Creating a Modern Electronic Medical Records (EMR) System TJHSST Senior Research Project Quarter 3 Computer Systems Lab 2009-2010

Jeremy Chaikind

April 6, 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. 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 and low-featured. This project plans to remedy the situation by creating an EMR system designed in conjunction with physicians to ensure ease-of-use, using forward-thinking web-based languages, including PHP, HTML, CSS, and MySQL.

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.

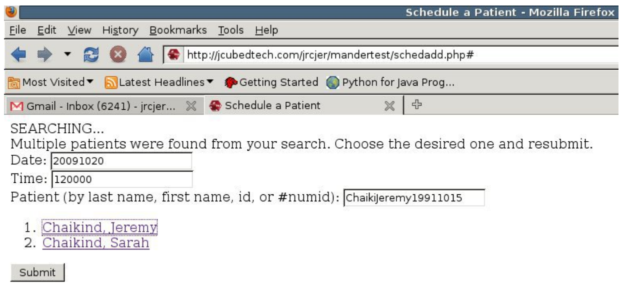
Durability ensures the integrity of all data by requiring database data to survive any malfunction. This could be accomplished with relative ease by instituting measures of redundancy and requiring multiple sources to match in order to display information. To prevent data loss on a larger scale, database 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 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.


Schedule a Patient - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Most Visited Latest Headlines Getting Started Python for Java Prog...
Gmail - Inbox (6241) - jrcjer... Schedule a Patient

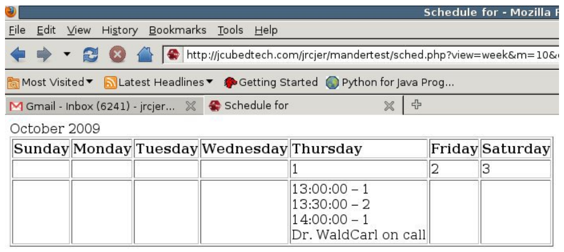
http://jcubedtech.com/jrcjer/mandertest/schedadd.php#

SEARCHING...
Multiple patients were found from your search. Choose the desired one and resubmit.
Date: 20091020
Time: 120000
Patient (by last name, first name, id, or #numid): ChaikiJeremy19911015

1. Chaikind, Jeremy
2. Chaikind, Sarah

Submit

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


Schedule for - Mozilla F
File Edit View History Bookmarks Tools Help
Most Visited Latest Headlines Getting Started Python for Java Prog...
Gmail - Inbox (6241) - jrcjer... Schedule for

http://jcubedtech.com/jrcjer/mandertest/sched.php?view=week&m=10&

October 2009

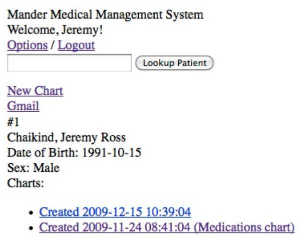
Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
				1	2	3
				13:00:00 – 1 13:30:00 – 2 14:00:00 – 1 Dr. WaldCarl on call		

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

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*.


Mander Medical Management System
Welcome, Jeremy!
Options / Logout

New Chart
Gmail
#1
Chaikind, Jeremy Ross
Date of Birth: 1991-10-15
Sex: Male
Charts:

- Created 2009-12-15 10:39:04
- Created 2009-11-24 08:41:04 (Medications chart)

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

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 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 *Charts* (with all related medical information for a patient), which might have many *Sheets* (each representing a specific interaction with a patient), each of which might contain a few *Records* (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.

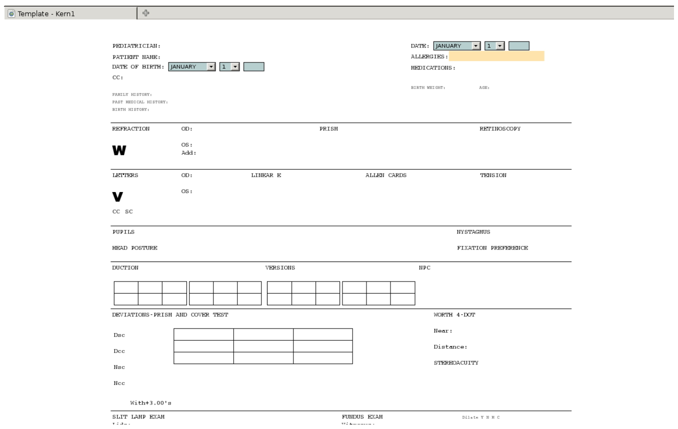
5.3 Third Quarter

The previous two quarters of this project were spent largely in design phases. A prototype experiment exploring PHP and MySQL was conducted in first quarter. The database design and interface functinos were finalized durnig the second quarter. 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 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.

Next, research began into the use of Javascript to enhance the user interface of the templates. First, exploration began on a way to allow multiple text inputs for a single construct. Suppose a patient had not one but 5 drug-related allergies. The physician should be able to note this, just like the physician should be able to write all of a patient's allergies. The solution adopted was to use a Javascript function to write a new instance of the text input each time a button was pressed. If the physician added too many inputs and needed to remove one, he/she would simply have to click the button to remove inputs. Each input would have the same name with a number at the end to signify the instance of that element. While this user interface improvement has yet to be implemented, its creation could be used in future templates, and, more importantly, it served as an introduction to the Javascript language with which the researcher was unfamiliar.

The more useful UI element created this quarter was the autocompletion input. This element would be used to allow the end user to automate the entry of commonly-used phrases. For example, a physician may ask a patient what medications he/she is taking. An autocomplete input could be used so that if the medication is one commonly used (i.e. Asprin, MVI, etc.) the physician does not need to type out the full name each time this phrase is used. An autocomplete input is best created using a variant on AJAX (*A*synchronous *J*avascript *a*nd *X*ML) to transmit the typed letters to a PHP page, which looks up the query in an SQL table, and prints the query, returning it via AJAX to the original HTML page. Initial work on the autocomplete system transmitted plaintext information instead of XML. This system easily allowed an autosuggestion, but not autocomplete feature. Discussions with Jason Koenig led to the adoption of an AJAJ- (*A*synchronous *J*avascript *a*nd *J*avascript Object Notation [JSON]) based implementation, which returned a text string to the HTML page which the in-page Javascript would be able to parse as an array, allowing for easy manipulation of the suggestions. JSON functions provided by Koenig enabled the implementation of a true autocompletion input. Further modification of this system was done to enable multiple, semicolon-spliced autocompletion answers (i.e. multiple medications in the same input).

Development continued with a revision of templating. 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 set of PHP


Template - Kern1

PEDIATRICIAN:
PATIENT NAME:
DATE OF BIRTH: JANUARY 1
CC:

FAMILY HISTORY:
PAST MEDICAL HISTORY:
BIRTH HISTORY:

DATE: JANUARY 1
ALLERGIES:
MEDICATIONS:

BIRTH WEIGHT: AGE:

REFRACTION OD: PRISM RETINOSCOPY
W OS:
Add:

LETTERS OD: LINEAR E ALLEN CARDS TENSION
V OS:
CC SC

PUPILS NYSTAGMUS
HEAD POSTURE FIXATION PREFERENCE

DUCTION VERSIONS NPC

DEVIATIONS-PRISM AND COVER TEST WORTH 4-DOT
Dsc Near:
Dcc Distance:
Nsc STEREOACUITY
Ncc
With+3.00's

SLIT LAMP EXAM FUNDUS EXAM Dilate Y N M C

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

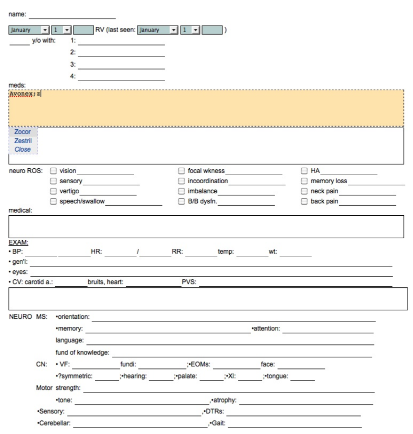

name:
January 1 RV (last seen: January 1)
y/o with: 1:
2:
3:
4:
meds:
Avonex; z
Zocor
Zestril
Close
neuro ROS: vision focal wkness HA
sensory incoordination memory loss
vertigo imbalance neck pain
speech/swallow B/B dysfn. back pain
medical:
EXAM:
• BP: HR: / RR: temp: wt:
• gen'l:
• eyes:
• CV: carotid a.: bruits, heart: PVS:
NEURO MS: •orientation:
•memory: •attention:
language:
fund of knowledge:
CN: • VF: fundi: ;•EOMs: face:
•?symmetric: ;•hearing: ;•palate: ;•XI: ;•tongue:
Motor strength:
•tone: ,•atrophy:
•Sensory: ,•DTRs:
•Cerebellar: ,•Gait:

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

functions were created to generate appropriately-formatted text inputs and organizational elements. Each function is specific enough to write code that would generate a polished, unique user interface, yet general enough to allow broad application for any Mander EMR template. In addition to functions to create the various types of form inputs (i.e. text, textarea, checkbox, autocomplete, date [comprised of a month drop-down box, a day drop-down box, and a four-digit text input]), functions for overall page layout were created. A function to create different, bottom-bordered sections of the template, as well as an invaluable function which would properly write the HTML to translate any two-dimensional array into a table, were created. Alongside development of the PHP library, a copy of Dr. Kern's template was developed using the new functions. Because of a few formatting tweaks, the function-generated template approximated the paper template even better than the hand-coded one. In order to further test the viability of the template-generation library and allow for preliminary user interface testing to begin, a second template was created from the library functions based on the paper template of neurologist Faye Rosenbaum, M.D.

5.4 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.

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.

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 in these first three quarters. In the Fourth Quarter, the database and user interface will finally be integrated, and the product of the union can be expected to prove that an EMR can be created with modern programming languages and techniques, as well as an easy to understand user interface for intermediate and end users.

References

[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.