Android Application Project report

Project Name: ClassWorld

Group Members:

Li Xinhui

Chen Jiawei

Wang Keyu

Zhu Yuda

June 22, 2013

CONTENTS

1. INTRODUCTION

1.1. Motivation

Social network applications are pervasive in our daily life. We always find different kinds of social network applications in everyone's smart phone. Facebook and twitter connect people together and Weibo and RenRen are also very popular in China. However, a significant issue has become in front of us, what is the trend of future social network. Some newly developed social network sites have given us some inspirations.

Several years ago, popular social network models are only those sites which target on a large scale of people covering all the aspect of our life. But applications like Pinterest and Instagram are both focus on the photo sharing, Momo is used to find friends based on localization. From these applications, we can find that there are still a lot of separation markets in social network field. This kind of vertical separation marketing conception is also applied to the setup of company-based social network in recent years, such as Yammar. It connects the people in a same company, the information about the work are sharing inside of their own Yammar circle. This kind of idea exactly inspires us to build a social network application to combine the students in a same school or specifically connecting students in a same class.

Class is just a type of social network; a lot of class related affairs or information we want to solve and share together. Hence, we build a class-based social network application ClassWorld to implement these kinds of issues.

Besides different target people and social circle, another characteristic of our application is the data we collect through this app from the perspective of this application provider. In big data eras, social network data has attracts more and more attention of the researchers and billions of data has been collected every minutes and analysis by the SNS provider, like Facebook, but whether all of this kind of data is valuable as we thought. The user architecture is difficult to separate and analyze about the results for specific aim. So in our application we build up an HTTP server based on java programming to collect all the related information in our own application. What we can do is to use this kind of data to predict more accurate social network architecture for the service provider, because we focus on a more specific structure and we have more targeted data.

1.2. Contributions

ClassWorld has following main contributions:

1. Achieve a new topology-based social network architecture to connect people.
2. Propose a new concept of method for service provider to analyze the social data

for specific aim.

3. This application can satisfy different students' various requirement, such as check attendance notification and problem solving discussion.
4. A portable and scalable syllabus for each student is accomplished.

2. FUNCTIONAL DESIGN

2.1 Function overview

The carrier of this method is the class syllabus; every user can modify and check their syllabus. The remote database can collect all this information to build the social circle based on the class registration. And a student can check his/her classmate list based on the analysis result given back from the remote database.

After getting the classmates list, connection can be set up either by offline notification or online chatting. Each student also can choose the classmate to communicate with.

2.2 Function architecture

2.2.1 User Function Description

In student part, the basic action he or she can take is like Fig 2.1. They are all focus on how to manage this own syllabus, which is stored in each people's local database SQLite (introduced later).

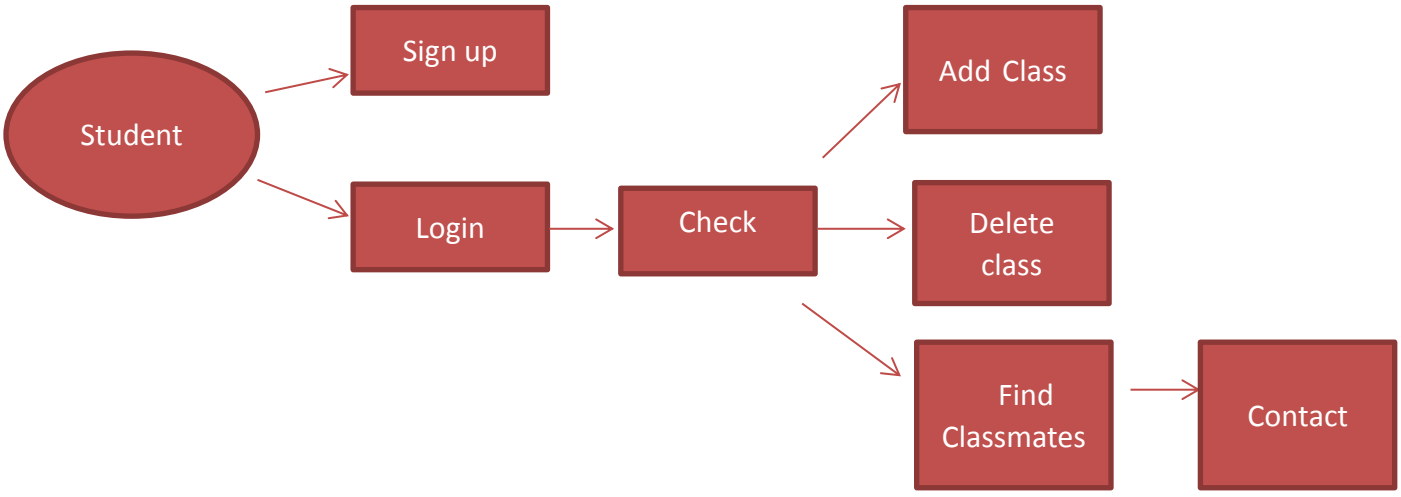

Figure 2.1 Student action

2.2.2 Database Function Description

In the remote database part, the basic action is to collect the information about all the students' class registration information and map this information with student register ID. Based on this relationship, it can determine the class circle, which should be used when user want to get his or her classmate list. The function can also be illustrated in Fig 2.2

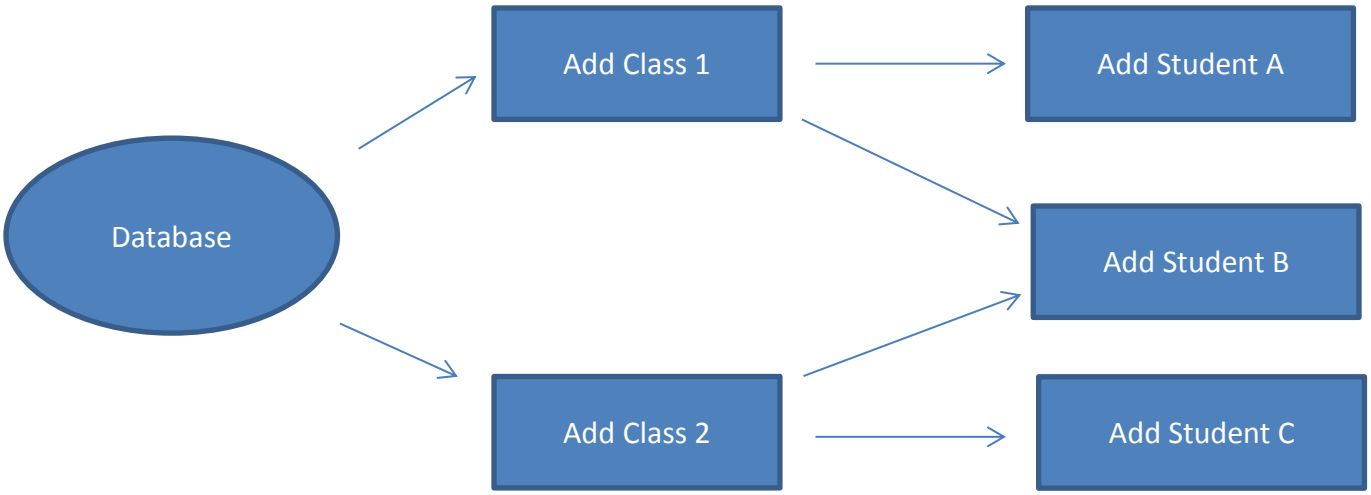

Figure 2.2 Database action

2.2.3 Connection Server Function Description

In the connection part, the basic action is to set up connection between students either by offline notification or online chatting. As soon as a student gets his/her classmate list, any classmate can be chose to communicate with. The function can also be illustrated in the following figure.

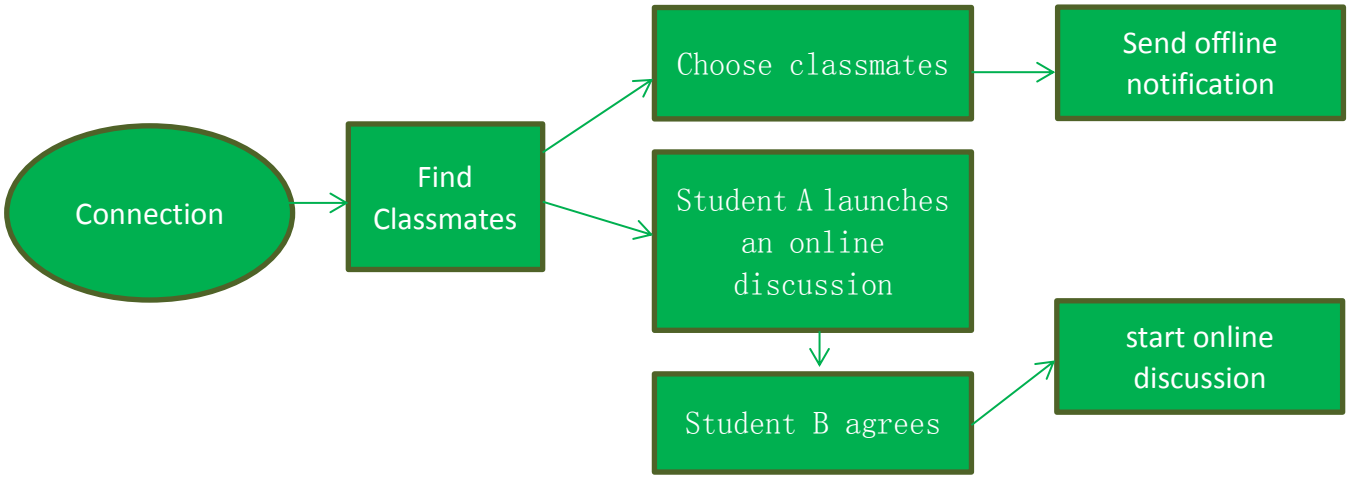

Figure 2.3 Connection action

3. TECHNICAL DESIGN: ARCHITECTURE AND IMPLEMENT

3.1. System Architecture

Figure 3.1 shows the architecture of the system.

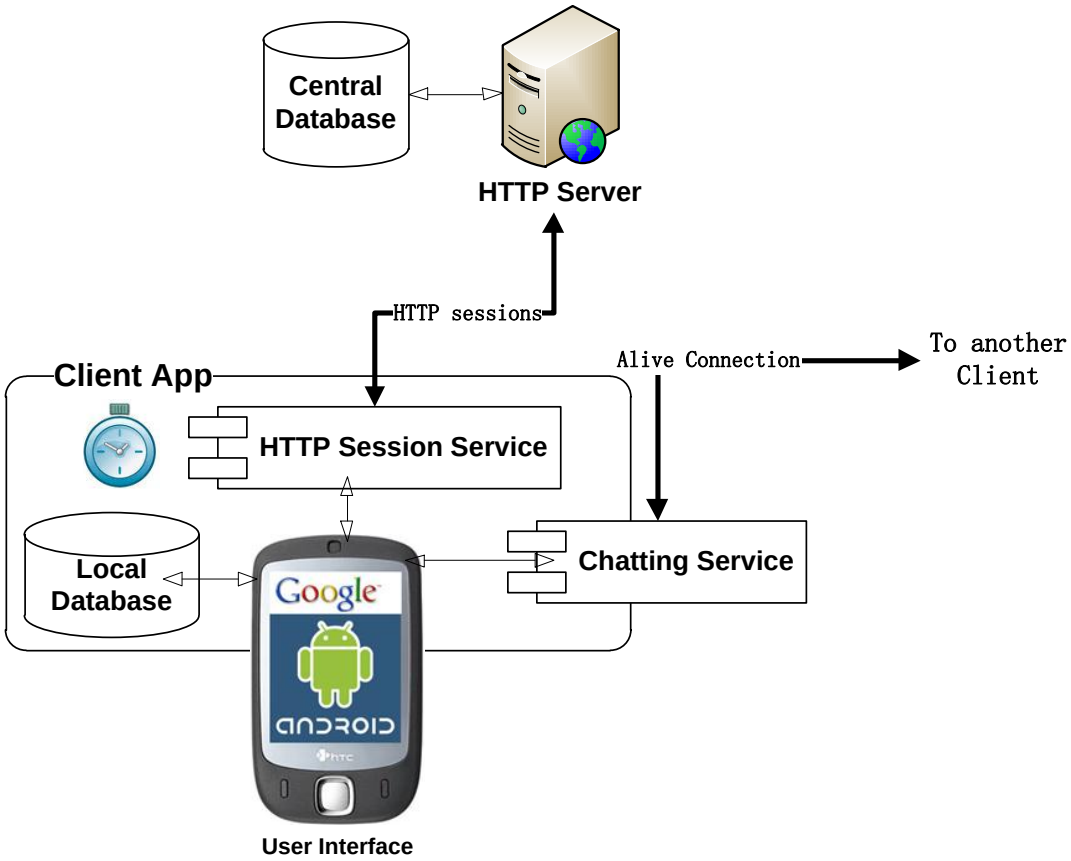

Figure 3.1 Architecture of the system

We divided our development into two parts: the Remote Server, including the Central Database and the HTTP Server, locates in a computer as the server (actually it was a laptop in our test bench); and the Client App, running on Android smart phones.

3.2. Central Database

We build our central database based on a relational database management system, MySQL. In the database, we create three tables, "Accounts", "Courses" and "Ownerships". The "Accounts" stores information of each registered user (student) such as usernames, passwords and so on. The "Courses" stores the user-defined courses and their details like their instructor's name, classroom, etc. The "Ownerships" stores what courses each student owns.

In the following section, you will see our server is written in Java. Java language has provided us with language-specific APIs include libraries for accessing MySQL

databases called JDBC.

3.3. HTTP Server and Client

3.3.1. Introduction

We use HTTP protocol to exchange most data between every user and the central server. A user posts requests or data to the server, and then the server processes the data, maybe querying from or updating the database, and finally transmits the result of the operation. Obviously, we should implement the communication part of both the server and the clients.

3.3.2. Common Part

To exchange the data, we define data structures of Accounts, Courses and Ownerships as well as the encapsulation method when transferred on the Internet.

3.3.3. The Server

The structure of the server is shown in the following figure.

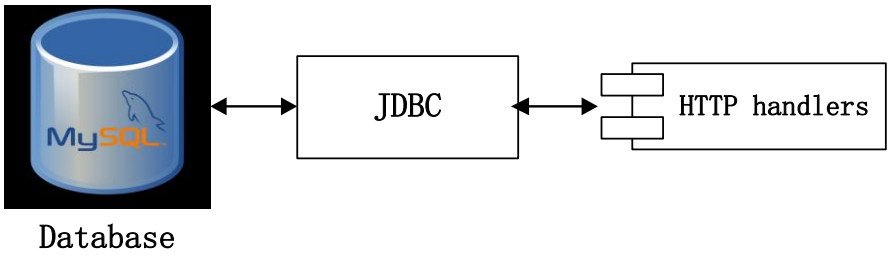

Figure 3.2 the structure of the HTTP Server

We wrote the server program in Java. As the server program launches, it connects to the database through JDBC.

Then the different request handlers are loaded into the memory. We wrote these handlers to process 12 kinds of requests from the client, such as logging, querying a curriculum schedule and so on. The program starts to listen to the HTTP port for coming request. This is done by calling API com.sun.net.httpserver.HttpServer.

As a request comes, the HttpServer API will classify the type of request and deliver it to our handlers. For example, if a client requests a “new course” operation, the request will be delivered to a handler called “NewCourseHandler()”, which will query the database to see if the course exists and update or add an entry, and then return the client a response.

Chinese characters confused us at one time. At the beginning, we found once Chinese characters stored into MySQL database, it become gibberish as fetched again. We later found it was due to two reasons. First, the MySQL, originally designed for

pure English databases, arbitrarily modify the encoding of data. Secondly, strings will also be tampered if we does not use an appreciate code page. We force MySQL not to change the binary code of all characters and set encoding in Java as “GB15000” to solve the problem.

3.3.4. The Client

The HTTP client, embedded in Client Apps, is much simpler than the server. When the client needs to communicate with server, it just encapsulates the request and data, and posts them to HTTP server (the posting can be done by calling API “java.net.URLConnection”). Then the program just waits for and processes response.

To enhance user experience, the client App starts another thread to do the communication in the background. The user interface is not fully freeze when the client is interacting with the server.

3.4. Local Database – SQLite

3.4.1. Introduction:

SQLite is a light weight database and because its portability and high efficiency in operation and solidarity, it is always embedded in the mobile operation system like Android and iOS.

We use this database as a local database and store the information of current user. It is more convenient for the user to change and store the content information in its application and also help the user to synchronize with the remote database, which can be wrapped as a package of data to transmit to the remote database.

3.4.2. Technology review

This database is sharing the process resource with the application; it has no separated process space. In this way, it may not seem as an RDBMS (Relational Database Management System). But it is has its own database engine, and it can manage all the command, such as create, delete etc., which are same as the RDBMS. Therefore, it can be looked as an integrated database from the perspective of application itself.

3.4.3. Implementation method

In android programming, we use a database Helper class “DBHelper”, to manage all the command set by user. Its method “execSQL” can execute all the database standard command to manage this local database. Hence, the problem becomes the

same as dealing with database management problem. The basic implementations are illustrate in Figure 3.3.

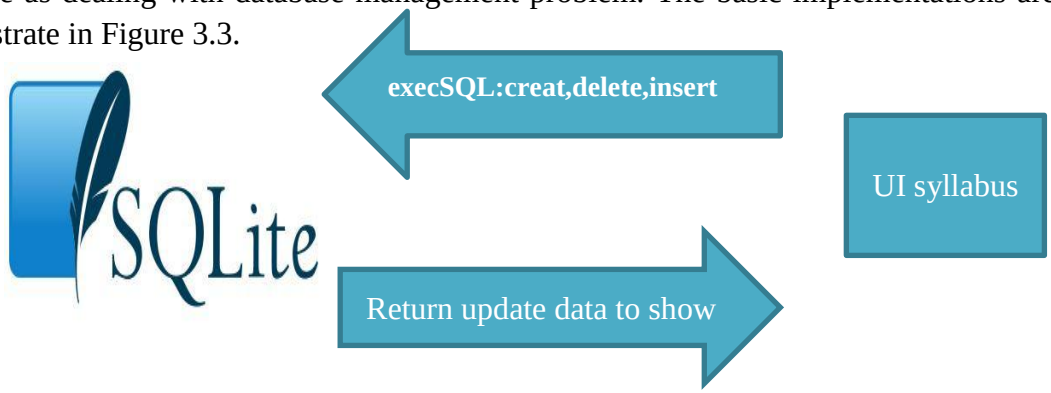

Figure 3.3 Local database management

3.5. Offline Notification

"Offline" has two meanings: First, if user A has sent user B a notification when B is not online, B can read the notification as soon as he/she logins in next time; second, the message may have some latency (1 minute or so) even if both user are both online.

When a user sends an offline message, the client posts the message to the server through HTTP protocol. The server puts the message in the message queue.

Actually, as a client logs in, it start a service in the background which polls every one minute the server whether there are new messages. Of course, if there are several new messages, the server will push all the messages to the client.

Obviously, the latency can be reduced if we shorten the interval between two polling. However, too frequent polling may lead to too much flow and network congestion. If we need to provide a platform for online discussion, we need to use another different technology which is illustrated in the next section.

3.6. Online discussion

3.6.1. Launching a discussion

Literally, only users online can launch an online discussion with each other.

The server program temporarily stores IP addresses of all online users. If user A tries to launch a discussion with B, the following session will occur.

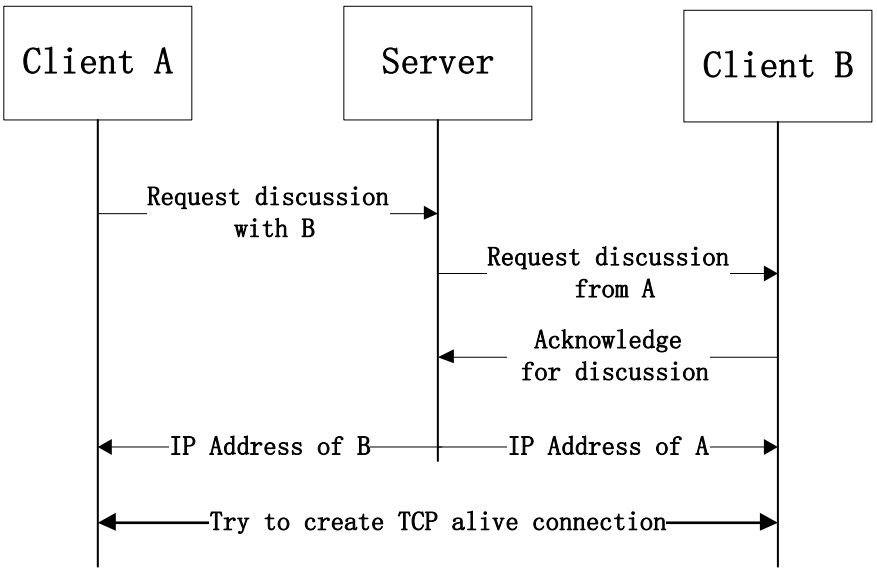

Figure 3.4 procedure to launching an online discussion

When the direct connection is created, user A and user B can begin to discuss with each other. Notice the connection will keep alive until the discussion terminates.

3.6.2. Why keep the connection alive

In most applications, TCP connections will be established for every request and terminated soon after the response finishes. This is appropriate for server-clients model because the server should reduce the number of simultaneous connections. However, in our application, the connection for discussion is direct between clients.

In our application, we use keep-alive connection for online discussion. The connection is established when a discussion launches, and terminated when the discussion finishes. Keeping the connection alive can have at least the following advantages:

1. Lower CPU and memory consumption because it don't need to create and terminate connection frequently.
2. Reduced network congestion because the amount of TCP requests is decreased
3. Faster response because it don't need to repeatedly handshake to create/terminate a connection

Maintaining a keep-alive connection is not just open a connection and keep it aside. The clients should periodically (of course, not too frequently) send message to each other to ensure the connection is still alive. We call this kind of message "heartbeat". If the heartbeat disappears, our application will realize the connection is dead (often because of change of network environment) and try to establish a new connection with the help of server.

4. SOFTWARE MANUALS

4.1 Login and Sign up

Firstly, The user have to install ClassWorld on his Android smart phone and have access to the Internet.

After opening the application, there is an entrance interface, See the Fig 4.1:

Figure 4.1 Entrance Interface **Figure 4.2 Sign up Activity**

The entrance interface has two operations: log in and sign up. Click the sign up button, the user interface jump to sign up activity. See the Fig 4.2:

In the Sign up activity, the user need to input his user name ID and input his password twice to make sure the password is correct. After successful register, user will come back to the Log in interface shown in Fig 4.3 . Meanwhile the user information has been created in remote database.

Figure 4.3 Sign up success

4.2 Class Management

If you already have an ID or sign up successfully, input your ID and password and click the Log in button, entering the class table. The software get the the user's class information from server database. See the class table in Fig 4.4:

Figure 4.4 Class table **Figure 4.5 Edit class**

The class table have three elements: class name, classroom and the teacher name. On the top of class table are five tabs: Monday, Tuesday, Wednesday, Thursday and Friday. Click the class name, you can have two operations: Edit the class or manage classmates.

User can edit class name, classroom and the teacher name, or delete the class completely. Click the classmates list tab on the top, entering the classmates list, The software will get the classmates list from the server database, see the figure below:

Figure 4.6 Classmates management

4.3 Classmates management

After entering the classmates list, The user can see the interface below:

Figure 4.7 Classmates management

The classmates management have two functions: Online Chatting and Offline Notification.

Online Chatting: online chatting is an important function for all social network applications, the ClassWorld has this function too. They can keep two people chat all the time with little delay. But both of the people should be access to the Internet

Figure 4.8 Online Chatting

Figure 4.9 Offline Notification

Offline Notification: Another important feature of ClassWorld feature is Offline notification. Any android smart phone which has the ClassWorld application can get notification like Fig 4.9

5. APPLICATION LIMITATION

Because of the time limitation, there are still some drawbacks of our application.

1. Passive selected: all the user may be selected as the communication target passively. If there are some message or people some student do not want to receive or contact, they should have the ability to limit such kind of bad communication. So the black list may be a better way to solve this problem
2. Checking online: people may need to know whether other people are online or offline, which may help them choose the communication method. But in our application, the classmates list is just given by remote database based on the registration. The better way is to not only show the classmate, but also show the current state of these classmate.

6. EXTENSION

Although our application is merely based on the class circle, it can be easily implanted to another specific scenario and target its own small social network members. An example can illustrate this implantation method. If the class schedule is changed to a schedule of international conference proceeding and all the participants can install this application. Since all the professors or students want to look at which session and its topic are interesting to them and the presentation will be held in which room, they will always open this application and see this electronic version schedule. Meanwhile, the function of the online chatting and offline notification function is always works out in this scenario. Because you may miss some important talk or you cannot attend two important sessions at the same time but in the different rooms, these two communication method help them solve this problem. Your friends may notify you some presentations they thought you may be interested in, and he will notify you in this case. And also if you cannot attend at once, the online chatting can help you get the important information from your friend in time.

All of these communicate log can be collected by the service provider's server. And he can use this kind of data to analyze the structure of social connection relationship among all these scholars. Based on this model some other function can also be realized such as recommendation. And the data in this model is more accurate than the data you collect by analyze the content in each professor's homepage.

Hence from this example, we can easily find that our application has a broad improvement space, but as the limited time we can only accomplish the class schedule application design.

7. GROUP MEMBER CONTRIBUTION

Name	Contribution
Xinhui Li	Remote database, Http Server coding, whole system test, writing report
Jiawei Chen	Local database, UI design and coding, test, writing report, making ppt for presentation
Keyu Wang	UI design and coding, test, writing report
Yuda Zhu	Writing report, test

8. CODE FUNCTION AND AMOUNT DESCRIPTION

In this section, we will describe the functions and the code amount of every important java files in the project. The first table shows about the common code, the second shows about the server program (common code eliminated), and the second shows about the client application (common code eliminated).

The total amount of code to develop the system is **2859 lines** in java language.

Table 8.1 java file function of the client application

File name	Function	Code amount
ChatActivity.java	the interface for online discussion	604 lines
ChatMessageService.java	the background service for offline notification	90 lines
ClassList.java & MyCurrentClass.java	the interface for curriculum schedule	155 lines + 69 lines
ChatMsgViewAdapter.java	the intermediary between chatting message data structure and the interface	109 lines
CourseDetail.java	the interface for add/edit/delete a course	277 lines
DBHelper.java	functions initializing the local database	59 lines
FriendName.java & FriendTab.java	the interface for classmate list	191 lines + 46 lines
MainActivity.java	the entrance interface	265 lines
Regit.java	the interface for register a new account	73 lines
Common.java &	global variables	11 lines + 18 lines

MyApplication.java		
UrlSessionUtl.java	functions to manage HTTP sessions	53 lines
TimeTableRegister.java	functions to register a new account	17 lines
TimeTableClient.java	functions to communicate with the remote server	90 lines

Table 8.2 java file function of the server application

File name	Function	Code amount
SQLconnection.java	functions initializing the JDBC to remote database	53 lines
TimeTableServer.java	HTTP server program and the offline notification queue	518 lines
ChattingLauncher.java	functions to help clients set up online discussion	92 lines

Table 8.3 java file function of the common code

File name	Function	Code amount
Account.java	data structure and encapsulation method of user account	14 lines
ChatMsgPkg.java	data structure and encapsulation method of chatting message	19 lines
Course.java	data structure and encapsulation method of course	20 lines
Ownership.java	data structure and encapsulation method of ownership	16 lines