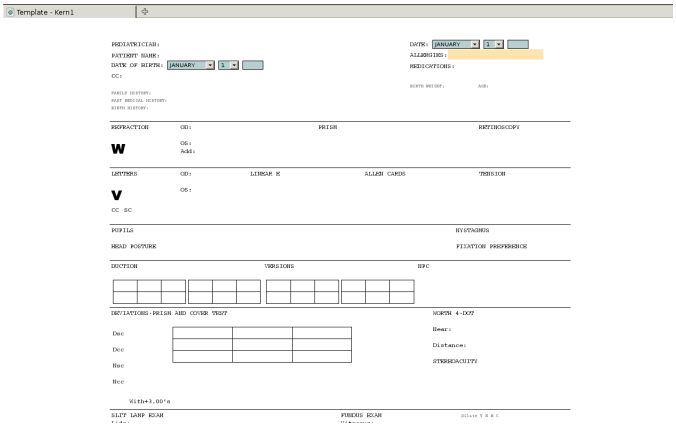

Figure 4: A screenshot of Dr. Kern's hand-coded template.

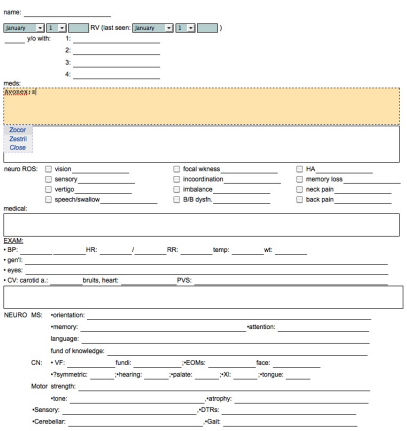

Figure 5: A screenshot of Dr. Rosenbaum's function-based template.

*a*nd *J*avascript Object Notation [JSON]) based implementation, which would allow PHP and Javascript to pass information in arrays, allowing for easy manipulation of the suggestions. JSON functions provided by Koenig enabled the implementation of the final autocompletion input. Further modification of this system was done to enable multiple, semicolon-spliced autocompletion answers (i.e. multiple medications in the same input).

5.4 Fourth Quarter

The work done in the final quarter of the year focused on combining the structural, database development of the first two quarters with the graphical template completed in the third quarter. The hierarchical database structure had been implemented through the *Sheet* level, but implementation of the fundamental *Record* level was necessary to allow the database to store any day-to-day medical data. The first implementation of a *Record* was a mechanism to attach image files to the database. This would allow physicians to link any relevant photographs or drawings to the database. More importantly, the ability to attach images would allow a physician to scan medical data that was created on paper (i.e. reports from other offices, past paper records for the patient) into the EMR system. To ensure medical privacy, a system was devised to assign each uploaded image a randomly-generated filename. This image would be wrapped in a PHP file that would require proper authentication to view it. This file would then be referenced in the necessary table in the database so that the *Record* could be displayed. Additionally, a generic *records* table was created to store information common to all types of records (i.e. add/delete data). This image storage system, while important in its own right, was perhaps most useful as a test implementation of the database's *Record* level.

Most of the fourth quarter was spent implementing the templates created in the third quarter. Even before the templates accessed the database, change needed to be made in order to properly display the template. Functions created in the second quarter to generate the consistent environment for the Mander system as a whole was modified so that the CSS and Javascript code from the templates could be properly implemented. A file was created for part of template form processing, but the function to access the database and submit the query for the template was appended to the end of the individual template file. Because each template is unique and may use different inputs than other templates, the most convenient way to process these data was from this template file. As for the linked-file *Record* type, the database submission for a template includes adding the data

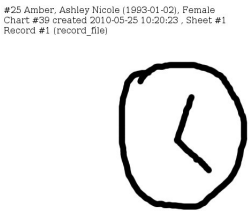

Figure 6: An example of a drawing made by the patient as part of an exam. This drawing has been attached into this patient's chart.

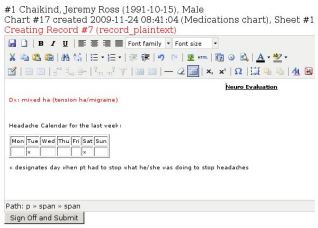

Figure 7: An example of a doctor's report written in TinyMCE.

to a table unique to the template, but also requires the creation of a row in the generic *records* table to interface with each sheet's general *Records* list. Attempted implementation of Dr. Rosenbaum's template also required the use of two additional tables to support the multiple-selection checkboxes with associated text used. In the process of developing the *record_template*-related functions, user interface elements were designed to allow the end user to easily create *Sheets* within a *Chart*.

One final feature added to the database system during the fourth quarter was the implementation of the TinyMCE WYSIWYG editor to allow a user to write formatted notes. This would permit a physician to write up his/her report about a patient easily from within the Mander system. Prior experimentation with TinyMCE for attempted use in template design provided preliminary code from which to start. Because TinyMCE automatically converts its rich text formatting to HTML code that can be sent as plain text, the creation of a system to allow rich-text notes required only minor modifications to the database and the functions needed to access the database.

5.5 Testing

For the First Quarter and Second Quarter stages of the project, false data were used. Approximately 20 fake "patients" including the researcher, various friends and family members, and various computer science figures, were added as test data. For the First Quarter, physicians inputed into the system were based on the names of doctors with whom the experimenter is familiar. First Quarter pages were run on a private, remote web server using preinstalled PHP and MySQL support. PHP files were written through a browser-based text editor provided by the company maintaining the web server. MySQL databases were managed through PHP MyAdmin, also preinstalled on the remote server. Second quarter work was written to a personal LAMP server through the Secure Shell (SSH) protocol. MySQL databases were also manages through PHP MyAdmin, installed on the server by the researcher.

Third Quarter testing was primarily done without saved data. Because the user interface was almost entirely disconnected from the database during development, no data were used for testing. An exception was made for the autocomplete inputs, which connected to a database table. Allergy autocomplete suggestion data were written by the researcher; common medication autocomplete suggestion data were based on a list of 25 common medications a patient would be taking provided by Dr. Rosenbaum.

Because work done during Fourth Quarter again used the database, the same false patients created from First and Second Quarter were used. Test data used for testing the new templates were primarily random strings, integers, and selections. Future testing with Dr. Kern and Dr. Rosenbaum will determine the fesibility of the Mander EMR system for managing true medical data, although the true medical data are expected to behave in the same manner as the false testing data used.

6 Discussion

First and Second Quarter work, while relatively unexciting, serves as necessary groundwork for this project. First Quarter work was primarily used as an experiment in PHP/MySQL development and initial prototypes for database organization. Second Quarter work was much more important to the final application itself. In creating the LAMP server, the library of database functions, and overall design of the database, the work done Second Quarter set up the all the theory to be applied later.

Third Quarter work differed from that completed earlier in the year because of its primarily high-level focus. In stark contrast to the structural work of Second Quarter, the work in Third Quarter was all centered around designing the user interface and user experience for the Mander EMR. Projects as focused as creating an autocomplete input or the live add field system were a necessary part of Third Quarter work. Broader scale work such as the creation of the templating-functions library allowed for some of the most visible results of the year.

Fourth Quarter work was necessary to bring what was done in the previous three quarters together into a coherent, functional medical records system. For the first time, the Mander EMR system was able to store real medical data entered as linked image files, template-generated data, and rich-text notes. Linkage of templates from Third Quarter with the database design from First and Second Quarter enabled a few automations to be developed, such as automatically filling in the patient's name, date of birth, and age, as well as the date the note is created and any other information pulled from previous records that is specified in the template. The work done this quarter successfully allowed the Mander EMR to become a functional, prototype medical records system.

7 Conclusion

The goal of this project is simple: to prove the feasibility of a modern, web language-based Electronic Medical Records system. The scope of the project has fluctuated greatly over the year as the researcher has better grasped the time required to complete certain tasks. This project will not yield a marketable EMR system by the end of the year; such a goal is too difficult to achieve in the time provided. However, the pieces of a strong database design and an intuitive user interface have been created this year. First and Second Quarter work developed a strong database design using relative and hierarchical architecture to ensure a highly scalable system, while use of the ACID paradigm guided the creation of the reliable, persistent database system necessary for the medical field. Third Quarter work developed a strong foundation for a graphical user interface to enter medical data using a mechanism that closely resembles paper records, minimizing the changes physicians must make to transition to electronic records, while implementing autocompletion and other "shortcut" features to create intuitive, automated solutions for everyday tasks for physicians. Fourth Quarter work connected the strong database design to the unique user interface to create a functional prototype of a modern EMR system. There is no question that this EMR system developed is not sufficiently robust to compete in the market today. However, Mander does illustrate that it is possible to create a medical records system with modern program languages and design principles that takes into account ease of use and, more importantly, ease of transition for physicians used to paper medical records. If these features of Mander can be considered in future EMR evolution, replacing paper charts with electronic ones may be a faster, less complicated process.

References

[Butcher] Butcher, L. (2010, March 4). CMS Proposes Criteria to Qualify for EHR Incentives of Up to $44,000. Neurology Today, 10(5), 26. doi:10.1097/01.NT.0000369550.82449.2b. This source was used to provide background information regarding the government policies supporting EMR/EHR system adoption.

[McWey] Chaikind, Jeremy. (2009). [Interview with Russell McWey, M.D., physician and technical

consultant at the Virginia Hospital Center in Arlington, VA]. This source was used to determine that encryption would not be necessary for an intraoffice EMR to maintain HIPAA compliance.

[Davis] Davis, J. (2005, February 9). Is your storage management process HIPAA compliant? In TechRepublic. Retrieved from http://articles.techrepublic.com.com/5100-10878_11-5567432.html This source was used as an introductory look into the security measures needed for HIPAA compliance.

[Haerder] Haerder, T., & Reuter, A. (1983, December). Principles of Transaction-Oriented Database Recovery. Computing surveys, 15(4), 287-317. Retrieved from http://portal.acm.org/citation.cfm?id=289.291 This source was used for extensive information about the ACID paradigm and its implementation.

[Hristidis] Hristidis, V., Clarke, P. J., Prabakar, N., Deng, Y., and White, J. A., M.D.(2006, November 11). A Flexible Approach for Electronic Medical Records Exchange. Retrieved from School of Computing and Information Sciences, Florida Intenational University website: http://portal.acm.org/citation.cfm?id=1183568.1183576&coll=Portal&dl=ACM&CFID=52786661&CFTOKEN=17622714 This source provided some background into the implications of the HIPAA Privacy Rule on EMR transfer. It also provided some background into methods of data transfer for EMR.

[Karger] Karger, D. R., Ostler, S., and Lee, R. (2009). The web page as a WYSIWYG end-user customizable database-backed information management application . UIST '09: Proceedings of the 22nd annual ACM symposium on User interface software and technology, 257-260. doi:10.1145/1622176.1622223 This source provided information about the WYSIWYG-based active content editor.

[Seyed-Abbassi] Seyed-Abbassi, B. (1993). Object oriented relational database with SQL interface. In Proceedings of the 1993 ACM Conference on Computer Science (Indianapolis, Indiana, United States, February 16 - 18, 1993). CSC '93. ACM, New York, NY, 497-504. DOI= http://doi.acm.org/10.1145/170791.171128

["Summary of the HIPPA"] Summary of the HIPAA Privacy Rule. (n.d.). Retrieved from Office for Civil Rights, US Department of Health and Human Services website: http://hhs.gov/ocr/privacy/hipaa/understanding/summary/index.html This source was used as an introduction to the regulations for physicians' offices created by the HIPAA Privacy Rule.