A Secure Healthcare System: From Design to Implementation
Ebru Celikel Cankaya\textsuperscript{1} Than Kywe\textsuperscript{2}

\textsuperscript{1, 2}: University of Texas at Dallas Department of Computer Science, Richardson, TX USA
{exc067000, txk102220}@utdallas.edu

Abstract

We introduce the design and development of a comprehensive electronic health record system (EHR) that incorporates AES encryption to assure security. Our work adopts a didactic approach to introduce the formal design steps of an EHR with its underlying database from a software engineering perspective. For this, we adopt two formal development methodologies as software engineering perspective and database development approach and combine the two to present a guideline to design and develop similar projects in other domains. For informative purposes, the steps of the development process are formalized based on database ER-model, and the final design is normalized into 3NF. We provide insight on rationale for employing specific methodologies, and using particular material and tools.

1. INTRODUCTION

With advances in healthcare informatics, one can provide better means to process patient records and therefore speed up the treatment, which in turn reduces the overall cost. Many tools exist for facilitating patient record processing: from assisting data entry to manipulating records, from generating output in required form to transferring it to other physicians for further examination, or to save it digitally for future use. The significance of our work relies on the fact that we bring a software engineering oriented systematic approach to design and develop an electronic health record system (EHR). Our scheme is fundamentally a computer based patient record (CPR) system \cite{1, 2, 3}. In particular, this work extends a CPR system by incorporating a software engineering approach during development, and incorporates many aspects from database design, web deployment, and security.

While developing our design, we follow formal patient privacy requirements for healthcare systems that are enforced by Hipaa \cite{4, 5}. 
A healthcare system is much complicated and comprehensive as it involves many aspects from hospital management to staff tracking. We focus particularly on patient record manipulation, as even this concentration is sufficiently complex to provide a comprehensive analysis of data, database design, and implementation steps.

2. RELATED WORK

The need for a comprehensive electronic health record (EHR) system is pervasive, explaining the reason why there are many implementations as mentioned in [6, 7, 8, 9, 10]. Yet, each implementation focuses mainly on certain aspects while disregarding the others due to such factors as time, performance, number of users, user acceptance, and state or nationwide policies [3, 11, 12, 13, 14, 15, 16]. Actually, considering an EHR w.r.t. each factor listed above could be a topic of research itself, and is beyond the scope of this paper.

In [6], Lien et al. introduce an EHR system similar to our work. Additionally, our design integrates security to the EHR system by incorporating cryptology. When a user logs into the system, our system first checks the username in the backend database. If found, then corresponding encrypted password is retrieved from database and is decrypted. Next the user entered password is compared with the decrypted password. If they match, the password is valid and the system proceeds by checking the user permission level to grant appropriate access. Otherwise, the access is denied. [7] presents another EHR system designed to use for several types of databases with mobile applications via j2se program. However, this work also is missing a cryptography feature. The main advantage of our design is that we apply java cryptography architecture throughout so as to store all security sensitive data in the form of a cipher. When we retrieve this sensitive data from database, we decrypt it based on the user access level.

Other similar EHR system applications utilize different underlying infrastructure for the operating systems, as well as the database. For example, [17] designs an EHR for web based and mobile platforms, which runs on a Linux (Ubuntu 10.04) operating system with MySQL database and ApacheTomcat. Our design runs on a newer version of Linux (Ubuntu 13.10) with Oracle 10g database.

Observing patient privacy is the fundamental requirement that an EHR system should possess. And many EHR system designs practice privacy preservation through HIPPA policy that remains only at the database level [18]. While manipulating sensitive patient records within database level as other similar systems do, our system extends secure data handling to user interaction level as well.

The concern to provide and maintain security of sensitive patient data is the main concentration of our system. For this, we adopt a multi-tiered architecture where the main ideology is to protect data anywhere at any time. This simply means that data must be protected whether it is stored in a database server or in the cloud; or is interacting with users in any means—from client computer, or from the web browser, or from mobile devices.

3. SYSTEM DESIGN AND DEVELOPMENT

The overall diagram of our design is illustrated in Figure 1 below.

Figure 1. Overall system schema for the EHR design.
As can be seen from Figure 1, there are five essential components in our design: The User interface provides a GUI based interaction with the end user who is either a physician, or a nurse. The JDBC API offers database connection services, while Swing API supports the GUI services. The Java Crypto API is employed to complement the application with AES symmetric encryption. The last component is the Oracle database that lies at the bottom of the abstraction hierarchy to store patient data records, as well as encrypted content, such as user login data, disease categories, physician information, and patient records for visits and treatment.

The formal categorization and definition of the entities in our design are listed as follows:

- **Patient**: Subject of medical records (or agent).
- **Patient health data**: Data about patient's health or treatment enabling identification of patient.
- **Healthcare provider**: Mostly the physician, who has access to personal health information. The other type of a provider is nurse.
- **Medical Office Staff**: The staff of the healthcare center that help manage and maintain information regarding healthcare providers and patients.

For implementation, we use Netbeans IDE 8.0, JDK 1.7, and Oracle 10g. The reasons behind these choices are concerns on cost effectiveness, as well as performance. We see that these choices prove to be well made, as we get responses from our system with no major problems in a timely manner.

### 3.1. REQUIREMENTS GATHERING AND ANALYSIS

We start designing our EHR system by gathering the facts and requirements about a generic patient entity first. From a software engineering perspective, a patient record should store information about patient’s identity differentiated by a unique patient id (for this, SSN is used), address, contact phone number, date of birth, gender, blood type, and a special field to add remarks about the patient.

As for the manipulation of a patient health data, three roles are defined: a physician, a nurse, and a staff member. A physician can add, modify, or delete his treatment records for a patient. He/she can view the complete treatment records of the patients assigned to him/her. The patient health data should also be manipulated by a nurse, though with lesser rights granted this time. Namely, a nurse should be able to add health related treatment record for a patient. A nurse should also be able to browse the visit history of a patient in a restricted way, which means that a nurse cannot view the detailed treatment records of physician(s) about a patient. The third role owner, i.e. a staff member, can view the patient’s basic information related to his identity, as well as his visit date and time, and the cost of visit. The staff should also be able to track patient wait time, and assign an appropriate and available healthcare provider to the patient.

In addition to the three roles defined above, the system also introduces an admin role, who is responsible for defining roles, assigning users to these roles and managing role granting and revocation on a dynamic basis (as physicians/nurses/staff can be hired, or may leave the job, etc.).

### 3.2. DESIGN

After initial gathering of facts and requirements completed in Section 3.1., the collected data is further analyzed to formally define the database tables, fields of each table, primary and foreign key assignments, relationship and functional dependency definitions.

To exercise the design phase of the formal software engineering approach, we combine it with the formal database design step that involves the steps described in subsections 3.2.1 through 3.2.4:

#### 3.2.1. ER DIAGRAM

The Entity-relationship (ER) diagram first formally defines and constructs the entities and relationships in the system, then brings them together with relationships specified.
Figure 2 shows the overall ER diagram for our EHR system:

As seen in Figure 2, the entities in our EHR design consists of fundamental components, rather than complex entities; and the relationships are simple accordingly. This is a deliberate design choice to make the process less complicated and easy to follow for educational purposes.

3.2.2. RELATIONAL DATABASE SCHEMA

The relational database schema given in Figure 3 displays a schematic view of the entities and the cardinalities associated with each entity.

As we see from Figure 3 the relational database schema illustrates the data type and domain information pertaining to each entity and relation, where the legend of symbols are listed as follows:

As we see from Figure 3 the relational database schema illustrates the data type and domain information pertaining to each entity and relation, where the legend of symbols are listed as ♦ to indicate primary key, ○ to indicate a non-nullable field, and ◊ to indicate a nullable field.
3.2.3. DATABASE CONSTRAINTS

Cardinality Constraints

Employee-Doctor 1:M

Employee-Nurse 1:M

1:M relationship between Employee and Doctor and Employee and Nurse as a there could be many doctors as employees, and one doctor can only register once as an employee. The same cardinality applies to the relationship between Employee and Nurse entities.

Patient-Doctor (M:N)

M:N relationship between Patient and visiting a doctor, as a patient can visit many doctors, and the same doctor can be visited by many patients.

Doctor manipulates visit history (M:N)

M:N relationship between Doctor and manipulating visit history, as a doctor can manipulate many visit histories, and the same visit history can be associated with more than one doctor.

Patient has visit history (1:M)

1:M relationship between Patient and having a visit history, as a patient have many visit histories, and the same visit
history can only belong to one patient.

**Key Constraints – Primary Key (PK)**

SSN is the PK in Employee table, as it uniquely identifies each employee.

DoctorID is the PK in Doctor table, as it uniquely identifies each doctor.

NurseID is the PK in Nurse table, as it uniquely identifies each nurse.

PatientID is the PK in Patient table, as it uniquely identifies each patient.

VisitID is the PK in VisitHistory table, as it uniquely identifies each visit history.

VisitControllerID is the PK in VisitController table, as it uniquely identifies each visit controller.

**Referential integrity constraints - Foreign Key (FK)**

DoctorID is FK in VisitController.

NurseID is FK in VisitController.

PatientID is FK in VisitController.

VisitID is FK in VisitController.

SSN is FK in Doctor.

SSN is FK in Nurse.

### 3.2.4. DEPENDENCY DIAGRAMS

Doctor Table: (DoctorID, SSN, Qualifications, Speciality)

Corresponding Relational Schema for the Doctor table in 3NF

![Doctor Diagram]

Nurse Table: (NurseID, Qualifications, Speciality)

Corresponding Relational Schema for the Nurse table in 3NF

![Nurse Diagram]

VisitHistory Table: (VisitID, VisitDate)

Corresponding Relational Schema for the VisitHistory table in 3NF

![VisitHistory Diagram]
Employee Table:
(SSN, FirstName, LastName, Address, TelNo, Sex, DOB, LoginName, LoginPassword, AccessLevel).

Corresponding Relational Schema for the Employee table in 3NF:

```
| SSN | FirstName | LastName | Address | TelNo | Sex | DOB | LoginName | LoginPassword | AccessLevel |
```

Patient Table: (PatientID, SSN, FirstName, LastName, Address, TelNo, Sex, DOB, BloodType, Remarks).

Corresponding Relational Schema for the Patient table in 3NF:

```
| PatientID | SSN | FirstName | LastName | Address | TelNo | Sex | DOB | BloodType | Remarks |
```

VisitorController Table:
(VisitorControllerID, DoctorID, NurseID, VisitID, PatientID, TreatmentRecord).

Corresponding Relational Schema for the VisitorController table in 3NF:

```
| VisitorControllerID | DoctorID | NurseID | VisitID | PatientID | TreatmentRecord |
```

3.3. IMPLEMENTATION

The implementation section starts with designing GUI for the end users.

The initial login screen seen in Figure 4 below provides a means to collect user credentials for further verification by checking the already stored encrypted user passwords in the system.

Figure 4. Initial login screen and encrypted credentials.
After a successful login, the user is displayed the main screen of operations in Figure 5 that provides the menu of options he can do, based on his role in the system:

![Figure 5. Admin screen of operations.](image)

The screen displayed in Figure 5 captures the current user name and his role, and displays it together with the access time in status bar, for log purposes. Based on the role of the user, the functions that can be accessed are different: If the current user role is physician (Figure 6), then he can access the doctor and patient options from the main menu. If the current user is a nurse, he can access the nurse and patient options from the main menu. Similarly, if the current user is a staff, he can access the office admin menu. When the current user is an admin, he can view the log data regarding who accesses the system at what time, for what purposes.

![Figure 6. Physician screen of operations.](image)

The process of patient registration is accomplished by a staff member via following screens in Figure 7:

![Figure 7. Main screen of operations.](image)
The staff member can add, modify, or delete a patient record in the system. Figure 7 shows a partial view of the patient registration records listed in ascending order of FirstName column.

The admin can view and check the system login information for error detection purposes. The login information (Figure 8) is directly associated with the encrypted log file displayed next to it. A symmetric encryption algorithm AES [19] along with 128 bits is used to encrypt the log file so as to provide the confidentiality of the log file from unintended third parties. The rationale behind using AES is twofold: Once it is fast (as compared to an asymmetric encryption algorithm), and also computationally hard to break due to its comparatively large key size (128 bits).

Addressing the patient together with patient health data entries, we create a patient visit records screen as seen in Figure 9:

This screen allows a nurse to add treatment record(s) for a patient, as well as allows him to view the visit history of a patient in a limited manner (most probably, not to the degree that a physician is granted). The same screen allows a physician to add treatment record(s) for a patient, and also view the complete treatment records of the patients assigned to him. A staff member can also view patient basic information together with visit date and time for tracking purposes. The staff can also assign an available and a matching (meaning a specialist for a particular disease) health care provider to each patient.

4. CONCLUSION AND FUTURE WORK

We present the design and development of a novel EHR system incorporating two formal development methodologies as software engineering perspective, together with database development approach. This work...
provides a concrete example that combines these two formal methodologies to design and develop software, and can be used as an educational guideline to model after for designing similar projects in other domains. As part of future work, we plan on extending our design to provide an interactive tool that will help user get involved actively in the design process and self teach its steps.

REFERENCES
A Secure Healthcare System: From Design to Implementation

CC BY-NC-ND 4.0 License
©2016 Elsevier B.V.

Citation: