



Undergraduate Report

A WeChat Multi Thread Chat Application



	Course	Application of Java Programming
	Academic Advisor	Professor Weiming Lu
	Teaching Assistants	
	Name	Leming Shen
	ID Number	3180103654
	College	College of Computer Science and Technology
	Major	Software Engineering

December 31, 2020

Contents

1	Introduction to the Project.....	4
1.1	Background.....	4
1.2	Goal of Design.....	4
1.3	Development & Running Configuration Environment	4
1.3.1	Development Environment	4
1.3.2	Running Configuration Environment.....	4
2	Overall Design.....	5
2.1	Project Structure	5
2.2	Class Design	6
2.3	Flowchart.....	8
3	Detailed Design.....	9
3.1	Communication	9
3.1.1	Client.java	9
3.1.2	Client_Reader.java.....	9
3.1.3	Server.java	10
3.1.4	Server_Handler.java	10
3.2	GUI.....	11
3.2.1	Chat_Window.java	11
3.2.2	Friend_List.java.....	11
3.2.3	Login.java.....	12
3.2.4	Register.java	12
3.2.5	Session_Client.java	12
3.2.6	Session_Server.java	13
3.3	JDBC.....	13
3.3.1	Database.java	13
3.3.2	SendEmail.java.....	14
4	Testing & Running	14
4.1	Login	14
4.2	Register.....	15
4.3	Friend List	18
4.4	Chat Window	19
5	Summary.....	23

6	References	23
7	Source Code.....	24
7.1	Communication	24
7.2	GUI	29
7.3	JDBC.....	43

1 Introduction to the Project

1.1 Background

In the course Java Application Technology, we are asked to write a program that implements a simple chat application. We need to use socket to implement communication between any two users. Java Swing library should be used to implement a simple chat interface.

Besides all this, the user's personal information and chat records are all stored in a database through the connection between Java and MySQL.

1.2 Goal of Design

1. A simple user interface to:
 - Sign in & Sign up (Using Email address and verify through email) to the system.
 - View the friend list.
 - Chat with friends with multiple windows.
2. A C/S socket model is developed to implement communication between users.
3. Link the program with the local database to:
 - Record the user's personal information
 - Record the user's chat message

1.3 Development & Running Configuration Environment

1.3.1 Development Environment

1. Operating System: Linux Ubuntu 20.04 LTS
2. Java version "1.8.0_251"
3. Java(TM) SE Runtime Environment (build 1.8.0_251-b08)
4. Java HotSpot(TM) Client VM (build 25.251-b08, mixed mode, sharing)
5. **JDK 15.0**
6. IDE: JetBrains IntelliJ IDEA 2020.2.3
7. Chrome Driver
8. Third-party Jar
 - activation.jar
 - javax.mail.jar
 - mail.jar
 - mysql-connector-java-8.0.16.jar

1.3.2 Running Configuration Environment

1. Make sure that you have already correctly installed MySQL and the MySQL-Server is normally running.
2. Make sure that you have already created a database named "wechat" and create all tables provided below:
 - User.sql

```
1. create table user
2. (
3.     user_id varchar(255) not null,
```

```

4.     password varchar(20) not null,
5.     user_name varchar(255) not null,
6.     name varchar(255),
7.     phone varchar(20),
8.
9.     primary key ( user_id )
10.);

```

- Friend_pair.sql

```

1. create table friend_pair
2. (
3.     user1_id varchar(255) not null,
4.     user2_id varchar(255) not null,
5.
6.     primary key ( user1_id, user2_id )
7. );

```

- Chat_record_single.sql

```

1. create table chat_record_single
2. (
3.     id int not null auto_increment,
4.     send_user_id varchar(255),
5.     receive_user_id varchar(255),
6.     chat_time datetime not null,
7.     content text,
8.
9.     primary key ( id )
10.);

```

3. The user's name and password to the database should be "slm" and "123456". And if you want to connect to your local database or remote database, please modify concerned parameters in Database.java.
4. Open the whole folder with IntelliJ IDEA

2 Overall Design

2.1 Project Structure

| -wechat

| ----.idea

| ----img

 | ----background.jpg

 | ----profile

 | ----3180103654@zju.edu.cn.jpg

 | ----zjuslm@163.com.jpg

 | ----zjuslm@126.com.jpg

```
|----lib
    |----activation.jar
    |----javax.mail.jar
    |----mail.jar
    |----mysql-connector-java-8.0.16.jar
|----out
|----src
    |----communication
        |----Client.java
        |----Client_Reader.java
        |----Server.java
        |----Server_Handler.java
    |----GUI
        |----Chat_Window.java
        |----Friend_List.java
        |----Login.java
        |----Register.java
        |----Session_Client.java
        |----Session_Server.java
    |----jdbc
        |----Database.java
        |----SendEmail.java
    |----Main.java (run the main program)
|----wechat.imi
```

2.2 Class Design

1. Communication

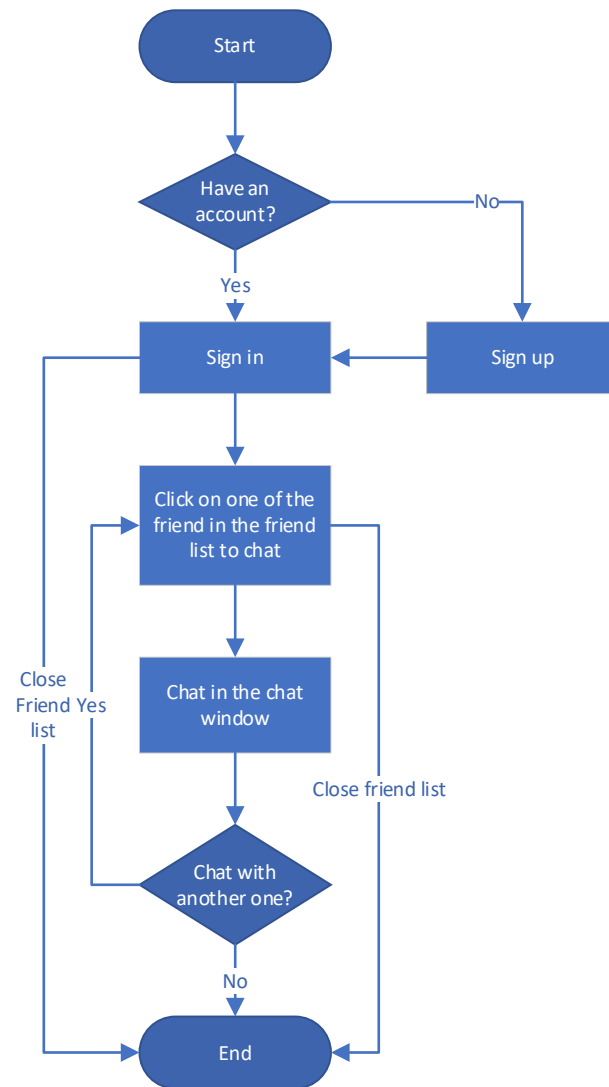
1) Client

The client class can generate a thread that individually handles the communication between the client and the server.

2) Client_Reader

- The client_reader class mainly deals with the message that the current client receives. After receive the message, the class decrypts it and post it on the user interface to let the user see the message he/she receives.
- 3) Server
The server class generates a thread that handles the information the server gets.
 - 4) Server_Handler
The server_handler class mainly deals with the encrypted message from client and decides who to re-send.
2. GUI
- 1) Chat_Window
The Chat_Window class mainly provides a comfortable user interface of chatting.
 - 2) Friend_List
The Friend_List class mainly provides a comfortable user interface that presents all the user's friend.
 - 3) Login
The Login class mainly provides a user interface for user to input user id and password to login, and can generate a new window for user to create a new account.
 - 4) Register
The Register class mainly provides a user interface for user to input his/her email, password, user name, real name, phone to create a new account. Before signing up, the inputted email address will be verified through the authentication code in the email sent by the program.
 - 5) Session_Client
Every time the user clicks a friend in the friend list, a Chat_Window class and a Session_Client class will be created that generate a client thread.
 - 6) Session_Server
Every time the program is started, the session_server will be created.
3. JDBC
- 1) Database
The Database class provides a well encapsulated interface to implement insertion, deletion, query, modification on the Database according to the inputted SQL statements.
 - 2) SendEmail
The SendEmail class provides a static method that send a email with a random authentication code according to the given email address.

2.3 Flowchart



3 Detailed Design

3.1 Communication

3.1.1 Client.java

```
1. public class Client
2. {
3.     private int port = 9000;          /* the port to listen */
4.     private String ip = "127.0.0.1";  /* the ip address */
5.     private static Socket socket;     /* the communication socket */
6.     private String client_id;         /* set the client thread's id as client_id */
7.     private final int BUFFER_LENGTH = 8192; /* the maximum length of message */
8.     private JPanel chat_record_area; /* the chat record area from the chat window */
9.
10.    /* the constructor function helps to initialize the client */
11.    public Client(String client_id, JPanel chat_record_area);
12.
13.    /* initialization */
14.    private void init(String client_id, JPanel chat_record_area);
15.
16.    /* create a new thread of client_reader according to the client_id */
17.    public void handle(JPanel chat_record_area);
18.
19.    /* send a single message to the server, with declaration of target id */
20.    public void send(String message, String target_id);
21.
22.    /* end the socket */
23.    public boolean cancel()
24. }
```

3.1.2 Client_Reader.java

```
1. public class Client_Reader implements Runnable
2. {
3.     private InputStream is; /* the input stream of the socket */
4.     private final int BUFFER_LENGTH = 8192; /* the maximum length of message */
5.     private JPanel chat_record_area; /* the chat record area */
6.     /* vaerify the send user id and receive user id */
7.     /* according to the encrypted message */
8.     private String user_id, user_id_source;
9.
10.    /* the constructor to initialize */
11.    public Client_Reader(InputStream is, JPanel chat_record_area, String user_id);
12.
13.    /* override the runnable interface's run method */
14.    /* which display the message the client receives */
15.    @Override
16.    public void run();
17. }
```

3.1.3 Server.java

```
1. public class Server
2. {
3.     private int port = 9000;    /* the listening port */
4.     private ServerSocket server; /* the server socket */
5.     private Socket socket;      /* the captured client socket */
6.     private String server_name; /* the server name */
7.     private final int BUFFER_LENGTH = 8192;
8.
9.     /* a hash map that stores the list of clients */
10.    private HashMap<String, Socket> client_list = new HashMap<String, Socket>();
11.
12.    /* initialize the server */
13.    public Server(String server_name);
14.
15.    /* initialize the server */
16.    private void init(String server_name);
17.
18.    /* create a new server handler thread */
19.    /* that deals with the listened socket */
20.    private void handle(Socket socket);
21.
22.    /* return the current client list */
23.    public HashMap<String, Socket> getClient_list();
24. }
```

3.1.4 Server_Handler.java

```
1. public class Server_Handler implements Runnable
2. {
3.     private Socket socket;    /* the listened client socket */
4.     private OutputStream os;  /* the output stream of socket */
5.     private InputStream is;   /* the input stream of socket */
6.     private final int BUFFER_LENGTH = 8192;
7.     private HashMap<String, Socket> client_list;
8.
9.     /* initialize the private variables */
10.    public Server_Handler(Socket socket, HashMap<String, Socket> client_list)
11.
12.    /* the override run() function mainly deals with */
13.    /* the message the server receives. */
14.    /* it can recognize the send user id, receive user id */
15.    /* and the message by decrypting the raw byte stream */
16.    /* and then decide who to send the message */
17.    @Override
18.    public void run()
19. }
```

3.2 GUI

3.2.1 Chat_Window.java

```
1. public class Chat_Window
2. {
3.     private Database sql; /* the jdbc interface */
4.     private String user_id; /* the current user's id */
5.     private String target_user_id; /* the send target user id */
6.     private JFrame frame = new JFrame(); /* the window frame */
7.
8.     /* initialize the window, search in the database */
9.     /* to get all chat history between the current user and target user */
10.    Chat_Window(String user_id, String target_user_id, Database sql);
11.
12.    /* display the chat record area and send message area */
13.    /* every time the chat window is created, a new thread of client session */
14.    /* will be created that deals with sending and receiving messages */
15.    public void init(String user_name, String target_user_name);
16. }
```

3.2.2 Friend_List.java

```
1. public class Friend_List
2. {
3.     private JFrame frame = new JFrame("Friend List");
4.     private Database sql;
5.     private String user_id; /* the current user id */
6.     private int friend_number = 0; /* the number of friends the user has */
7.     private ArrayList<String> friends = new ArrayList<>(); /* friend list */
8.
9.     /* initialize the frame */
10.    Friend_List(String user_id, Database sql);
11.
12.    /* search from the database and display the user's information */
13.    public void set_personal_information();
14.
15.    /* search from the database and display a list of friends' info */
16.    /* if the user click on one the list item, a new chat window will be created */
17.    public void init_friend();
18. }
```

3.2.3 Login.java

```
1. public class Login
2. {
3.     private JFrame frame = new JFrame("Login");
4.     private Database sql;
5.
6.     /* initialize */
7.     public Login(Database sql);
8.
9.     /* set the background picture of the fram */
10.    public void set_background();
11.
12.    /* display user id and password field */
13.    public void set_login_panel();
14.
15.    /* search in the database to authenticate */
16.    /* the inputted user id and password */
17.    public void handle_login(String user_id, String password)
18. }
```

3.2.4 Register.java

```
1. public class Register
2. {
3.     private JFrame frame = new JFrame("Register");
4.     private Database sql;
5.     private int code;
6.
7.     /* initialize */
8.     Register(Database sql);
9.
10.    /* display the register information field */
11.    /* send an authentication email and register */
12.    public void set_register_panel();
13. }
```

3.2.5 Session_Client.java

```
1. public class Session_Client implements Runnable
2. {
3.     private int length;    /* length of friends */
4.     private String user_id;
5.     private String user_name;
6.     private JButton send_button;
7.     private JTextArea send_message_area;
8.     private String target_user_id;
9.     private JPanel chat_record_area;
10.
11.    /* initialize */
12.    Session_Client(
13.        String user_name,
14.        String user_id,
15.        String target_user_id,
```

```

16.     JPanel chat_record_area,
17.     JTextArea send_message_area,
18.     JButton send_button,
19.     int length);
20.
21.     /* handle the received message */
22.     @Override
23.     public void run();
24. }

```

3.2.6 Session_Server.java

```

1. public class Session_Server implements Runnable
2. {
3.     /* Run a server session */
4.     @Override
5.     public void run()
6.     {
7.         Server server = new Server("leming");
8.     }
9. }

```

3.3 JDBC

3.3.1 Database.java

```

1. public class Database
2. {
3.     /* the configuration parameters */
4.     static final String JDBC_DRIVER = "com.mysql.cj.jdbc.Driver";
5.     static final String DB_URL = "jdbc:mysql://localhost:3306/wechat?useSSL=false&allowPublicKeyRetrieval=true&serverTimezone=UTC";
6.
7.     static final String USER = "slm";
8.     static final String PASS = "123456";
9.
10.    private Connection connection = null;
11.
12.    /* initialize */
13.    public Database();
14.
15.    /* execute insert, update, delete in database */
16.    public void modify(String sql, String[] args, String[] types);
17.
18.    /* execute select in database */
19.    public ResultSet query(String sql, String[] args, String[] types);
20. }

```

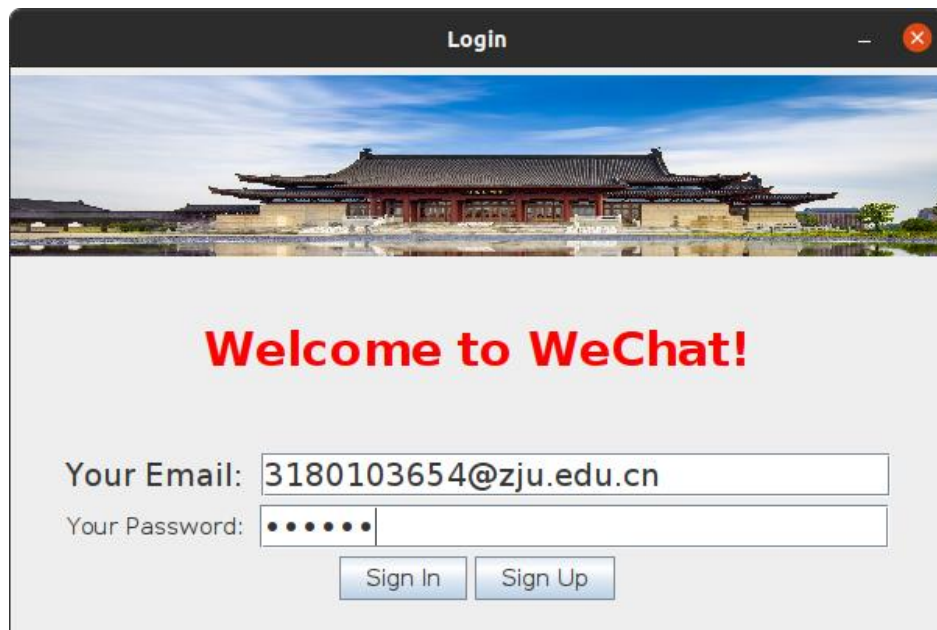
3.3.2 SendEmail.java

```
1. public class SendEmail
2. {
3.     /* send email to the recipient's email address */
4.     public static void send(String recipient, String code);
5. }
```

4 Testing & Running

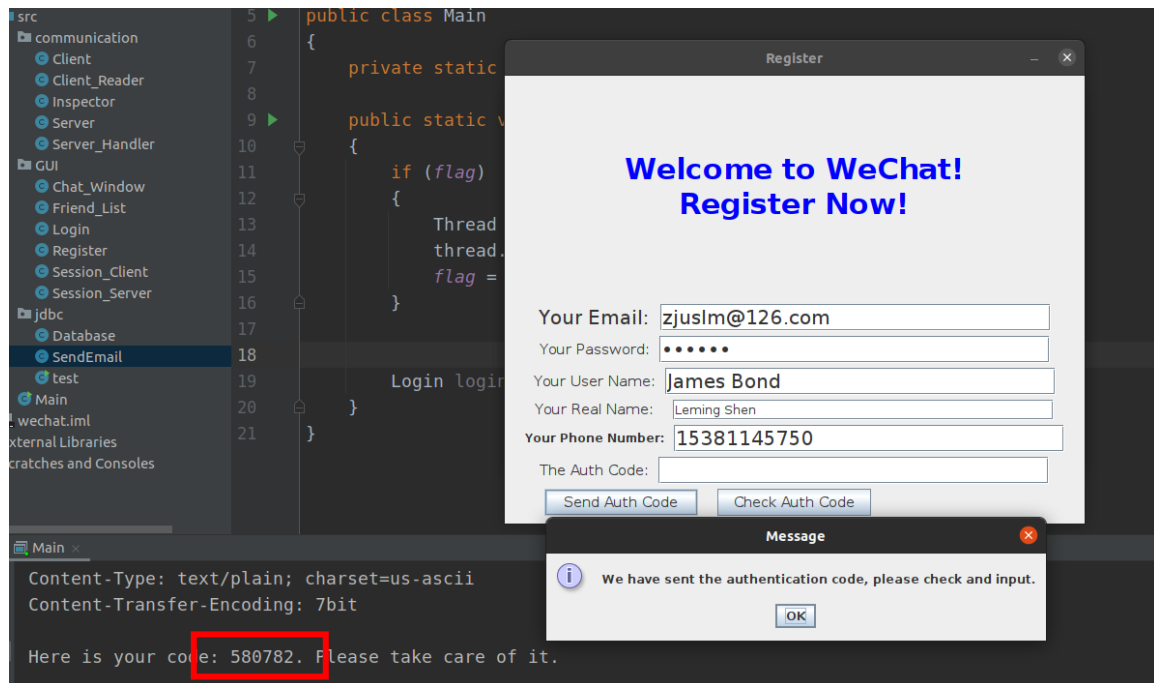
4.1 Login

The login window:

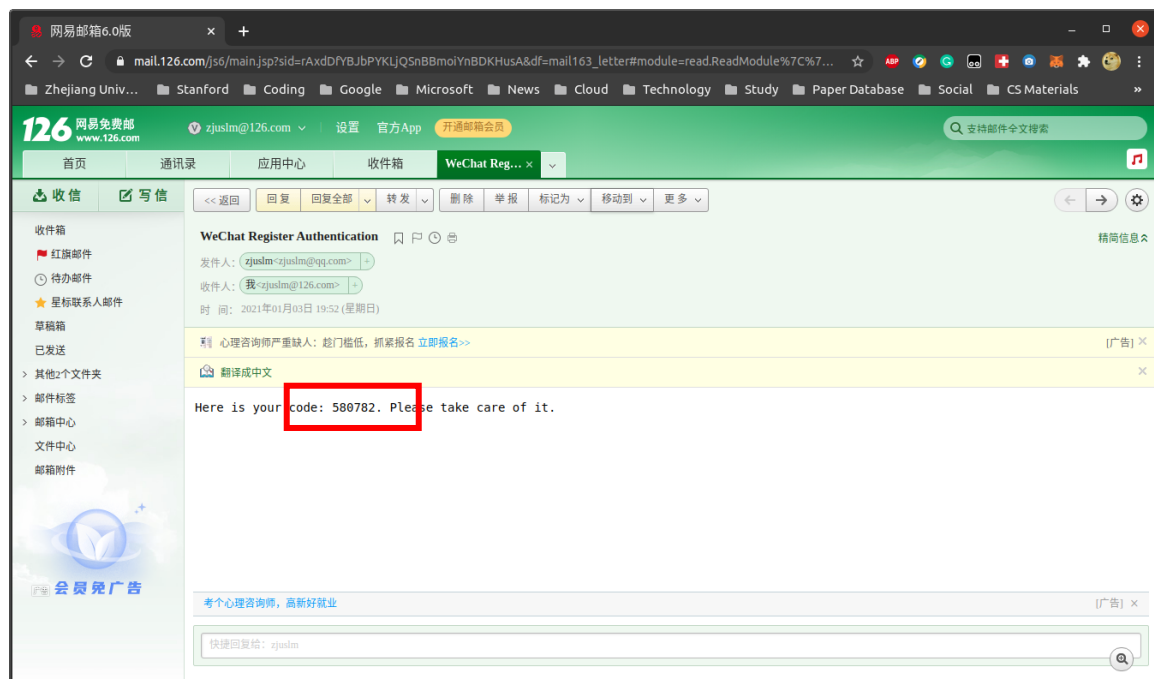


4.2 Register

The register window and sending an email to verify:



The email received the email:



Input the code and verify:



The screenshot shows a web browser window titled "Register". The main heading in blue text reads "Welcome to WeChat! Register Now!". Below this, there is a registration form with the following fields and values:

- Your Email: [Empty text box]
- Your Password: [Masked with dots]
- Your User Name: James Bond
- Your Real Name: Leming Shen
- Your Phone Number: 15381145750
- The Auth Code: 580782

At the bottom of the form are two buttons: "Send Auth Code" and "Check Auth Code". A modal dialog box titled "Succeed" is overlaid on the form, displaying an information icon, the text "Authentication successful!", and an "OK" button.

Register:



The screenshot shows the same "Register" window as above, but with a different modal dialog box. The modal dialog box is titled "Succeed" and displays an information icon, the text "Register Succeed!", and an "OK" button. The registration form fields and values remain the same as in the previous screenshot.

And the database has a new user:

MySQL Workbench

wechat

File Edit View Query Database Server Tools Scripting Help

Schemas

Filter objects

sys

wechat

Tables

Views

Stored Procedures

Functions

Query 1

Limit to 1000 rows

1 • select * from user

	user_id	password	user_name	name	phone
1	3180103654@zju.edu.cn	123456	ZJU_SLM	Leming Shen	15381145750
2	zjuslm@126.com	123456	James Bond	Leming Shen	15381145750
3	zjuslm@163.com	123456	test1	Leming	15381145750
*	NULL	NULL	NULL	NULL	NULL

user 4

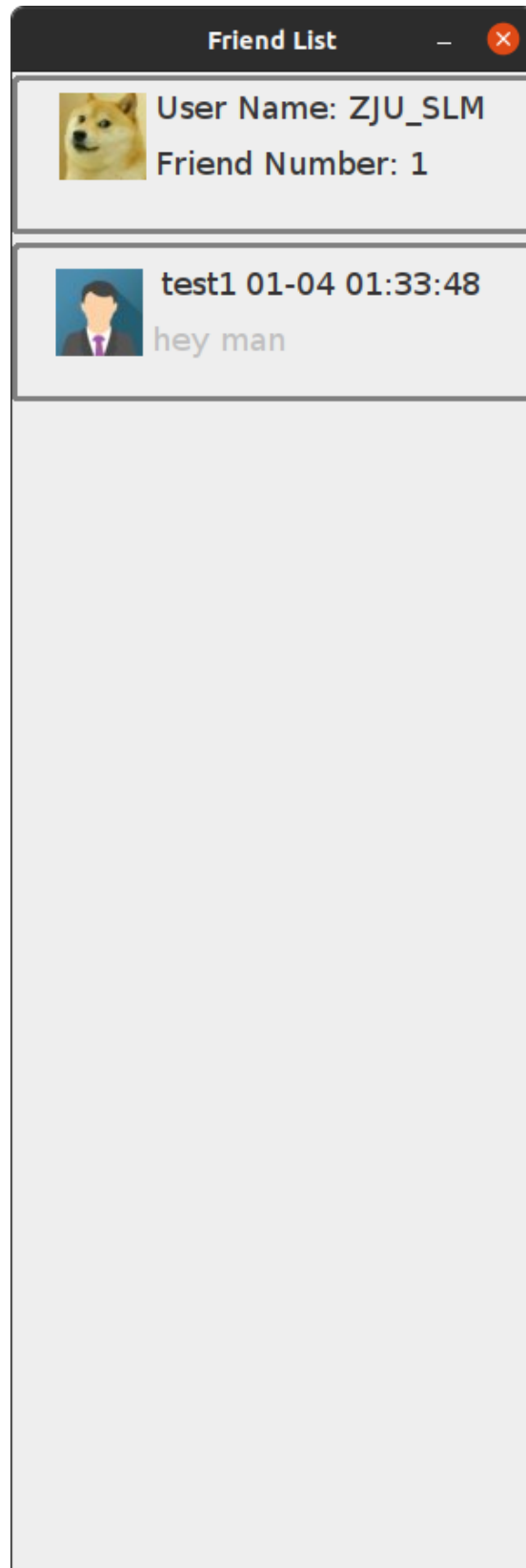
user 6

Action Output

#	Time	Action	Message
✓ 11	19:55:25	delete from user where user_id='zjuslm@126.com'	1 row(s) affected

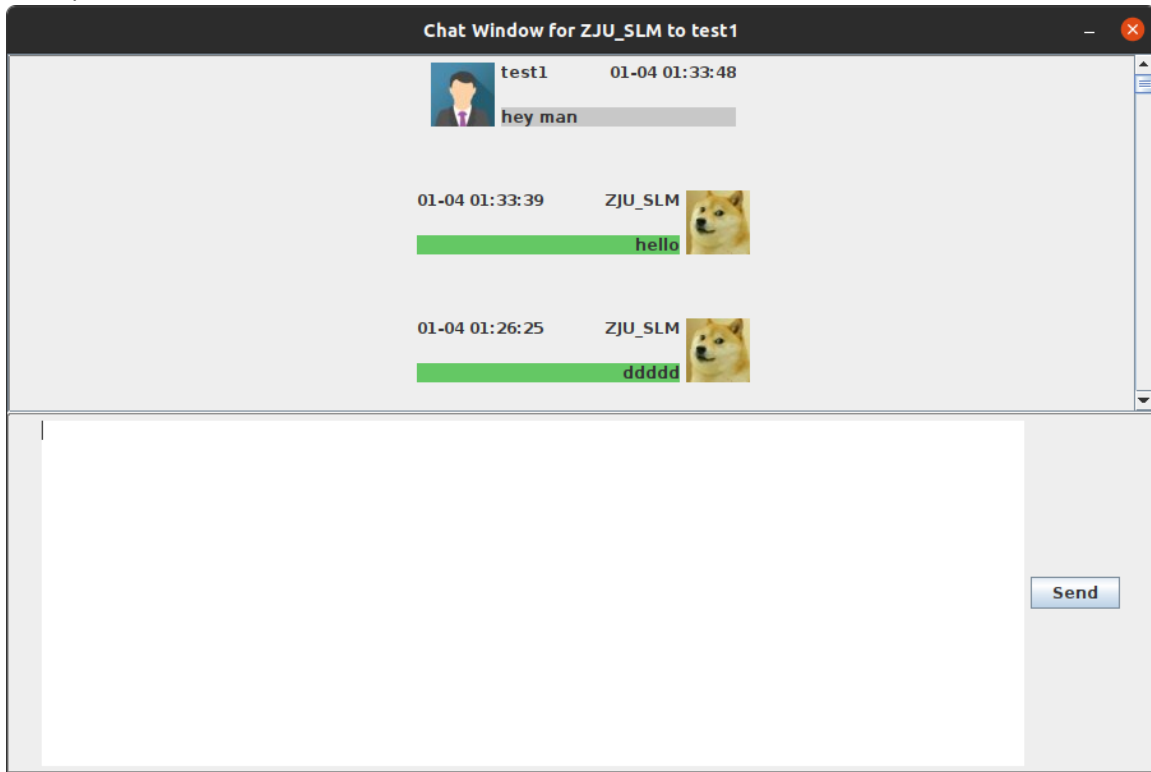
4.3 Friend List

Friend List:

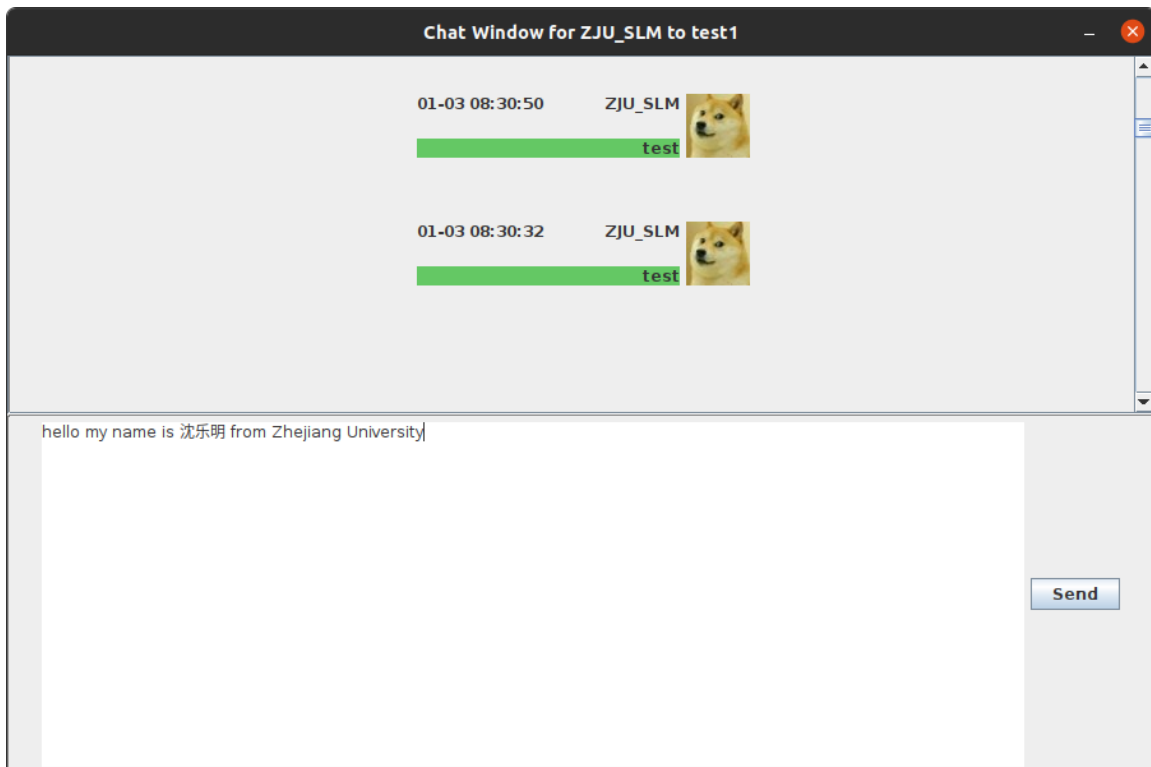


4.4 Chat Window

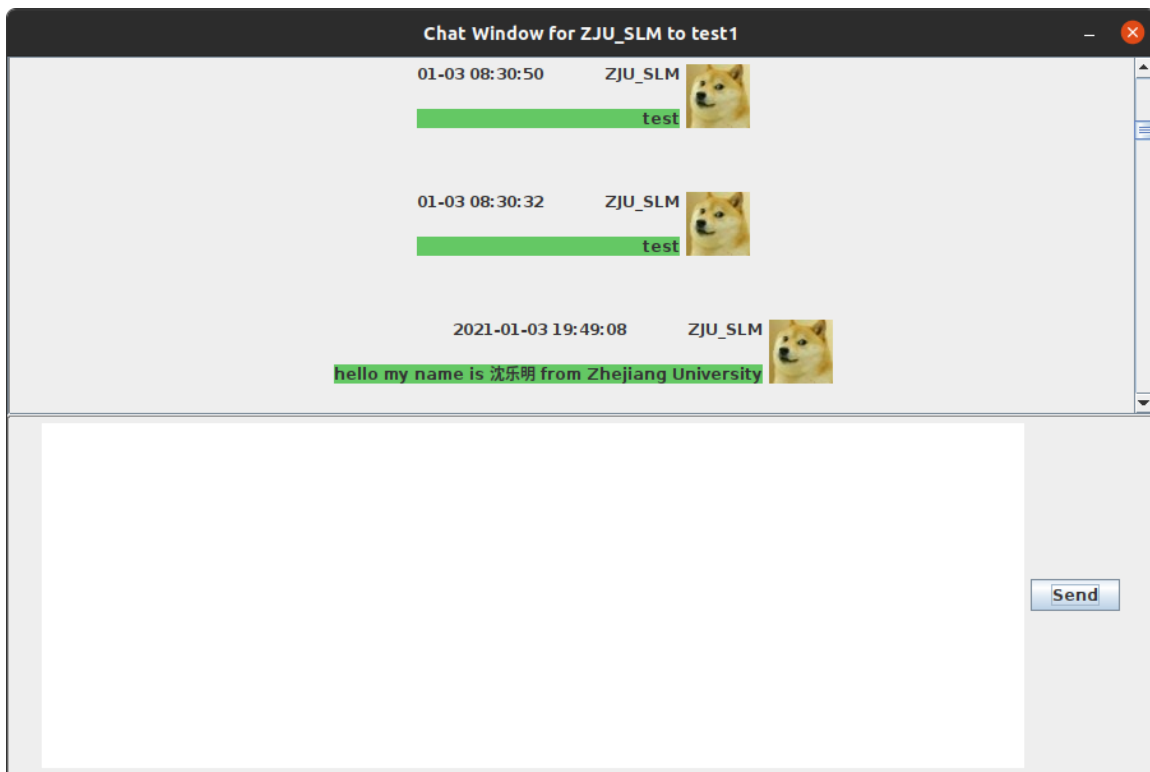
A simple chat window:



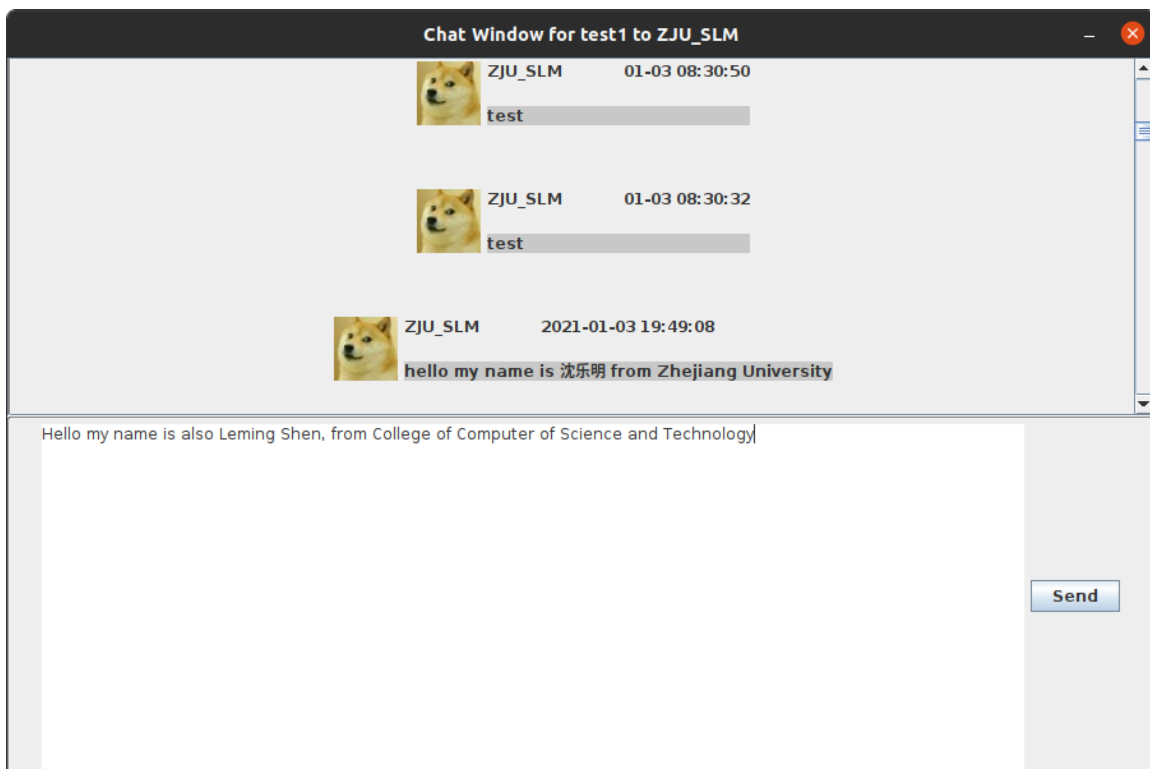
Input a message



Send the message:



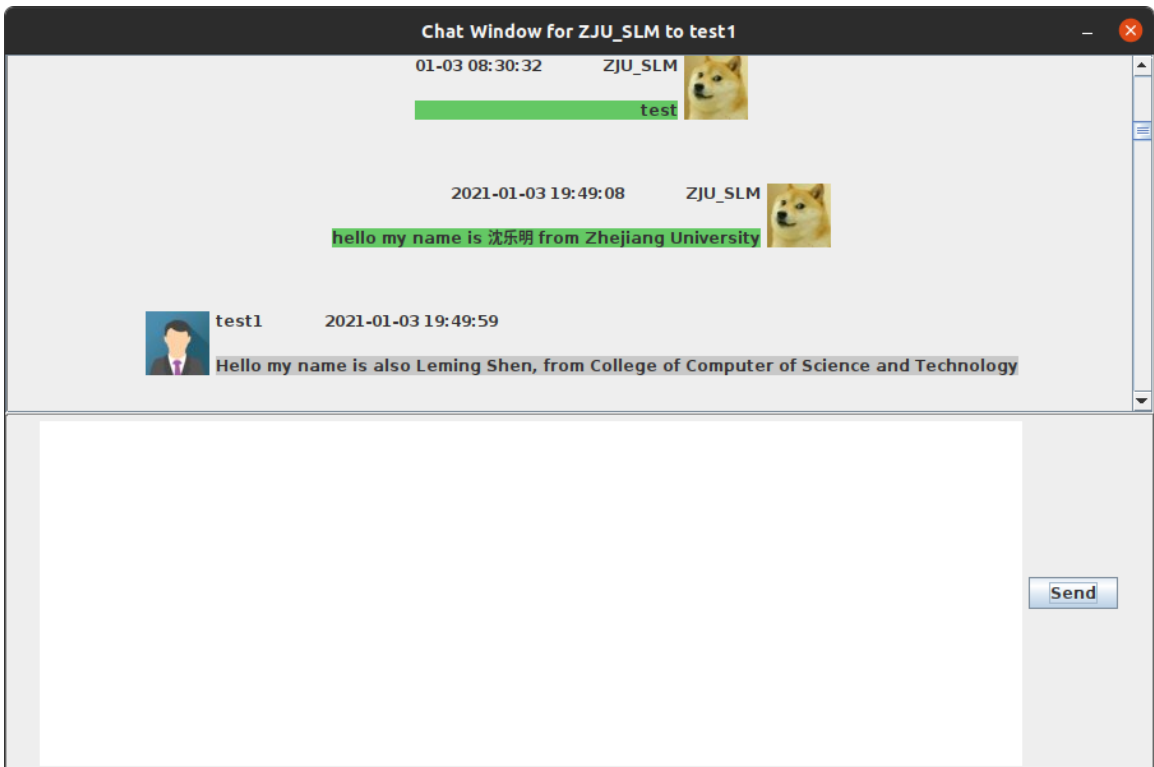
The other user receives the message:



The other user send back a message:



And the first user receives the message:



The chat records are all inserted into the database table:

The screenshot shows the MySQL Workbench interface with a query executed on the 'wechat' database. The query is `select * from chat_record_single`. The result set displays 18 rows of chat records. A red box highlights the last three rows, which contain messages from 'Zhelei' and 'Leming She'.

#	id	send_user_id	receive_user_id	chat_time	content
12	12	zjuslm@163.com	3180103654@zju.edu.cn	2021-01-03 15:11:12	asdfkjalsdgal;gna;lghaldgjaoghioag...
13	13	zjuslm@163.com	3180103654@zju.edu.cn	2021-01-03 15:11:14	asdfkjalsdgal;gna;lghaldgjaoghioag...
14	14	3180103654@zju.edu.cn	zjuslm@163.com	2021-01-03 17:26:25	dddd
15	15	3180103654@zju.edu.cn	zjuslm@163.com	2021-01-03 17:33:39	hello
16	16	zjuslm@163.com	3180103654@zju.edu.cn	2021-01-03 17:33:40	hey man
17	17	3180103654@zju.edu.cn	zjuslm@163.com	2021-01-03 19:49:07	hello my name is 沈乐明 from Zhelei...
18	18	zjuslm@163.com	3180103654@zju.edu.cn	2021-01-03 19:49:58	Hello my name is also Leming She...

The Action Output pane shows the following results:

#	Time	Action	Message
✓ 13	19:56:24	select * from user LIMIT 0, 1000	3 row(s) returned
✓ 14	19:57:00	select * from user LIMIT 0, 1000	3 row(s) returned
✓ 15	19:57:05	select * from chat_record_single LIMIT 0, 1000	18 row(s) returned

Query Completed

Add more friends to the user:

The screenshot shows the MySQL Workbench interface with a query executed on the 'wechat' database. The query is `select * from friend_pair`. The result set displays 6 rows of friend pairs. The Action Output pane shows the results of three insert statements.

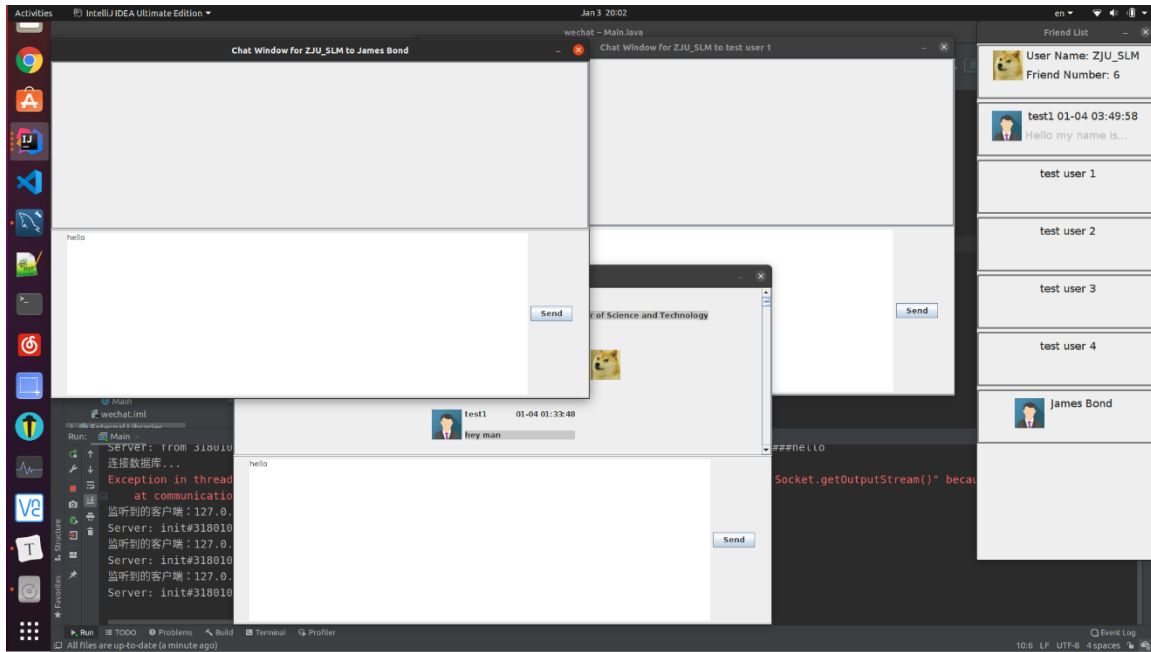
#	user1_id	user2_id
1	3180103654@zju.edu.cn	zjuslm@163.com
2	test1@test	3180103654@...
3	test2@test	3180103654@...
4	test3@test	3180103654@...
5	test4@test	3180103654@...
6	zjuslm@126.com	3180103654@...

The Action Output pane shows the following results:

#	Time	Action	Message	Duration / Fetch
✓ 1	19:59:31	insert into user values('test1@test','123456','test user ...	1 row(s) affected	0.970 sec
✓ 2	19:59:42	insert into user values('test2@test','123456','test user ...	1 row(s) affected	0.441 sec
✓ 3	19:59:49	insert into user values('test3@test','123456','test user ...	1 row(s) affected	0.419 sec

Query Completed

And then multiple window chat can be implemented:



5 Summary

The socket and database communication are easy to program.

However, I got really really bad impression on Java Swing, especially for its layout rule, which nearly drives me mad! 😞

6 References

1. *Introduction to JAVA Programming*, comprehensive version, 10th edition, Y. Daniel Liang.
2. *JAVA Concepts Early Objects*, 7th edition, Cay Horstmann, San Jose State University.
3. *Java Software Solutions*, Foundations of Program Design, Global Edition, John Lewis, William Loftus.
4. *Starting out with JAVA Early Objects*, 5th edition, Tony Daddis.

7 Source Code

7.1 Communication

1. Client.java

```
1. package communication;
2.
3. import javax.swing.*;
4. import java.io.IOException;
5. import java.net.Socket;
6. import java.net.UnknownHostException;
7. import java.nio.charset.StandardCharsets;
8. import java.util.Scanner;
9.
10. public class Client
11. {
12.     private int port = 9000;
13.     private String ip = "127.0.0.1";
14.     private static Socket socket;
15.     private String client_id;
16.     private final int BUFFER_LENGTH = 8192;
17.     private JPanel chat_record_area;
18.
19.     public Client(String client_id, JPanel chat_record_area)
20.     {
21.         try
22.         {
23.             init(client_id, chat_record_area);
24.             handle(this.chat_record_area);
25.         }
26.         catch (Exception e)
27.         {
28.             e.printStackTrace();
29.         }
30.     }
31.
32.     private void init(String client_id, JPanel chat_record_area) throws Exception
33.     {
34.         this.chat_record_area = chat_record_area;
35.         this.client_id = client_id;
36.         socket = new Socket(ip, port);
37.
38.         try
39.         {
40.             String information = "init#" + client_id;
41.             socket.getOutputStream().write(information.getBytes(StandardCharsets.UTF_8));
42.             socket.getOutputStream().flush();
43.         }
44.         catch (Exception e)
45.         {
46.             e.printStackTrace();
47.         }
48.     }
49.
50.     public void handle(JPanel chat_record_area) throws Exception
51.     {
52.         Thread thread = new Thread(new Client_Reader(socket.getInputStream(), chat_record_area, client_id));
```



```

53.         thread.setName(client_id);
54.         thread.start();
55.     }
56.
57.     public void send(String message, String target_id) throws IOException
58.     {
59.         try
60.         {
61.             String data_sent = "from " + client_id + " to " + target_id + "####mess
age####" + message;
62.             socket.getOutputStream().write(data_sent.getBytes(StandardCharsets.UTF_
8));
63.             socket.getOutputStream().flush();
64.             System.out.println(data_sent);
65.         }
66.         catch (Exception e)
67.         {
68.             e.printStackTrace();
69.         }
70.     }
71.
72.     public boolean cancel() throws IOException
73.     {
74.         try
75.         {
76.             socket.close();
77.         }
78.         catch (Exception e)
79.         {
80.             e.printStackTrace();
81.             return false;
82.         }
83.
84.         return true;
85.     }
86. }

```

2. Client_Reader.java

```

1. package communication;
2.
3. import jdbc.Database;
4.
5. import javax.swing.*;
6. import java.awt.*;
7. import java.io.BufferedReader;
8. import java.io.IOException;
9. import java.io.InputStream;
10. import java.io.InputStreamReader;
11. import java.sql.ResultSet;
12. import java.sql.SQLException;
13. import java.text.SimpleDateFormat;
14. import java.util.Date;
15.
16. public class Client_Reader implements Runnable
17. {
18.     private InputStream is;
19.     private final int BUFFER_LENGTH = 8192;
20.     private JPanel chat_record_area;
21.     private String user_id, user_id_source;

```

```

22.
23.     public Client_Reader(InputStream is, JPanel chat_record_area, String user_id)
24.     {
25.         this.is = is;
26.         this.user_id_source = user_id.split("_")[1];
27.         this.user_id = user_id.split("_")[0];
28.         this.chat_record_area = chat_record_area;
29.     }
30.
31.     @Override
32.     public void run()
33.     {
34.         try
35.         {
36.             while (true)
37.             {
38.                 byte[] b = new byte[BUFFER_LENGTH];
39.                 int length = is.read(b);
40.                 String message = new String(b, 0, length);
41.                 System.out.println(message);
42.                 System.out.println(user_id + "    " + user_id_source);
43.
44.                 message = message.split("####message####")[1];
45.
46.                 String query = "select user_name from user where user_id=?";
47.                 ResultSet result = (new Database()).query(query, new String[]{user_
id_source}, new String[]{"String"});
48.                 result.next();
49.
50.                 JPanel single_chat = new JPanel();
51.                 single_chat.setSize(900, 100);
52.                 single_chat.setPreferredSize(new Dimension(900, 100));
53.                 //single_chat.setLayout(new GridLayout(1, 2));
54.                 ImageIcon user_image = new ImageIcon("img/profile/" + user_id_sourc
e + ".jpg");
55.                 JLabel profile_label = new JLabel();
56.                 profile_label.setIcon(user_image);
57.                 profile_label.setSize(100, 100);
58.
59.                 JPanel chat_panel = new JPanel();
60.                 chat_panel.setLayout(new GridLayout(2, 1, 0, 20));
61.                 JLabel user_name_time = new JLabel(result.getString("user_name") +
"
" + (new SimpleDateFormat("yyyy-MM-
dd HH:mm:ss")).format(new Date()));
62.                 JLabel chat_content = new JLabel(message);
63.                 chat_content.setOpaque(true);
64.                 chat_panel.add(user_name_time);
65.                 chat_panel.add(chat_content);
66.
67.                 profile_label.setHorizontalAlignment(SwingConstants.LEFT);
68.                 user_name_time.setHorizontalAlignment(SwingConstants.LEFT);
69.                 chat_content.setHorizontalAlignment(SwingConstants.LEFT);
70.                 chat_content.setBackground(new Color(200, 200, 200));
71.
72.                 single_chat.add(profile_label);
73.                 single_chat.add(chat_panel);
74.
75.                 chat_record_area.setLayout(new GridLayout(100, 1));
76.                 chat_record_area.add(single_chat);
77.             }
78.         }

```

```

79.         catch (IOException | SQLException e)
80.         {
81.             e.printStackTrace();
82.         }
83.     }
84. }

```

3. Server.java

```

1. package communication;
2.
3. import java.io.IOException;
4. import java.net.ServerSocket;
5. import java.net.Socket;
6. import java.util.HashMap;
7. import java.util.Map;
8. import java.util.Scanner;
9.
10. public class Server
11. {
12.
13.     private int port = 9000;
14.     private ServerSocket server;
15.     private Socket socket;
16.     private String server_name;
17.     private final int BUFFER_LENGTH = 8192;
18.
19.     private HashMap<String, Socket> client_list = new HashMap<String, Socket>();
20.
21.     public Server(String server_name)
22.     {
23.         try
24.         {
25.             init(server_name);
26.         }
27.         catch (IOException | InterruptedException e)
28.         {
29.             e.printStackTrace();
30.         }
31.     }
32.
33.     private void init(String server_name) throws IOException, InterruptedException
34.     {
35.         this.server = new ServerSocket(port);
36.         this.server_name = server_name;
37.
38.         while (true)
39.         {
40.             socket = server.accept();
41.             handle(socket);
42.         }
43.     }
44.
45.     private void handle(Socket socket) throws IOException, InterruptedException
46.     {
47.         String key = socket.getInetAddress().getHostAddress() + ":" + socket.getPort();
48.         System.out.println("监听到的客户端: " + key);
49.

```

```

50.     Thread thread = new Thread(new Server_Handler(socket, this.client_list));
51.     thread.start();
52. }
53.
54. public HashMap<String, Socket> getClient_list()
55. {
56.     return this.client_list;
57. }
58. }

```

4. Server_Handler.java

```

1. package communication;
2.
3. import java.io.IOException;
4. import java.io.InputStream;
5. import java.io.OutputStream;
6. import java.net.Socket;
7. import java.nio.charset.StandardCharsets;
8. import java.util.HashMap;
9.
10. public class Server_Handler implements Runnable
11. {
12.     private Socket socket;
13.     private OutputStream os;
14.     private InputStream is;
15.     private final int BUFFER_LENGTH = 8192;
16.     private HashMap<String, Socket> client_list;
17.
18.     public Server_Handler(Socket socket, HashMap<String, Socket> client_list) throws
        IOException
19.     {
20.         this.socket = socket;
21.         this.is = socket.getInputStream();
22.         this.os = socket.getOutputStream();
23.         this.client_list = client_list;
24.     }
25.
26.     @Override
27.     public void run()
28.     {
29.         try
30.         {
31.             while (true)
32.             {
33.                 byte[] b = new byte[BUFFER_LENGTH];
34.                 int length = is.read(b);
35.                 String message = new String(b, 0, length);
36.
37.                 System.out.println("Server: " + message);
38.
39.                 Thread.sleep(1000);
40.
41.                 if (message.contains("init#"))
42.                 {
43.                     byte[] init_information = new byte[BUFFER_LENGTH];
44.                     int init_len = length;
45.                     String client_id = message.split("#")[1];
46.
47.                     this.client_list.put(client_id, socket);

```

```

48.             continue;
49.         }
50.
51.         String[] source = message.split("####message####");
52.         String[] header = source[0].split(" ");
53.         String client_target = header[3];
54.
55.         Socket target_socket = this.client_list.get(client_target);
56.         target_socket.getOutputStream().write(message.getBytes(StandardChar
sets.UTF_8));
57.         target_socket.getOutputStream().flush();
58.     }
59. }
60. catch (IOException | InterruptedException e)
61. {
62.     e.printStackTrace();
63. }
64. }
65. }

```

7.2 GUI

1. Chat_Window.java

```

1. package GUI;
2.
3. import jdbc.Database;
4. import jdk.dynalink.Operation;
5.
6. import javax.swing.*;
7. import java.awt.*;
8. import java.sql.ResultSet;
9. import java.sql.SQLException;
10.
11. public class Chat_Window
12. {
13.     private Database sql;
14.     private String user_id;
15.     private String target_user_id;
16.     private JFrame frame = new JFrame();
17.
18.     Chat_Window(String user_id, String target_user_id, Database sql) throws SQLExce
ption
19.     {
20.         this.sql = sql;
21.         this.user_id = user_id;
22.         this.target_user_id = target_user_id;
23.
24.         String query = "select user_name from user where user_id=?";
25.         ResultSet result = sql.query(query, new String[]{user_id}, new String[]{"St
ring"});
26.         result.next();
27.         String user_name = result.getString("user_name");
28.         query = "select user_name from user where user_id=?";
29.         result = sql.query(query, new String[]{target_user_id}, new String[]{"Strin
g"});
30.         result.next();
31.         String target_user_name = result.getString("user_name");
32.

```

```

33.         frame.setTitle("Chat Window for " + user_name + " to " + target_user_name);
34.
35.         frame.setSize(900, 600);
36.         frame.setLocationRelativeTo(null);
37.         frame.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
38.         frame.setLayout(new GridLayout(2, 1));
39.         frame.setVisible(true);
40.         frame.setResizable(false);
41.
42.         init(user_name, target_user_name);
43.     }
44.
45.     public void init(String user_name, String target_user_name) throws SQLException
46.     {
47.         JPanel chat_record_area = new JPanel();
48.         JPanel send_message_area = new JPanel();
49.         JScrollPane scrollPane = new JScrollPane(chat_record_area, ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED, ScrollPaneConstants.HORIZONTAL_SCROLLBAR_NEVER);
50.         frame.add(scrollPane);
51.         frame.add(send_message_area);
52.
53.         chat_record_area.setBorder(BorderFactory.createLineBorder(Color.gray, 1, true));
54.         send_message_area.setBorder(BorderFactory.createLineBorder(Color.gray, 1, true));
55.
56.         //chat_record_area.setLayout(new GridLayout());
57.
58.         JTextArea message_area = new JTextArea();
59.         message_area.setColumns(70);
60.         message_area.setLineWrap(true);
61.         message_area.setRows(18);
62.
63.         JButton send = new JButton("Send");
64.         send.setSize(100, 50);
65.         send.setVerticalAlignment(SwingConstants.BOTTOM);
66.
67.         send_message_area.add(message_area);
68.         send_message_area.add(send);
69.
70.         String query = "select chat_time, content, send_user_id, receive_user_id from chat_record_single where (send_user_id=? and receive_user_id=?) or (send_user_id=? and receive_user_id=?) order by chat_time desc";
71.         ResultSet result = sql.query(query, new String[]{user_id, target_user_id, target_user_id, user_id}, new String[]{"String", "String", "String", "String"});
72.         int length = 0;
73.         while (result.next())
74.         {
75.             length++;
76.             JPanel single_chat = new JPanel();
77.             single_chat.setSize(900, 100);
78.             single_chat.setPreferredSize(new Dimension(900, 100));
79.             //single_chat.setLayout(new GridLayout(1, 2));
80.             ImageIcon user_image = new ImageIcon("img/profile/" + user_id + ".jpg");
81.             ;
82.             ImageIcon target_user_image = new ImageIcon("img/profile/" + target_user_id + ".jpg");
83.             JLabel profile_label = new JLabel();

```

```

83.         profile_label.setSize(100, 100);
84.
85.         JPanel chat_panel = new JPanel();
86.         chat_panel.setLayout(new GridLayout(2, 1, 0, 20));
87.         JLabel user_name_time = new JLabel();
88.         JLabel chat_content = new JLabel(result.getString("content"));
89.         chat_content.setOpaque(true);
90.         chat_panel.add(user_name_time);
91.         chat_panel.add(chat_content);
92.
93.         if (result.getString("send_user_id").equals(user_id))
94.         {
95.             profile_label.setIcon(user_image);
96.             user_name_time.setText(result.getTimestamp("chat_time").toString().
substring(5, result.getTimestamp("chat_time").toString().indexOf('.')) + "
" + user_name);
97.
98.             profile_label.setHorizontalAlignment(SwingConstants.RIGHT);
99.             user_name_time.setHorizontalAlignment(SwingConstants.RIGHT);
100.            chat_content.setHorizontalAlignment(SwingConstants.RIGHT);
101.            chat_content.setBackground(new Color(100, 200, 100));
102.
103.            single_chat.add(chat_panel);
104.            single_chat.add(profile_label);
105.        }
106.        else
107.        {
108.            profile_label.setIcon(target_user_image);
109.            user_name_time.setText(target_user_name + " " + result
t.getTimestamp("chat_time").toString().substring(5, result.getTimestamp("chat_time"
).toString().indexOf('.')));
110.
111.            profile_label.setHorizontalAlignment(SwingConstants.LEFT);
112.            user_name_time.setHorizontalAlignment(SwingConstants.LEFT);
113.            chat_content.setHorizontalAlignment(SwingConstants.LEFT);
114.            chat_content.setBackground(new Color(200, 200, 200));
115.
116.            single_chat.add(profile_label);
117.            single_chat.add(chat_panel);
118.        }
119.
120.        chat_record_area.setLayout(new GridLayout(100, 1));
121.        single_chat.setLayout(new FlowLayout());
122.
123.        chat_record_area.add(single_chat);
124.    }
125.
126.    Thread thread = new Thread(new Session_Client(user_name, user_id, target
_user_id, chat_record_area, message_area, send, length));
127.    thread.start();
128. }
129. }

```

2. Friend_List.java

```

1. package GUI;
2.
3. import jdbc.Database;
4.
5. import javax.swing.*;

```

```

6. import java.awt.*;
7. import java.awt.event.MouseEvent;
8. import java.awt.event.MouseListener;
9. import java.sql.ResultSet;
10. import java.sql.SQLException;
11. import java.sql.Timestamp;
12. import java.util.ArrayList;
13. import java.util.Collections;
14.
15. public class Friend_List
16. {
17.     private JFrame frame = new JFrame("Friend List");
18.     private Database sql;
19.     private String user_id;
20.     private int friend_number = 0;
21.     private ArrayList<String> friends = new ArrayList<>();
22.
23.     Friend_List(String user_id, Database sql) throws SQLException
24.     {
25.         this.sql = sql;
26.         this.user_id = user_id;
27.
28.         String query = "select * from friend_pair where user1_id=? or user2_id=?";
29.         ResultSet result = sql.query(query, new String[]{user_id, user_id}, new String[]{"String", "String"});
30.
31.         while (result.next())
32.         {
33.             friend_number++;
34.
35.             if (result.getString("user1_id").equals(user_id))
36.             {
37.                 this.friends.add(result.getString("user2_id"));
38.             }
39.             else
40.             {
41.                 this.friends.add(result.getString("user1_id"));
42.             }
43.         }
44.
45.         frame.setLocationRelativeTo(null);
46.         frame.setLayout(new GridLayout(9, 1, 0, 5));
47.         frame.setSize(300, 900);
48.         frame.setResizable(false);
49.         frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
50.
51.         set_personal_information();
52.         init_friend();
53.
54.         frame.setVisible(true);
55.     }
56.
57.     public void set_personal_information() throws SQLException
58.     {
59.         String query = "select user_name from user where user_id=?";
60.         ResultSet result = sql.query(query, new String[]{user_id}, new String[]{"String"});
61.         result.next();
62.         String user_name = result.getString("user_name");
63.

```



```

64.         JPanel personal_information = new JPanel();
65.
66.         personal_information.setLayout(new GridLayout(1, 2, 20, 20));
67.         ImageIcon profile = new ImageIcon("img/profile/" + user_id + ".jpg");
68.         JLabel profile_label = new JLabel();
69.         profile_label.setSize(100, 100);
70.         profile_label.setIcon(profile);
71.         personal_information.add(profile_label);
72.
73.         JPanel right_panel = new JPanel();
74.         right_panel.setLayout(new GridLayout(2, 1, 0, 10));
75.         JLabel user_name_label = new JLabel("User Name: " + user_name);
76.         user_name_label.setFont(new Font(null, Font.PLAIN, 18));
77.         right_panel.add(user_name_label);
78.
79.         JLabel user_friend_number = new JLabel("Friend Number: " + friend_number);
80.         user_friend_number.setFont(new Font(null, Font.PLAIN, 18));
81.         right_panel.add(user_friend_number);
82.
83.         personal_information.add(right_panel);
84.
85.         personal_information.setLayout(new FlowLayout());
86.         personal_information.setBorder(BorderFactory.createLineBorder(Color.gray, 3
, true));
87.         frame.add(personal_information);
88.     }
89.
90.     public void init_friend() throws SQLException
91.     {
92.         for (int i = 0; i < friend_number; i++)
93.         {
94.             String current_friend_id = friends.get(i);
95.             String current_friend_name = "";
96.             String recent_chat_content = "";
97.             Timestamp chat_time = null;
98.             String query = "select content, chat_time from chat_record_single where
(send_user_id=? and receive_user_id=?) or (send_user_id=? and receive_user_id=?) o
rder by chat_time desc";
99.             ResultSet result = this.sql.query(query, new String[]{user_id, current_
friend_id, current_friend_id, user_id}, new String[]{"String", "String", "String",
"String"});
100.
101.             if (result.next())
102.             {
103.                 recent_chat_content = result.getString("content");
104.                 if (recent_chat_content.length() > 20)
105.                 {
106.                     recent_chat_content = recent_chat_content.substring(0, 16) +
"...";
107.                 }
108.                 else
109.                 {
110.                     recent_chat_content += String.join("", Collections.nCopies(2
0 - recent_chat_content.length(), " "));
111.                 }
112.
113.                 chat_time = result.getTimestamp("chat_time");
114.             }
115.
116.             query = "select user_name from user where user_id=?";

```

```

117.         result = this.sql.query(query, new String[]{current_friend_id}, new
String[]{"String"});
118.         result.next();
119.         current_friend_name = result.getString("user_name");
120.
121.         JPanel single_friend = new JPanel();
122.
123.         single_friend.setSize(290, 100);
124.         single_friend.setPreferredSize(new Dimension(290, 100));
125.         single_friend.setBorder(BorderFactory.createLineBorder(Color.gray, 3
, true));
126.         single_friend.setLayout(new GridLayout(1, 2, 0, 0));
127.
128.         ImageIcon profile = new ImageIcon("img/profile/" + current_friend_id
+ ".jpg");
129.         JLabel profile_label = new JLabel();
130.         profile_label.setIcon(profile);
131.         profile_label.setSize(100, 100);
132.         profile_label.setHorizontalAlignment(SwingConstants.CENTER);
133.         profile_label.setVerticalAlignment(SwingConstants.CENTER);
134.         single_friend.add(profile_label);
135.
136.         JPanel main_panel = new JPanel();
137.         main_panel.setLayout(new GridLayout(2, 1, 0, 0));
138.         JPanel user_name_time = new JPanel();
139.         user_name_time.setLayout(new GridLayout(1, 2, 20, 0));
140.         JLabel user_name = new JLabel(current_friend_name);
141.         user_name.setSize(90, 50);
142.         user_name.setFont(new Font(null, Font.PLAIN, 18));
143.         user_name.setHorizontalAlignment(SwingConstants.LEFT);
144.         user_name_time.add(user_name);
145.         JLabel user_time = new JLabel(chat_time == null ? "" : chat_time.toS
tring().substring(5, chat_time.toString().indexOf('.')));
146.         user_time.setSize(100, 50);
147.         user_time.setFont(new Font(null, Font.PLAIN, 18));
148.         user_time.setHorizontalAlignment(SwingConstants.RIGHT);
149.         user_name_time.add(user_time);
150.         main_panel.add(user_name_time);
151.         user_name_time.setLayout(new FlowLayout());
152.
153.         JLabel chat_record = new JLabel(recent_chat_content);
154.         chat_record.setHorizontalAlignment(SwingConstants.LEFT);
155.         chat_record.setForeground(Color.LIGHT_GRAY);
156.         chat_record.setFont(new Font(null, Font.PLAIN, 18));
157.         main_panel.add(chat_record);
158.
159.         single_friend.add(main_panel);
160.         single_friend.setLayout(new FlowLayout());
161.
162.         single_friend.addMouseListener(new MouseListener()
163.         {
164.             @Override
165.             public void mouseClicked(MouseEvent e)
166.             {
167.                 try
168.                 {
169.                     Chat_Window chat_window = new Chat_Window(user_id, curre
nt_friend_id, sql);
170.                 }
171.                 catch (SQLException throwables)
172.                 {

```

```

173.             throwables.printStackTrace();
174.         }
175.     }
176.
177.     @Override
178.     public void mousePressed(MouseEvent e)
179.     {
180.
181.     }
182.
183.     @Override
184.     public void mouseReleased(MouseEvent e)
185.     {
186.
187.     }
188.
189.     @Override
190.     public void mouseEntered(MouseEvent e)
191.     {
192.
193.     }
194.
195.     @Override
196.     public void mouseExited(MouseEvent e)
197.     {
198.
199.     }
200.     });
201.
202.     frame.add(single_friend);
203. }
204. }
205. }

```

3. Login.java

```

1. package GUI;
2.
3. import communication.Client;
4. import communication.Server;
5. import jdbc.*;
6.
7. import javax.swing.*;
8. import java.awt.*;
9. import java.awt.event.ActionEvent;
10. import java.awt.event.ActionListener;
11. import java.io.IOException;
12. import java.sql.ResultSet;
13.
14. public class Login
15. {
16.     private JFrame frame = new JFrame("Login");
17.     private Database sql;
18.
19.     public Login(Database sql)
20.     {
21.         this.sql = sql;
22.
23.         frame.setLayout(new GridLayout(3, 1, 0, 0));
24.         frame.setSize(600, 400);

```

```

25.         frame.setLocationRelativeTo(null);
26.         frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
27.         frame.setResizable(false);
28.
29.         set_background();
30.         set_login_panel();
31.
32.         frame.setVisible(true);
33.     }
34.
35.     public void set_background()
36.     {
37.         JPanel background_panel = new JPanel();
38.         background_panel.setSize(500, 350);
39.
40.         ImageIcon bg = new ImageIcon("img/background.jpg");
41.         bg.setImage(bg.getImage().getScaledInstance(600, 200, Image.SCALE_DEFAULT))
;
42.         JLabel background_label = new JLabel();
43.         background_label.setIcon(bg);
44.         background_panel.add(background_label);
45.
46.         frame.add(background_panel);
47.     }
48.
49.     public void set_login_panel()
50.     {
51.         JLabel welcome = new JLabel("Welcome to WeChat!");
52.         welcome.setFont(new Font(null, Font.BOLD, 30));
53.         welcome.setHorizontalAlignment(SwingConstants.CENTER);
54.         welcome.setForeground(Color.RED);
55.         frame.add(welcome);
56.
57.         JPanel login_panel = new JPanel(new GridLayout(3, 2, 10, 10));
58.
59.         JLabel login_hint = new JLabel("Your Email: ");
60.         login_hint.setFont(new Font(null, Font.PLAIN, 20));
61.         login_panel.add(login_hint);
62.
63.         JTextField user_id = new JTextField(20);
64.         user_id.setFont(new Font(null, Font.PLAIN, 20));
65.         login_panel.add(user_id);
66.
67.         JLabel password_hint = new JLabel("Your Password: ");
68.         password_hint.setFont(new Font(null, Font.PLAIN, 15));
69.         login_panel.add(password_hint);
70.
71.         JPasswordField password = new JPasswordField(20);
72.         password.setFont(new Font(null, Font.PLAIN, 20));
73.         login_panel.add(password);
74.
75.         JButton login = new JButton("Sign In");
76.         login.setFont(new Font(null, Font.PLAIN, 15));
77.         login.addActionListener(new ActionListener()
78.         {
79.             @Override
80.             public void actionPerformed(ActionEvent e)
81.             {
82.                 handle_login(user_id.getText(), new String(password.getPassword()))
;
83.             }

```

```

84.     });
85.     login_panel.add(login);
86.
87.     JButton register = new JButton("Sign Up");
88.     register.setFont(new Font(null, Font.PLAIN, 15));
89.     login_panel.add(register);
90.
91.     login_panel.setLayout(new FlowLayout());
92.
93.     register.addActionListener(new ActionListener()
94.     {
95.         @Override
96.         public void actionPerformed(ActionEvent e)
97.         {
98.             Register register1 = new Register(sql);
99.         }
100.    });
101.
102.    frame.add(login_panel);
103. }
104.
105.    public void handle_login(String user_id, String password)
106.    {
107.        try
108.        {
109.            String query = "select * from user where user_id=? and password=?";
110.            ResultSet result = sql.query(query, new String[]{user_id, password},
111.            new String[]{"String", "String"});
112.            result.last();
113.            if (result.getRow() > 0)
114.            {
115.                System.out.println("Login success");
116.                Friend_List friend_list = new Friend_List(user_id, sql);
117.                this.frame.setVisible(false);
118.            }
119.            else
120.            {
121.                System.out.println("fail");
122.            }
123.        }
124.        catch (Exception e)
125.        {
126.            e.printStackTrace();
127.        }
128.    }
129. }

```

4. Register.java

```

1. package GUI;
2.
3. import jdbc.Database;
4. import jdbc.SendEmail;
5.

```

```

6. import javax.mail.MessagingException;
7. import javax.swing.*;
8. import java.awt.*;
9. import java.awt.event.ActionEvent;
10. import java.awt.event.ActionListener;
11. import java.nio.charset.StandardCharsets;
12. import java.security.GeneralSecurityException;
13. import java.sql.SQLException;
14.
15. public class Register
16. {
17.     private JFrame frame = new JFrame("Register");
18.     private Database sql;
19.     private int code;
20.
21.     Register(Database sql)
22.     {
23.         this.sql = sql;
24.
25.         frame.setLayout(new GridLayout(2, 1, 0, 0));
26.         frame.setSize(600, 500);
27.         frame.setLocationRelativeTo(null);
28.         frame.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
29.         frame.setResizable(false);
30.
31.         set_register_panel();
32.
33.         frame.setVisible(true);
34.         frame.setVisible(true);
35.     }
36.
37.     public void set_register_panel()
38.     {
39.         JLabel welcome = new JLabel("<html><body><p align='center'>Welcome to WeChat<br>Register Now!</p></body></html>");
40.         welcome.setFont(new Font(null, Font.BOLD, 30));
41.         welcome.setHorizontalAlignment(SwingConstants.CENTER);
42.         welcome.setForeground(Color.BLUE);
43.         frame.add(welcome);
44.
45.         JPanel register_panel = new JPanel(new GridLayout(7, 2, 10, 10));
46.
47.         JLabel register_hint = new JLabel("Your Email: ");
48.         register_hint.setFont(new Font(null, Font.PLAIN, 20));
49.         register_panel.add(register_hint);
50.
51.         JTextField user_id = new JTextField(20);
52.         user_id.setFont(new Font(null, Font.PLAIN, 20));
53.         register_panel.add(user_id);
54.
55.         JLabel password_hint = new JLabel("Your Password: ");
56.         password_hint.setFont(new Font(null, Font.PLAIN, 15));
57.         register_panel.add(password_hint);
58.
59.         JPasswordField password = new JPasswordField(20);
60.         password.setFont(new Font(null, Font.PLAIN, 20));
61.         register_panel.add(password);
62.
63.         JLabel user_name_hint = new JLabel("Your User Name: ");
64.         user_name_hint.setFont(new Font(null, Font.PLAIN, 15));
65.         register_panel.add(user_name_hint);

```

```

66.
67.     JTextField user_name = new JTextField();
68.     user_name.setFont(new Font(null, Font.PLAIN, 20));
69.     user_name.setColumns(20);
70.     register_panel.add(user_name);
71.
72.     JLabel name_hint = new JLabel("Your Real Name: ");
73.     name_hint.setFont(new Font(null, Font.PLAIN, 15));
74.     register_panel.add(name_hint);
75.
76.     JTextField name_field = new JTextField();
77.     name_field.setFont(new Font(null, Font.PLAIN, 30));
78.     name_field.setColumns(30);
79.     register_panel.add(name_field);
80.
81.     JLabel phone_hint = new JLabel("Your Phone Number: ");
82.     name_field.setFont(new Font(null, Font.PLAIN, 13));
83.     register_panel.add(phone_hint);
84.
85.     JTextField phone = new JTextField();
86.     phone.setFont(new Font(null, Font.PLAIN, 20));
87.     phone.setColumns(20);
88.     register_panel.add(phone);
89.
90.     JLabel auth_code_hint = new JLabel("The Auth Code: ");
91.     auth_code_hint.setFont(new Font(null, Font.PLAIN, 15));
92.     register_panel.add(auth_code_hint);
93.
94.     JTextField auth_code = new JTextField();
95.     auth_code.setFont(new Font(null, Font.PLAIN, 20));
96.     auth_code.setColumns(20);
97.     register_panel.add(auth_code);
98.
99.     JPanel button_panel = new JPanel();
100.     button_panel.setLayout(new GridLayout(1, 3, 20, 0));
101.     register_panel.add(button_panel);
102.
103.     JButton send_auth_code = new JButton("Send Auth Code");
104.     send_auth_code.setFont(new Font(null, Font.PLAIN, 15));
105.     button_panel.add(send_auth_code);
106.
107.     send_auth_code.addActionListener(new ActionListener()
108.     {
109.         @Override
110.         public void actionPerformed(ActionEvent e)
111.         {
112.             code = (int)(Math.random() * 1000000) + 100000;
113.             System.out.println(code);
114.
115.             if (!user_id.getText().equals("") && user_id.getText().contains(
116.                 "@"))
117.             {
118.                 try
119.                 {
120.                     SendEmail.send(user_id.getText(), Integer.toString(code)
121.                 );
122.                     JOptionPane.showMessageDialog(null, "We have sent the au
123.                 thentication code, please check and input.");
124.                 }
125.                 catch (MessagingException messagingException)
126.                 {

```

```

124.             messagingException.printStackTrace();
125.         }
126.         catch (GeneralSecurityException generalSecurityException)
127.         {
128.             generalSecurityException.printStackTrace();
129.         }
130.     }
131. }
132. });
133.
134. JButton check = new JButton("Check Auth Code");
135. check.setFont(new Font(null, Font.PLAIN, 15));
136. JButton register = new JButton("Register");
137. register.setFont(new Font(null, Font.PLAIN, 15));
138. register.setVisible(false);
139. check.addActionListener(new ActionListener()
140. {
141.     @Override
142.     public void actionPerformed(ActionEvent e)
143.     {
144.         String input_code = auth_code.getText();
145.
146.         if (input_code.equals(Integer.toString(code)))
147.         {
148.             JOptionPane.showMessageDialog(null, "Authentication successf
149. ul!", "Succeed", JOptionPane.INFORMATION_MESSAGE);
150.             register.setVisible(true);
151.         }
152.         else
153.         {
154.             JOptionPane.showMessageDialog(null, "Authentication failed!"
155. , "Error", JOptionPane.ERROR_MESSAGE);
156.         }
157.     }
158. });
159.
160. register.addActionListener(new ActionListener()
161. {
162.     @Override
163.     public void actionPerformed(ActionEvent e)
164.     {
165.         String email = user_id.getText();
166.         String passcode = new String(password.getPassword());
167.         String username = user_name.getText();
168.         String real_name = name_field.getText();
169.         String phone_number = phone.getText();
170.
171.         String insert = "insert into user values(?, ?, ?, ?, ?)";
172.         try
173.         {
174.             sql.modify(insert, new String[]{email, passcode, username, r
175. eal_name, phone_number}, new String[]{"String", "String", "String", "String", "Stri
176. ng"});
177.             JOptionPane.showMessageDialog(null, "Register Succeed!", "Su
178. cceed", JOptionPane.INFORMATION_MESSAGE);
179.         }
180.         catch (SQLException throwables)
181.         {
182.             throwables.printStackTrace();
183.         }
184.     }
185. });

```



```

180.         }
181.     });
182.
183.     button_panel.add(check);
184.     button_panel.add(register);
185.
186.     register_panel.setLayout(new FlowLayout());
187.
188.     frame.add(register_panel);
189. }
190. }

```

5. Session_Client.java

```

1. package GUI;
2.
3.
4. import communication.Client;
5. import jdbc.Database;
6.
7. import javax.swing.*;
8. import java.awt.*;
9. import java.awt.event.ActionEvent;
10. import java.awt.event.ActionListener;
11. import java.io.IOException;
12. import java.sql.SQLException;
13. import java.sql.Timestamp;
14. import java.text.SimpleDateFormat;
15. import java.util.Date;
16.
17. public class Session_Client implements Runnable
18. {
19.     private int length;
20.     private String user_id;
21.     private String user_name;
22.     private JButton send_button;
23.     private JTextArea send_message_area;
24.     private String target_user_id;
25.     private JPanel chat_record_area;
26.
27.     Session_Client(String user_name, String user_id, String target_user_id, JPanel
chat_record_area, JTextArea send_message_area, JButton send_button, int length)
28.     {
29.         this.user_id = user_id;
30.         this.user_name = user_name;
31.         this.send_button = send_button;
32.         this.send_message_area = send_message_area;
33.         this.target_user_id = target_user_id;
34.         this.chat_record_area = chat_record_area;
35.     }
36.
37.     @Override
38.     public void run()
39.     {
40.         Client client = new Client(user_id + "_" + target_user_id, chat_record_area
);
41.
42.         send_button.addActionListener(new ActionListener()
43.         {
44.             @Override

```

```

45.         public void actionPerformed(ActionEvent e)
46.         {
47.             String message = send_message_area.getText();
48.
49.             if (message.equals(""))
50.             {
51.                 return;
52.             }
53.
54.             try
55.             {
56.                 client.send(message, target_user_id + "_" + user_id);
57.
58.                 String query = "insert into chat_record_single values(null, ?,
?, now(), ?)";
59.                 (new Database()).modify(query, new String[]{user_id, target_use
r_id, message}, new String[]{"String", "String", "String"});
60.
61.                 JPanel single_chat = new JPanel();
62.                 single_chat.setSize(900, 100);
63.                 single_chat.setPreferredSize(new Dimension(900, 100));
64.                 //single_chat.setLayout(new GridLayout(1, 2));
65.                 ImageIcon user_image = new ImageIcon("img/profile/" + user_id +
".jpg");
66.                 JLabel profile_label = new JLabel();
67.                 profile_label.setIcon(user_image);
68.                 profile_label.setSize(100, 100);
69.
70.                 JPanel chat_panel = new JPanel();
71.                 chat_panel.setLayout(new GridLayout(2, 1, 0, 20));
72.                 JLabel user_name_time = new JLabel((new SimpleDateFormat("yyyy-
MM-dd HH:mm:ss")).format(new Date()) + " " + user_name);
73.                 JLabel chat_content = new JLabel(message);
74.                 chat_content.setOpaque(true);
75.                 chat_panel.add(user_name_time);
76.                 chat_panel.add(chat_content);
77.
78.                 profile_label.setHorizontalAlignment(SwingConstants.RIGHT);
79.                 user_name_time.setHorizontalAlignment(SwingConstants.RIGHT);
80.                 chat_content.setHorizontalAlignment(SwingConstants.RIGHT);
81.                 chat_content.setBackground(new Color(100, 200, 100));
82.
83.                 single_chat.add(chat_panel);
84.                 single_chat.add(profile_label);
85.
86.                 chat_record_area.setLayout(new GridLayout(100, 1));
87.                 //single_chat.setLayout(new FlowLayout());
88.
89.                 chat_record_area.add(single_chat);
90.                 send_message_area.setText("");
91.             }
92.             catch (IOException | SQLException ioException)
93.             {
94.                 ioException.printStackTrace();
95.             }
96.         }
97.     });
98. }
99. }

```

6. Session_Server.java

```
1. package GUI;
2.
3.
4. import communication.Client;
5. import communication.Server;
6.
7. import java.io.IOException;
8. import java.io.InputStream;
9. import java.io.OutputStream;
10. import java.net.Socket;
11. import java.nio.charset.StandardCharsets;
12. import java.util.HashMap;
13.
14. public class Session_Server implements Runnable
15. {
16.     @Override
17.     public void run()
18.     {
19.         Server server = new Server("leming");
20.     }
21. }
```

7.3 JDBC

1. Database.java

```
1. package jdbc;
2.
3. import java.awt.*;
4. import java.sql.*;
5. import java.util.ArrayList;
6. import java.util.Map;
7. import java.util.Date;
8.
9. public class Database
10. {
11.     static final String JDBC_DRIVER = "com.mysql.cj.jdbc.Driver";
12.     static final String DB_URL = "jdbc:mysql://localhost:3306/wechat?useSSL=false&allowPublicKeyRetrieval=true&serverTimezone=UTC";
13.
14.     static final String USER = "slm";
15.     static final String PASS = "123456";
16.
17.     private Connection connection = null;
18.
19.     public Database()
20.     {
21.         try
22.         {
23.             Class.forName(JDBC_DRIVER);
24.
25.             System.out.println("连接数据库...");
26.             connection = DriverManager.getConnection(DB_URL, USER, PASS);
27.         }
28.         catch (SQLException se)
29.         {
30.             se.printStackTrace();
31.         }
32.     }
33. }
```

```

32.         catch (Exception e)
33.         {
34.             e.printStackTrace();
35.         }
36.     }
37.
38.     public void modify(String sql, String[] args, String[] types) throws SQLException
on
39.     {
40.         try
41.         {
42.             PreparedStatement statement = this.connection.prepareStatement(sql);
43.             for (int i = 0; i < args.length; i++)
44.             {
45.                 if (types[i].equals("String"))
46.                 {
47.                     statement.setString(i + 1, args[i]);
48.                 }
49.                 else if (types[i].equals("Datetime"))
50.                 {
51.                     statement.setTimestamp(i + 1, new Timestamp((new Date()).getTim
e()));
52.                 }
53.                 else if (types[i].equals("Int"))
54.                 {
55.                     statement.setInt(i + 1, Integer.parseInt(args[i]));
56.                 }
57.             }
58.
59.             statement.executeUpdate();
60.         }
61.         catch (SQLException e)
62.         {
63.             e.printStackTrace();
64.         }
65.         catch (Exception e)
66.         {
67.             e.printStackTrace();
68.         }
69.     }
70.
71.     public ResultSet query(String sql, String[] args, String[] types)
72.     {
73.         try
74.         {
75.             PreparedStatement statement = this.connection.prepareStatement(sql);
76.             for (int i = 0; i < args.length; i++)
77.             {
78.                 if (types[i].equals("String"))
79.                 {
80.                     statement.setString(i + 1, args[i]);
81.                 }
82.                 else if (types[i].equals("Datetime"))
83.                 {
84.                     statement.setTimestamp(i + 1, new Timestamp((new Date()).getTim
e()));
85.                 }
86.                 else if (types[i].equals("Int"))
87.                 {
88.                     statement.setInt(i + 1, Integer.parseInt(args[i]));
89.                 }

```

```

90.         }
91.
92.         ResultSet result = statement.executeQuery();
93.
94.         return result;
95.     }
96.     catch (SQLException e)
97.     {
98.         e.printStackTrace();
99.     }
100.    catch (Exception e)
101.    {
102.        e.printStackTrace();
103.    }
104.
105.    return null;
106. }
107. }

```

2. SendEmail.java

```

1. package jdbc;
2.
3. import javax.mail.*;
4. import javax.mail.internet.InternetAddress;
5. import javax.mail.internet.MimeMessage;
6. import java.security.GeneralSecurityException;
7. import java.util.Properties;
8.
9. public class SendEmail
10. {
11.     public static void send(String recipient, String code) throws MessagingException, GeneralSecurityException
12.     {
13.         try
14.         {
15.             Properties properties = new Properties();
16.             properties.put("mail.transport.protocol", "smtp");// 连接协议
17.             properties.put("mail.smtp.host", "smtp.qq.com");// 主机名
18.             properties.put("mail.smtp.port", 465);// 端口号
19.             properties.put("mail.smtp.auth", "true");
20.             properties.put("mail.smtp.ssl.enable", "true");// 设置是否使用 ssl 安全连接 ---一般都使用
21.             properties.put("mail.debug", "true");// 设置是否显示 debug 信息 true 会在控制台显示相关信息
22.
23.             // 得到回话对象
24.             Session session = Session.getInstance(properties);
25.             // 获取邮件对象
26.             Message message = new MimeMessage(session);
27.             // 设置发件人邮箱地址
28.             message.setFrom(new InternetAddress("zjuslm@qq.com"));
29.             // 设置收件人邮箱地址
30.             message.setRecipient(Message.RecipientType.TO, new InternetAddress(recipient)); // 一个收件人
31.             // 设置邮件标题
32.             message.setSubject("WeChat Register Authentication");
33.             // 设置邮件内容

```

```
34.         message.setText("Here is your code: " + code + ". Please take care of i
    t.");
35.         // 得到邮差对象
36.         Transport transport = session.getTransport();
37.         // 连接自己的邮箱账户
38.         transport.connect("zjuslm@qq.com", "wohjryfplfvqhgd");// 密码为 QQ 邮箱
    开通的 stmp 服务后得到的客户端授权码
39.         // 发送邮件
40.         transport.sendMessage(message, message.getAllRecipients());
41.         transport.close();
42.     }
43.     catch (Exception e)
44.     {
45.         e.printStackTrace();
46.     }
47. }
48. }
```

