Självständigt arbete på grundnivå

*Independent degree project - first cycle*

Datateknik GR (C), Examensarbete  
*Computer Engineering BA(C), Final Project 15 credits.*

Android application of Doctor’s end in E-health system

Lingzhen Chen
Android application of Doctor’s end in E-health system

Lingzhen Chen

2013-07-13

MID SWEDEN UNIVERSITY
the Department of Information Technology and Media (ITM)

Examiner: Ulf Jennehag, Ulf.Jennehag@miun.se
Supervisor: Tingting Zhang, Tingting.Zhang@miun.se
Author: Lingzhen Chen, lich1200@student.miun.se
Degree programme: Bachelor of Science in Engineering, 180 credits
Main field of study: Computer Engineering
Semester, year: 06, 2013
Abstract

In recent years, with the arrival of information era and the rapid development of the medical health system, there has been a trend for hospitals and medical care centers to record the patients’ medical record electronically rather than by means of old-fashioned paper-based method. It has indeed reduced the workload of medical workers and it simplifies the procedure within the medical system. It is worth mentioning that the growth of mobile medical care is beginning to catch up as a consequence of the incredible popularization of mobile networks and the broad use of mobile devices. A significant possibility is that in the coming years, mobile medical care applications will play a large part of the whole electronic medical care system.

This project will mainly focus on presenting the process of developing an Android application of doctor’s end in the electronic healthcare system which makes it possible for doctors to retrieve data from their patients and diagnose them as well as setting up appointments using mobile devices. The development for the doctor’s end application will follow the waterfall development methodology. In this report the functions of the doctor’s end application will be introduced, and how these functions are realized will also be displayed in detail. After the implementation, the app will be tested on an Android device emulator and on real Android mobile devices in order to evaluate their performance. In addition, it will be tested by mobile phone specialists in relation to gathering feedback. Finally, a reasonable conclusion will be drawn with regards to this project including ethical issues.

Keywords: Android application, E-health care, doctor’s client end.
Acknowledgements

First of all, I would like to express my gratitude to Professor Tingting Zhang, who has given me this valuable opportunity to participate in this project. As my supervisor, she patiently points out the imperfections in my project and offers me useful suggestions. She has also spent time tutoring me with regards to thesis paper writing.

Secondly, I would like to thank Xin Zhang, who has offered me significant help during the project procedure. Finally I want to thank everyone who has been working on the E-health care system. The system’s improvement is based on the efforts of everyone involved.
# Table of Contents

Abstract ................................................................. iii  
Acknowledgements ......................................................... iv  
Table of Contents ........................................................... v  
Terminology ................................................................. vii  
1 Introduction ..................................................................... 1  
  1.1 Background and problem motivation ................................. 1  
  1.2 Overall aim ..................................................................... 2  
  1.3 Scope ............................................................................. 2  
  1.4 Concrete and verifiable goals ............................................ 2  
  1.5 Outline .......................................................................... 3  
  1.6 Contributions .............................................................. 3  
2 Theory work ....................................................................... 4  
  2.1 Android developing platform ........................................... 4  
  2.2 Object-Oriented Programming ......................................... 5  
  2.3 SQLite Database .......................................................... 5  
  2.4 AES Encryption .......................................................... 6  
  2.5 User-friendly application guidelines ................................. 6  
3 Methodology ...................................................................... 8  
  3.1 Selection of development platform .................................... 8  
  3.2 Selection of development model ....................................... 8  
  3.3 Modularized function realization .................................... 10  
  3.4 Methods for Achieving user-friendliness ......................... 11  
  3.5 Methods for Testing & Evaluation .................................. 11  
4 Implementation .................................................................. 12  
  4.1 General Structure ....................................................... 12  
    4.1.1 Middleware API module ........................................ 12  
    4.1.2 GUI module .......................................................... 13  
    4.1.3 Login module .......................................................... 14  
    4.1.4 Data modules .......................................................... 14  
    4.1.5 Contact module ...................................................... 14  
    4.1.6 Database access and operation module ...................... 15  
  4.2 Main functions .......................................................... 15  
    4.2.1 Log in ...................................................................... 16  
    4.2.2 Getting patient’s data .............................................. 16  
    4.2.3 Giving feedbacks/advices .......................................... 16  
    4.2.4 Contact operations .................................................. 16  
    4.2.5 Schedule management .............................................. 17  
    4.2.6 User settings management ....................................... 17  
  4.3 Data flow & Data storage ............................................. 17  
    4.3.1 Data Flow .............................................................. 18  
    4.3.2 Data Storage .......................................................... 19  
4.4 Development Environment Setting .................................. 21  
4.5 Interaction among activities ........................................... 23  
5 Evaluation ........................................................................ 27  
  5.1 User-friendly ............................................................. 27  
  5.2 Performance .............................................................. 27
5.3 Privacy preservation ................................................................. 28
6 Conclusions ................................................................................. 29
References ....................................................................................... 30
Terminology

UML: Unified Modeling Language
GSM: Global System for Mobile Communications
AES: Advanced Encryption Standard
DES: Data Encryption Standard
MD5: Message-digest Algorithm
1 Introduction

Electronic health care involves the application of electronic technology within a medical care system, which means carrying patients’ medical records electronically rather than recording them on paper. Additionally, as the social medical services become more and more developed, the procedures that a patient must go through in order to gain access to treatment is becoming much more complicated. The electronization of medical information has provided a great deal of convenience to the medical services and has saved many troublesome intermediate procedures.

E-health is improving people’s lives in different ways. There has already been EHR (Electronic Health Record) [1] which is used by some hospitals and health care centers, in which a systematic collection of patient’s health information is gathered electronically. It has the ability to carry the patient’s personal data such as age, gender, medical history, allergies, immunization status, radiology images, test results and so on. In the Democratic Republic of Congo, there has been a telephone hotline [2] established for callers to receive medical advice. In Peru, there is a program sending patients with H.I.V text messages [3] reminding them to take their medicines. In addition, there are now a notable number of different kinds of medical websites for patients to consult doctors or physicians, book a visit to a doctor, and to search for hospital and medical care center information.

It is imperative that the trend towards the electronization of medical health service continues. In developed countries, the E-healthcare system can help to reduce medical costs and improve the service quality. Meanwhile, in developing countries, the E-healthcare system is used to solve problems associated with a lack of and the poor distribution of medical resources. Therefore, many countries are paying considerable attentions to the developing E-healthcare systems, especially those countries in America and Europe. The coverage of the service is quite complete. The government of the Unite States has already invested tens of billions dollars to carry forward the electronization of medical records. In Switzerland, an electronic health center [4] was established to provide online medical service for patients by gathering around 20 thousand doctors and physicians.

1.1 Background and problem motivation

Our research group has been working on E-health system for a long time. A functional system has been established on a PC-base. For the patient’s client end, users are able to collect their health data by means of a sensor and then upload them to server. The system will send the analysis results to the patients. A wireless sensor system has been developed for collecting health data. Xuchen Lu et al. proposed a heterogeneous data source middleware [5], which uses an abstract class to create a data source service wrapper. The implementation of this middleware enables different types of data communication between the data source and the end application. Xin Zhang et al. proposed a distributed electronic health record system [6] to transfer and store data in a standard
format and to enable data transmission between a server and different applications.

With the unstoppable growth of the mobile Internet, the electronic medical service with stationary devices is no longer sufficient to satisfy the increasing requirement of the people, who has now begun to rely on mobile networks more and more. This means that it has become imperative to implement an Electronic-health System on mobile devices. In this project, a doctor’s end application on an Android platform has been proposed, which will make it more convenient regarding the communication and information exchange between the doctor and patient.

1.2 Overall aim

This project is aimed at developing an Android application for the doctor’s end in an E-healthcare system with a user-friendly interface. The app will allow the connection between mobile end-devices and the server for data transmission. It will be integrated with the local database and a database on the server side, thus allowing the application to provide an information exchange between them. By using this application, users, i.e doctors, will be able to receive patients’ personal information and health data, provide feedback based on the received information, contact patients and set appointment reminders.

1.3 Scope

The realization of this application is concentrated on achieving of functions in relation to doctor’s end, which will contain the functions of retrieving patients’ health data from the server and in sending data, such as the messages, to the patient. The privacy policy, wireless data transportation, communication between patients and doctors with SMS will also be involved, but these aspects are being completed by others in the research group. The doctor’s end will combine all these work together and provide the users with an integrated application which allows them to obtain patients’ medical data more easily and will save both time and energy with regards to diagnosing and providing feedback to patients.

1.4 Concrete and verifiable goals

To achieve the overall aim of developing a doctor’s end application on Android platform, the entire developing procedure was divided into several sections and some concrete goals were set. Firstly, the doctor’s end application will allow a user to perform the following operations: a) logging in with a unique username and password; b) viewing patient information including patients’ basic information, contact information, health data, testing data concerning their diseases, etc; c) giving advice or feedbacks based on the data they have obtained; d) arranging schedules and setting appointments with patients; e) managing their basic setting such as changes to the username or password.

Secondly, a user-friendly GUI (Graphical User Interface) containing the following pages will be designed: a) login page: For users to login; b) home page: For users to see newly updated data and messages from patients; c) contacts page: For users to obtain contact information regarding patients, nurses,
Android application of Doctor’s end in E-health system

Lingzhen Chen

2013-07-13

and colleague doctors; d) schedule page: For users to see appointment schedule; e) settings page: For users to change their personal information such as username, password for logging in, etc.

Finally, the application will be tested on different devices and a rational evaluation concerning the performance and privacy preservation will be conducted.

1.5 Outline

This chapter introduced the background, overall aim, concrete goals and the scope of the project. In the next chapter, chapter 2, the relevant theories concerning this project will be introduced. Chapter 3 describes the problem-solving direction and methods in relation to developing the application. Chapter 4 provides a view regarding the idea relating to the design of this application’s functions, user interface and the implementation results. Chapter 5 discusses the evaluation of this application. Finally in 6, the conclusion of this project, including advantages, limitations and future work will be revealed.

1.6 Contributions

Through years of efforts by our research group, a PC-based E-health system has been established. Since this January, Zongzhe Chen, Fanchen Li, Jifeng Jin and the author of this thesis began the implementation of an E-health system on an Android platform. Zongzhe Chen is responsible for the setting of the privacy rules within the E-health system. Fanchen Li is in charge of the Bluetooth data transmission between the patient and the doctor’s terminals. Jifeng Jin mainly focuses on the scheduled SMS healthcare reminder application while the author of this thesis is responsible for the doctor’s end application.
2 Theory work

Before conducting the design and implementation of the app, some related work has been involved, including the background of Android, OOP, and the SQLite database as well as the AES encryption method and the criteria for a user-friendly application. This chapter will mainly introduce these related theories and techniques used for the development of the doctor’s end application.

2.1 Android developing platform

Android is a Linux-based operating system developed mainly for mobile devices. Google founded OHA (Open Handset Alliance) in 2007 with 84 hardware and software manufacturers and telecom service providers in order to develop and improve the Android system. They devoted to advancing open standards for mobile devices. The first Android smart phone arrived on market in 2008, after which the system also expanded into tablet PCs and other areas. According to the survey data at the end of year 2010 [7], the market share for the Android system has exceeded that for the Nokia Symbian system, which has dominated the market for over 10 years.

The Android system provides a built-in small size relational database—SQLite for data storage. The system supports all types of GSM, multiple languages, multiple media, multiple streaming media, and multi-tasking. Due to its portability and openness, the Android system can be used on the majority of electronic products.

During the early period, Java is usually used in Android SDK (Software Development Kit) for application development. It is also accessible for use with C/C++ in Android NDK (Native Development Kit). At the same time, Google released the Simple language and the Google App Inventor, both of which enable beginners to start in an easy manner.

The core system services such as security, internal memory management, process management depend on the Linux 2.6 kernel, which also performs as the abstract layer between the hardware and software stack. Developers can have full access to the core application API framework. In general, an Android application consists of activity, content provider, service, and intent. Activity acts as the dialog box of the app and the content provider provides the data storage for the app. Intent is used for the switch between activities. The Service is designed to enable the backstage to maintain running. The main developing language for Android is Java. A significant advantage of developing an Android app is that Android SDK is released to public. The third-party or freelance developer can develop Android apps conveniently. In addition, there is no restriction on the Android system in relation to hardware device and the portability of application is high, which solves the problem of incompatibility regarding of different file formats between different smart phones.
2.2 Object-Oriented Programming

OOP is chosen to be the main programming design idea in this project. OOP, Object-oriented programming, is a type of programming paradigm, in which the concept “objects” are usually instances of classes. The basic rule of OOP is to divide the whole program into several sub-programs, i.e units or objects. Objects are related with methods and several objects are usually used together to interact with each other in the design of applications and programs. [8] [9] The use of Object-oriented programming increases the flexibility, expansibility, and reusability of software. This, in turn, makes it easier for the program to be understood and analyzed and thus OOP is widely used nowadays in programming design.

In the traditional way of thinking, a program is a set of functions, a set of instructions that informs the computer what to do. In contrast, OOP considers every object as a “micro-machine”. Each object is able to interact with other objects for data exchange and message passing. [10]

OOP has many notable advantages compared to the traditional functional-programming or logic-programming. Object provides the concept of modeling. In the process of programming, an object is the substantialization of the abstract concept—class. In OOP, after a class has been created, it is able to be reused. This is called reusability, which saves a great deal of work regarding redefinition or other repetitive declarations. An object has its own attributes and methods. If these are set as private, then other objects cannot access to these attributes and methods, which is a very effective way of preventing confusion and the abuse of data.

To be specific, in this application, during the process of programming, many classes are used to realize different functions and these classes are able to be inherited and overloaded. Thus classes and subclasses are created in different modules, which are able to interact with each other in order to enable the whole program to be functional.

2.3 SQLite Database

Ever since the appearance of commercial applications, the database has been an important constituent part and it is becoming increasingly important, taking more system resources, and complicating the management. With many applications having modularized components, a new kind of database system is in required other than the traditional large-scale ones. SQLite [11] is an embedded database system, which is an integral part of the applications rather than being accessed from the client application. SQLite [12] takes up minimal internal resources and realizes zero-configuration running model. It is ACID (atomicity, consistency, isolation, durability)-compliant and implements the majority of the SQL standard, using a dynamically weakly typed SQL syntax that does not guarantee domain integrity. Unlike the usual client/server structure, SQLite is not a detached process of application; instead it is an important embedded part. Thus the main communication protocol is a direct API invocation in the program. This reduces the total resource consumption, latency and complexity. The whole database including definition, tables, indexes and
data record, is stored in a single file. A piece of data can be read by several processes or threads at the same time while only one process or thread can write in the data at any time. SQLite3 is a standalone program that can be used to manage the SQLite database.

### 2.4 AES Encryption

Based on the public’s growing awareness of privacy issues, people are now paying more attention to information security. The comment encryption algorithms include DES, AES, RSA, Base64, MD5, and SHA-1.

AES is an encryption standard adopted by the federal government of the United States as a replacement for DES. It has now become the mostly used encryption method in symmetric-key algorithms and is used world-wide. This algorithm was originally designed by Belgian cryptologists Joan Daemen and Vincent Rijmen.

AES has a fixed block size of 128 bits, and a key size of 128, 192 or 256 bits. The majority of the AES calculation is completed in a special finite field. There are four core steps in an AES encryption algorithm, which are SubBytes step, ShiftRows step, MixColumns step, and AddRoundKey step. In AES, an intermediate 4*Nb matrix, referred to as state and a 4*Nk matrix, referred to as a cipher key, are used to determine the rounds of calculation, referred to as Nr. Nb = the length of data/32 bits while Nk = the length of key/32 bits. The SubBytes step is a non-linear replacing calculation. The S-box is set up after two calculation processes, which are the calculation of the multiplicative inverse over GF(2^8) followed by an invertible affine transformation. In the ShiftRows step, the bytes in every row in the state matrix are shifted cyclically by a particular offset. Then the bytes in each column are combined together by using an invertible linear transformation in the MixColumn step. Finally, in the AddRoundKey step, the round key is added to the state using a bitwise XOR calculation. [13]

The AES encryption algorithm is suitable for many conditions as it is efficient, flexible, and strongly secure. The length of the key and the round times are changeable. There is no high requirement for internal memory, which means that it can achieve a satisfactory performance even in circumstances involving strict memory restriction.

### 2.5 User-friendly application guidelines

For an application, it is important to involve user-friendly features. The Android guidelines provide user friendly design principles for a developer to follow. Fundamentally, it states that an app should combine beauty, simplicity and purpose to create a good experience for the user. It ought to be easy to understand so that even if people are using it for the first time, it is possible to grasp the most important features and can enable them to gain access to superb technology with clarity and grace. When it comes down to details, there are also some basic principles to follow. Firstly, things should be kept as brief as possible and an attempt should be made to use pictures, icons in order to explain ideas. Secondly, make the style of components be consistent with the
theme and make them visually distinct to assist users in discerning their functions. Thirdly, the common application user interface consists of a main action bar, view control, content area and split action bar, which is a relatively transparent manner in which user can operate. As in the GUI part, the layouts should be flexible in order to suit different devices. [14] [15] [16]
3 Methodology

Choosing the correct methodology and methods is the first step and the key in relation to project procedure. For developing this application, the application development platform must be firstly determined and one suitable development model chosen to be followed. Object oriented programming and modularized programming are adopted in this project. After testing the application, it will be evaluated according to its user-friendliness and other principles discussed. The concerning technology and methods will be introduced in this chapter.

3.1 Selection of development platform

In relation to the popularity and availability of the doctor’s end application, a suitable developing and running platform should be chosen. Based on the years of efforts by our e-health group, the system has been successfully established on a PC platform. The new requirement is its establishment on mobile devices such as smart phones and tablet PCs. Currently, the most popular operating systems on mobiles devices are Android, iOS, and Windows phone. In this project, the Android platform has been chosen due to the preponderances described below.

A software stack is adopted in the Android platform with a Linux core providing the basic function and various programmer developing diverse applications. Openness, independence from a service provider, abundant hardware options, no developer restriction and supporting Google Play are the notable advantages of the Android platform over any of the other available options. Android allows any mobile end manufacturers to join the developing platform, which leads to the diversity of novel applications. This also gathers a wide user base for itself. The internet service is not restricted to the Android platform and the choice of hardware is also significantly plentiful. Manufacturers promote all sorts of products with different features and specialties but this will not affect the compatibility of software or the data synchronism. Choosing the Android platform minimizes the problems that might occur in the programming process, simplifies the developing procedure and lays a foundation for the future popularity of the doctor’s end application.

3.2 Selection of development model

Over the years, a large diversity of application development models such as agile methodology, waterfall methodology, extreme programming and rapid action development has been taking shape. While each model has its own advantages and disadvantages, based on the requirements, scale, and urgency of different mobile application development projects, particular models are able to be applied. For example, for agile methodology, the whole development procedure is divided into several parts. This approach makes it easy to alter the project and reduces the overall risk in relation to the whole project. The waterfall model is best suited to steady and static projects for which the main emphasis is in relation to schedule planning and for which there are no significant changes throughout the entire process. If the application requires a quick delivery, then the best choice will be rapid action development. [17]
This project is a function-oriented application development aimed to realize an Android application with concrete functions. Hence, the waterfall model is adopted in this project. The project starts with the related information gathering and background searching. Once the general goal has been settled upon, the move is then towards schedule planning after which the whole project course is completed step-by-step. This model diagram can be seen in Figure 3.1.

**Figure 3.1: Waterfall Methodology model diagram**

The procedure strictly follows the chosen development model, with the project process being based on the waterfall model as shown in Figure 3.2.

**Figure 3.2: Project procedure**

Before the design and implementation can be initiated, it is important to have knowledge of the background and the current situation for the E-health system. This information is gained by reading the previously reported paper regarding the E-health system in the group as well as studying the paperwork of other Electronic health systems. In Figure 3.3, the topology of the E-health system and the role played by the doctor’s end in the whole system is very clear.
To accomplish this role, the first aspect to be considered is the required functions. The user population of this application will be doctors, who require patients’ information and their health data. Without doubt, a great deal of effort is required to be put into the data interaction. The data from the patients does not come directly from the patient’s end. Firstly, the patients upload different data to the server, which will process and save the data. When the doctor logs into the application, it will send a requirement to the server for the patient’s data. The remainder of the process is in displaying the data in a user-friendly manner. However, it is not sufficient for the doctor’s end to merely possess this function. This application should also enable interaction to take place between the doctors and patients.

As the function of this application has now been clarified, the realization of the application is based on these functions.

### 3.3 Modularized function realization

Modularization is a common means of solving complicated problems as it breaks the system down to small easy manageable modules in order to reduce the complexity. In software/application development, modularization is used to divide the whole project into several smaller, interconnected component elements that are able to achieve specific detailed functions. A module is an alterable, dividable, combinable unit that possesses attributes such as interface, method, logic, and state.

Adopting modularization in this project assists in detailing the core goal into concrete targets. Additionally, the modularized programming process in developing the doctor’s end application also makes the sub-modules independent, enhances their extensity and modifiability.
The specific divided modules will be presented in the next chapter.

3.4 Methods for Achieving user-friendliness

During the whole process of application development, the user-friendly application guidelines have been strictly adhered to. In addition, consideration was given in advance to the functions that user will require and based on this, the methods for both the design and for user interface are determined. An attempt was also made to utilize some existing health care apps in order to determine some good features that could be used as example and to avoid features which had not proved to be successful. The efforts made were in relation to making this app have a transparent, well-formed user interface that is easy for the user to use.

3.5 Methods for Testing & Evaluation

The testing and evaluation is a very significant part in the application development. It is of vital importance to have knowledge regarding whether the app is of an acceptable standard for use and also be aware of its shortcomings. Based on the feedbacks obtained from the testing, further modifications can be made to improve the performance of the app. After implementing the app, it will be tested on both an Android device emulator and real Android devices. At the same time, members in the research group will be asked to use this app and to provide feedback based on their experience. The final evaluation is conducted according to their feedback and also by comparing it with the user-friendly application guidelines.
4 Implementation

Following the chosen development methodologies, the design and implementation of this app are conducted and the procedures will be presented in this chapter.

4.1 General Structure

The programming process of this application was divided into several parts and modularized programming was applied in order to accomplish the goals. The general structure of the app is shown in Figure 4.1. The whole application can be divided into several modules: middleware API, database operation module, database operation module, GUI module, login module, patient info module, health data module, contact info module and contact module.

![General application structure](image)

Figure 4.1: General application structure

4.1.1 Middleware API module

This API is provided by other developers in the E-health system project group, who have been responsible for the data transmission between the server and doctor’s end. This API enables the doctor to obtain the most recent health data, basic information and contact information of his or her patients as well as the ability to send feedback to them. This module is the beginning of the data flow in doctor’s end.
The API act as a data processor to interpret the data in string form to tables and columns in the local SQLite database. After data processor “translates” the data from the server, it will be stored for use at a later stage.

4.1.2 GUI module

The GUI module provides a user-friendly interface that enables users to experience different functions at the doctor’s end. In the Android application development, there are two means of achieving UI layout—using Java code and by an XML layout file with both possessing advantages and disadvantages. The Android SDK provides an internal UI frame so that it is possible to use an XML file to define the layout in the application. There are two major classes in Android SDK—android.view.View and android.view.ViewGroup. For a static graphic interface, the XML files are used to set the layout. This method realizes the View part in the MVC (Model-View-Control) development; separates the graphic interface from the logic and the other function-achieving interface. In XML file, a tree structure describes the layout of the different components. These XML files are parsed by Java and in the action controlling interfaces and when the onCreate() method is called, these layouts will be displayed.

The other method in the GUI module is using Java code to directly generate the layout, but this will require a large amount of coding, and is difficult to maintain. However, in relation to dynamic layout, using Java code is a better choice as it has the ability to respond and be readily altered according to different requirements. For example, as can be seen in Figure 4.2, in the schedule page of the doctor’s end, the calendar view is generated by Java code due to the changeable dates of months in different years. Three classes are created to define the calendar cell style and the headline style.

Figure 4.2: Calendar view generated by java coding

The discussion above is the traditional means of determining a layout in Java, but there is a tree structure in layout XML file, which is also a set of component
elements. Using an XML file to define the layout is suitable for a static layout where the contents do not significantly change during the whole activity life circle.

### 4.1.3 Login module

This module is divided into 3 parts. The 1st part involves data gathering. Certain elements on the Login page are set to gather the username and password information that is input by the user. The first several letters of the username which is input will be compared with the data map in the SharedPreferences and if they match the already existing data, the application will call the auto-complete function to complete the input box. If this is not the case, the data will be stored as an object variable. The 2nd step is the confirmation of information. These data will be encapsulated as a data object and sent to the server for checking as to whether or not they are a match. Depending on the results, the server sends back a 0 (stands for not matching) or 1 (stands for matching) to decides whether the user is able to log in. The logging status information will thus be displayed. If the application enables access to the main page, then it will ask the server again to load the message and the data that the patients have uploaded. The 3rd part is data storing. If the user chooses to remember his or her username and password, then these data will be saved as SharedPreferences in Hash map form as well as a data object in the SQLite database. Then the next time the user wants to log in, it will not be necessary to input username and password again.

### 4.1.4 Data modules

These modules play a significant role in the data flow throughout the whole application. In the health data module, several different types of health data are defined due to a large diversity of health data types. Different data can have different forms, representing ways and measurement units. A HealthData object is created for data interaction. The methods are encapsulated in the HealthDataDBFunction class. There is a data_type table in the local database for mapping the health data with their own type. During transmission, the type of health data is represented by type_id. Method MatchingDataType() is called before displaying the data. This method ensures that the data is presented in the correct form and with the correct measurement units. In the patient information module and contact information module, data objects such as patientInfoData ContactInfoData are created for carrying different information data.

The data module is the core in the interaction between the database operation module and other functional modules. The tables, columns, and records in the SQLite database are transformed by this module into data objects so that the methods in other modules are able to make use of them.

### 4.1.5 Contact module

This module is designed for the doctor to contact their patients or colleagues. The approach can involve sending emails or making phone calls. It requires a basic service on the device. Firstly, a demonstration in Manifest.xml file must be made to obtain permission for using the calling and mailing service on the Android device. After that, the intent-filter attribute should be set. The contact
information is gathered in the contact information module, and then the intent.ACTION_CALL and intent.ACTION_SEND are used to achieve contacting.

The contact data is divided into two contact lists. One is for the patients of the user and the other is for the user’s doctor colleagues. When the user wants to obtain patients’ contact information, the application will send an ask to the server to load the contact list and store them in the local SQLite database. The contact information object will contain the patient’s name, age, gender, disease type, email address and phone number.

4.1.6 Database access and operation module

These two modules together are able to achieve the access to the local SQLite database on the Android device and achieve a variety of data operations. In the database access module, a DatabaseHelper class is created for the connection. It is an SQLite database object in which there is a method getWritableDatabase() which is defined.

In the database operation module, it makes good use of the various methods that are provided by the SQLite database, such as execSQL(), rawQuery(). These methods allow for both a database query and operation on the basis of an SQL statement. Some encapsulated methods such as insert(), update(), delete() also make the operation simpler. By combining the use of the pointer cursor and move(), the tables and columns as well as a data record can be created, updated, read and deleted. According to the functions of the doctor’s end application, it is necessary to read a patient’s health data, contact information and schedule data from the database and to insert new contact information data and schedule an event into the database. Hence, data objects and the corresponding DBFunction classes containing the required methods such as getContactList(), getPatientInfo(), addEvent(), addNewContact() are created. The core of these methods is the use of execSQL() and rawQuery() for the purpose of performing reading, creating, updating, and deleting data.

4.2 Main functions

The doctor’s end application will contain the function for a user to log in, view patients’ newly uploaded data and messages, obtain patients’ and other doctors’ contact information, add new contacts, provide feedback/advice to patients, contact patients, manage a schedule and so on. The basic functions are shown below in Figure 4.3.
4.2.1 Log in
It will require users, that is, doctors to input a unique username and password that they have registered in the e-health system in order to log in. This function guarantees the safety of patients’ data. However, when considering convenience, a user can choose the function “remember password” and thus on any other occasion that the user intends to log in, the username and password will be auto-completed based on the first letter of the username previously input by the user.

4.2.2 Getting patient’s data
The data that the user is able to obtain involves the patients’ health data, patients’ messages, and contact information. Different data will be displayed in the correct forms on different pages, such as a health data page, message page and contact list page. The display of the data should be user-friendly. For example, health data is displayed according to the patient’s id and the message data is ordered by time from the most recent to the most distant. A colleague contact list is separated from the patient contact list. The ability to obtain data forms a significant part in relation to the doctor’s end as the assistance that can be offered by a doctor to his or her patients is based on this information.

4.2.3 Giving feedbacks/advises
The doctor’s end application is able to provide bidirectional data communication between doctors and patients. This function allows doctors to leave their professional opinions after analysing the data and the user is able to comment on the health data and reply to the messages. After this, the feedback will be sent to the server and the patients are then able to obtain this information.

4.2.4 Contact operations
A user will be able to obtain patients’ contact information and their colleagues’ contact information including their email addresses, phone numbers and other
basic information. Contact information can be added, deleted and updated from the doctor’s end and the user is able to call and send email/SMS to his or her contacts. The instant communication will contribute to providing assistance during any emergency situations.

### 4.2.5 Schedule management

The users are able to set and check their arrangements and schedules. Everything will be displayed in a calendar view with certain events shown on a specific date below the view. For users, this function assists them to better manage medical health care arrangements.

### 4.2.6 User settings management

Users can change their basic settings such as username, password and their basic information on the settings page.

The function table in the doctor’s end is shown below in Figure 4.4.

<table>
<thead>
<tr>
<th>INPUT</th>
<th>ACTION</th>
<th>DATA</th>
<th>DATA CARRIER</th>
<th>OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>User login</td>
<td>User info</td>
<td>Call CheckUserInfo()</td>
<td>Username &amp; password</td>
<td>SharedPreferences</td>
</tr>
<tr>
<td>View patient or colleague doctor's disease/contact data</td>
<td>Click relevant buttons</td>
<td>Call GetPatientData() or GetFriendData()</td>
<td>Contact info data</td>
<td>SQLite</td>
</tr>
<tr>
<td>Add contact data</td>
<td>Click “Add” button</td>
<td>Call AddContact()</td>
<td>Contact info data</td>
<td>SQLite</td>
</tr>
<tr>
<td>Giving feedback to patient</td>
<td>Doctor's feedback or advise</td>
<td></td>
<td></td>
<td>SQLite</td>
</tr>
<tr>
<td>Call/send email</td>
<td>Click relevant buttons</td>
<td>Activate device internal call/sending mail service function</td>
<td>Email address</td>
<td>SQLite</td>
</tr>
<tr>
<td>Check schedule</td>
<td>Clicking on event date</td>
<td>Call CheckSchedule()</td>
<td>Event info from database</td>
<td>SQLite</td>
</tr>
<tr>
<td>Add schedule event</td>
<td>New schedule content</td>
<td>Call AddEvent()</td>
<td>Event info from database</td>
<td>SQLite</td>
</tr>
<tr>
<td>Change user settings</td>
<td>New setting info</td>
<td>Call ChangeSetting()</td>
<td>Username of password</td>
<td>SQLite</td>
</tr>
</tbody>
</table>

Figure 4.4: Function table in doctor’s end

### 4.3 Data flow & Data storage

Data plays an important role in the doctor’s end application. It contains the patients’ information and is critical in relation to the patient—doctor information exchange. The app will perform “getting data—storing data—reading data” for a user. The data flow and data storage process is described in the following sections.
4.3.1 Data Flow

As shown in Figure 4.5, the data displayed on the doctor’s end is being received from the server. Data flow between the doctor’s end and the server can be divided into 3 parts. The 1st part is the end side asking the server for data. The 2nd part is the data being transmitted from the server to the doctor’s end. The 3rd part is the data storage in the doctor’s end and the display of the data.

The server sends data as strings and, for example, if the server intends to send a string that contains 10 pieces of patient’s health data, the form will be similar to data1###data2###data3###...###data10. Sending data in this way saves the cost of data transmission. In addition, after receiving this string, the data processor on the doctor’s end will be aware that the symbol “###” actually means the division of different pieces of data. Then the processor will extract the proper data and store them into the local database SQLite. By using a database operation, the program obtains data from the local database and displays them by means of the GUI.

In the doctor’s end application, several data managing packages are created for dealing with the inflowing data. As shown in Figure 4.6, different classes are created for managing different data objects. According to the user’s command or input, certain corresponding methods will be called to respond.
4.3.2 Data Storage

As for the doctor’s end application, a great deal of the patients’ data, such as their basic information and newly uploaded data about their diseases will be transported from the server to mobile devices. This requires a local database to store all the data. The internal memory is limited in relation to mobile devices and thus a functional database that is required that has a low resource occupancy but has a high operating speed. SQLite is a perfect choice to satisfy all the above requirements. SQLite is a lightweight database, which is operational within embedded systems. The cost of the resource is very low—only several hundred Kilobytes in comparison to MySQL, PostgreSQL or some other generally used databases and thus SQLite is more suitable for the development of Android. Additionally it supports the majority of the SQL statements.

In terms of the doctor’s end application, there will be a database called ehealth.db, which contains the tables of tb_contacts_patient, tb_contacts_doctor, tb_schedule, tb_message, tb_patient_data, tb_datatype, as shown in Figure 4.7. The table structure is shown in Figure 4.8.
Android application of Doctor’s end in E-health system

Lingzhen Chen
2013-07-13

Figure 4.7 Local database for doctor’s end

Figure 4.8 ER/UML model of local database

The tables and columns in this database are shown below in Figure 4.9.
During Android programming, it is sometimes necessary to store private primitive data such as configuration information. In this case, it will be a waste of resources to save them in a database. SharedPreferences provides an excellent solution to this problem. [18] SharedPreferences supports the storage of data types such as Boolean, int, float, long and string. The data is mainly saved as key-values. A SharedPreferences file is saved under the corresponding private folder of the application as an XML file. These data can only be read by the application that has created them.

In this application, the username and password that the user provides to log in are saved as SharedPreferences.

### 4.4 Development Environment Setting

Developing an Android application requires several computer environment set-ups. Simply stated, Eclipse, JDK (Java development kit), Android SDK (software development kit) are the necessary components. Eclipse is used as the main programming software in this project. JDK, including JRE (Java Runtime Environment), is installed as well as Android SDK to set up a proper developing environment. Android SDK stands for Android software development kit, which is one of the very important tools for developing.
programs on the Android platform. After the set-up, it is possible to choose to create an AVD (Android virtual device). This tool allows the program to run on the computer instead of a real Android phone, which makes the test of application during programming much easier.

The working principle of the AVD is that the program is being compiled as an APK (application package file) and read by a virtual device. APK is the kind of file that can be read by the Android system. The whole process is shown as below in Figure 4.10.

![Figure 4.10: the process of building apk file](image-url)
4.5 Interaction among activities

Referring to Figure 4.11 above, after inputting a valid username and password, the checking function will be called to determine the access permission. After the user gains access to the home page, there are 4 basic activities in the doctor’s end application, which are the index page activity, contacts activity, schedule activity and settings activity. The entrances to these four activities are controlled by four buttons on Viewflipper and they contain the gateways to a different function page.
Android application of Doctor’s end in E-health system

Lingzhen Chen

2013-07-13

Figure 4.12 Interconnected data displaying activities

Refer to Figure 4.12, from the home page, the user can access the patient's list page in order to view the patient's health data. Patient list activity is called when the user presses the “My patients” button. Then, the particular patient id, which is stored in the SQLite database as p_id, will be passed to the health data activity. This activity displays the chosen patient’s newly uploaded health data by querying SQLite with a p_id. Messages from the patients can be seen by clicking “My message” button to enter the message activity. It is also the case that these activities regarding the showing of patient lists, patient health data, basic information, and contact information are actually interconnected and thus the user can jump from one to another according to his/her needs.
Figure 4.13 Contact operations

Referring to Figure 4.13, at the same time as the user views the patients’ or their colleagues’ contact information, they can choose to activate the calling and sending mail activity, which will lead to the dialing board or mail sending page. Calling and sending activities use the inbuilt device actions. The user is also able to alter the contact information. On the operation menu, by clicking “Delete” or “Edit”, the corresponding activities will be called. Deleting an activity does not provide a graphic interface but works backstage. For editing, the patient or doctor id will be passed to the editing activity. The new information which has been input will be stored as a contact data object in the SQLite database. The data update is thus realized.
Figure 4.14 Schedule management

Referring to Figure 4.14, below the calendar view of the schedule, the user is able to see the arrangements for a certain date. When the user clicks “Add event” button, the adding event activity will be called and a string for the chosen date will be passed to that page. After the user has input the information, it will be passed back to the schedule page again.
5 Evaluation

After the implementation of the entire doctor’s end app, it is time for the testing and evaluation part. The app was tested on an Android device emulator and real Android 2.3, 4.0, 4.1 devices to determine the performance. In addition, 5 members in our research group were allowed to use this app and provide feedback. The features of our app were compared with the guidelines on the Android developer’s official website in order to conduct the evaluation.

5.1 User-friendly

Since the application has been developed so that people are able to experience the convenience and usefulness associated with that technology, user friendliness is one of the key evaluation criteria for distinguishing between good and bad apps. For users, the user interface represents the software itself and it is on this that the user experience largely depends. During the procedure of this project, UI design has been the primary consideration and particularly graphic user interface design. Research into other existing health care concerned apps was conducted before the implementation of the doctor’s end app and with reference to their graphic interface, a clear picture has emerged in relation to the users’ requirements. The UI design has been conducted bearing in mind common user behaviour and also from the perspective of the doctors.

The GUI of the doctor’s end application is both simple and transparent. The style, colour, font, icons, buttons of each activity page are consistent. The colour contrast ratio is sufficient so that the user will not miss any content on the page. In relation to the understanding of the function of buttons for the user, icons and text are used. In order to display health data, a ListView is used to separate each piece of data by means of single lines. When there is both date and time information, this is always placed at the lower right corner of the data information. In order to match the different resolutions on different mobile devices, each image used in the application is stored in high-definition form, middle-definition form and low definition form.

In terms of the functions, the doctor’s end provides a calendar view for the users to record and check their daily medical arrangements. The use of this view makes the date and events more transparent than using plain text. It is separated from the calendar that is built within the device itself, so the user would not be confused about the arrangements of their personal life and professional life. The contact list function provided in the app also achieves the same goal.

5.2 Performance

As the user requires retrieving a great deal of data from the server, the data transmission speed is rather an important factor within the whole application performance. Taking the hardware performance of different devices into consideration, this application has been tested on Android device emulator, 2.3,
4.0, 4.1 devices and the response time for receiving data was determined to be no more than 2 seconds.

During the application development, great importance has been attached to memory reclamation in order to improve performance. The choice was to use more local variables rather than static variables because they are stored in a stack, which can be retrieved more quickly. Only the necessary Java objects are created so as to avoid waste in relation to system reclamation. In addition, the often used objects are put into the buffer memory. In relation to loops, reduplicated calculations of the variables have been reduced. In the final part, used resources are freed. An asynchronous data asking and receiving mechanism is adopted in order to reduce the latency time. Compressed images are used to save internal memory. As shown in Figure 5.1, the application has been tested on several devices and there is no obvious difference in relation to the graphic user interface performance.

![Android 4.1](image1.png) ![Android 4.0](image2.png) ![Android Device Emulator](image3.png)

Figure 5.1 GUI performances on different devices

### 5.3 Privacy preservation

In the medical care field, privacy issues have always been given great importance and thus this aspect must be given significant effort. For logging in to the application, every user requires a unique username and password. This method ensures that every doctor can only obtain the basic information and health data from his or her own patients. During the data transmission between the server and the doctor’s end, the information is preserved based on the privacy rules. For local data storage, the information is encrypted by AES. AES encryption is realized by adding a round key, sub-bytes, shifting rows, mixing columns. Even if the users have lost their phones, the patients’ information that has been stored inside the phone will not be easily revealed due to the encryption.
6 Conclusions

In order to further improve the function of the electronic health system that our research group has been working on, a doctor’s end application on an Android platform has been proposed and implemented in this project. The procedure follows the application design principles and methodology. It provides a user-friendly application that the doctor can use to retrieve a patient’s health data and provides the ability to offer both feedback and advice. The basic functions also include contacting patients, setting down and checking a medical arrangement schedule, and changing a user's basic settings. This application has been designed and implemented using eclipse with ADT plug-ins to program. To evaluate the whole project, the application was tested on an Android device emulator and several other mobile devices. The results are in accordance with expectations. The overall performance of the app is satisfactory. There are no display errors and little response time latency.

This app reduces the complexity and inconvenience between doctor-patient information exchanges. With the popularity of mobile end devices and the facility to have internet access, data transmission can be nearly real-time. This can significantly contribute to patient disease situation control and emergency response. The app brings the doctor—patient relationship even closer. However there are some issues that must be given attention to. In the medical care field, patients’ privacy is given significant importance and this involves many ethical issues. This application concerns a large amount of sensitive patient health data that must be preserved at all costs. During the development of this app, the data was encrypted by means of AES methods during the storage process. However, it is still not possible, even when using these methods, to completely ensure the security of the data. Thus, in the future, greater efforts will be required in relation to patients’ privacy preservation as well as the function and UI modification of the app.
References


[6] Xin Zhang, Tingting Zhang, “Distributed Electronic Health Record System based on Middleware” Department of Information Technology and Media, Mid Sweden University, March 2013


http://en.wikipedia.org/wiki.SQLite

http://en.wikipedia.org/wiki/Advanced_Encryption_Standard

[14] Android Developers, ”Creative vision”,  
Retrieved 2013-06-07

http://developer.android.com/intl/ja/design/get-started/principles.html
Retrieved 2013-06-07

[16] Android Developers, ”UI Overview”,  
Retrieved 2013-06-07

http://www.androidmobiledevelopment.com/methodology.html

http://developer.android.com/intl/ja/reference/android/content/SharedPreferences.html
Retrieve: 2013-04-16