

---

# **DOCUMENTATION OF SOCIAL NETWORK PROJECT**

---

June 29, 2017

He Yan

# Contents

1	Introduction . . . . .	1
1.1	Main Features . . . . .	1
1.2	Components . . . . .	1
2	Environment . . . . .	1
3	Data Structure . . . . .	2
3.1	Entity-Relationship Diagram . . . . .	2
3.2	MySQL Table . . . . .	2
4	Division of Labor . . . . .	5
5	Kernel Codes . . . . .	5
5.1	Google reCaptcha . . . . .	5
5.2	Two-step friends . . . . .	6
6	Website Preview . . . . .	8
7	References . . . . .	11

---

# 1 Introduction

This project aims to build a social network with JSP and MySQL.

## 1.1 Main Features

- *Compulsory:*
  - Sign up & in
  - Search for contacts & Post status and reply
  - 30 secs refreshment
- *Optional:*
  - Email address regex check
  - Ajax
  - Add Google's reCaptcha<sup>1</sup> validation
  - Two-step friends

## 1.2 Components

- Apache, Tomcat, Apache-Tomcat-Connector
- MySQL, MySQL Connector/J (JDBC)

Visit our project site at [Database Course Project](#).

# 2 Environment

This project is hosted on Amazon Linux AMI server provided by AWS. To build the environment for running our website, we took steps as below.

1. install OpenJDK-1.8.0
2. install and configure Apache (httpd) & Tomcat

---

<sup>1</sup>Completely Automated Public Turing test to tell Computers and Humans Apart

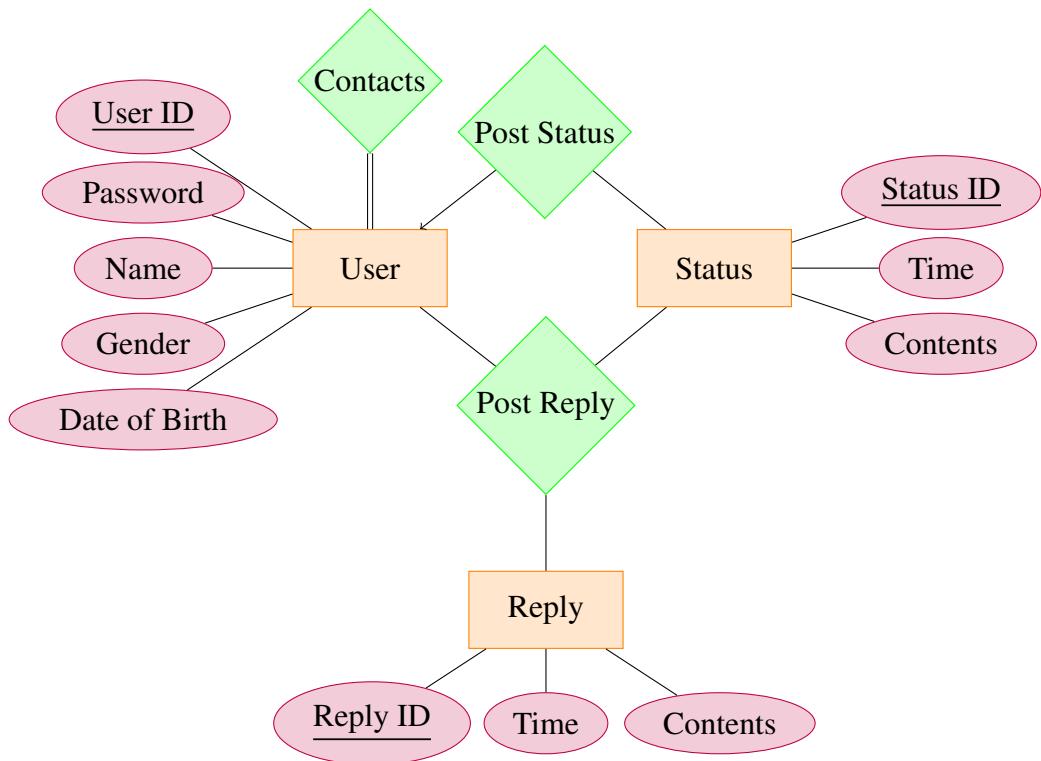
- 
3. link Apache and Tomcat with Apache Tomcat Connector <sup>2</sup>
  4. install MySQL<sup>3</sup> and prepare MySQL connector/J in WEB-INF/lib

Note: Our project has been hosted at GitHub. Visit our project at <https://github.com/PKU-2017-Database/Social-Network>.

## 3 Data Structure

### 3.1 Entity-Relationship Diagram

Here is an English version of ER Diagram redrawn by L<sup>A</sup>T<sub>E</sub>X.



**Figure 1:** Entity-Relationship Diagram

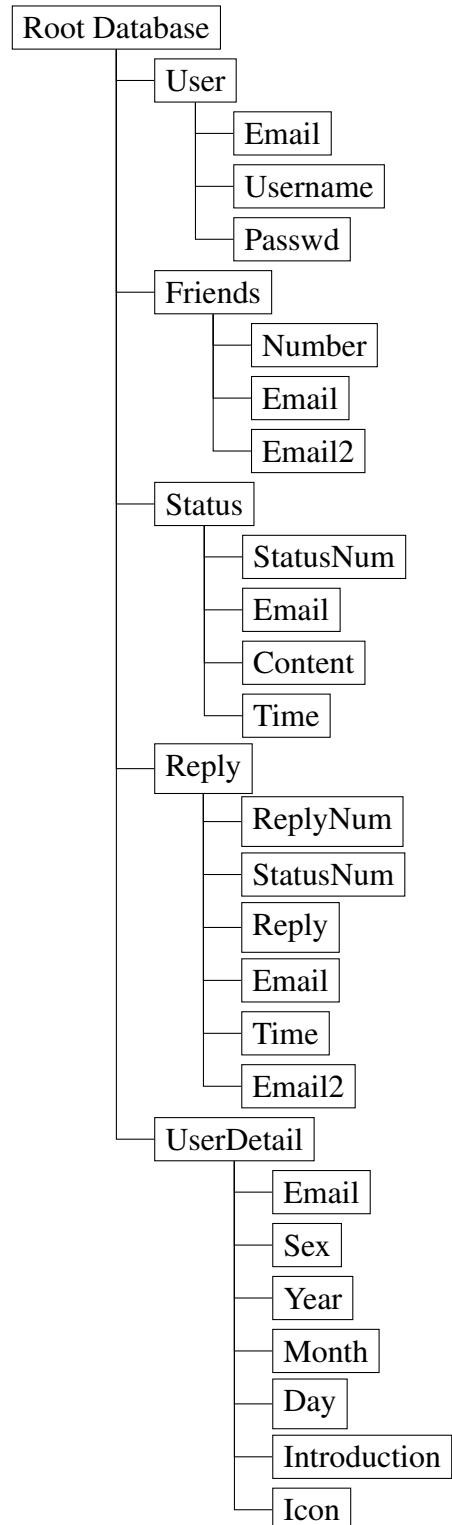
### 3.2 MySQL Table

According to the above ER Diagram, we've designed MySQL tables as below.

---

<sup>2</sup>This makes it possible to run static web pages on Apache and dynamic ones on Tomcat.

<sup>3</sup>MySQL is case insensitive for Windows and Mac OS, but that's not true for Linux.



**Figure 2:** MySQL Table Structure

---

Details about tables and attributes:

- User - deal with signing up & in
  - Email: primary key to identify users in registration and log-in
  - Username: nickname, which can be edited after registration
  - Passwd: password to validate a user
- Friends - record friend relationships
  - Number: auto-increment primary key for identification
  - Email: follower's email
  - Email2: followee's email
- Status - store posted statuses
  - StatusNum: auto-increment primary key for identification
  - Email: poster's email
  - Content: posted contents
  - Time: posting time
- Reply - store posted replies to status
  - ReplyNum: auto-increment primary key for identification
  - StatusNum: replied status number
  - Reply: reply contents
  - Email: replier's email
  - Time: replying time
  - Email2: repliee's email
- UserDetail - store user details
  - Email: primary & foreign key pointing to User.Email
  - Sex: user's sex
  - Year: user's year of birth
  - Month: user's month of birth

- 
- Day: user's day of birth
  - Introduction: simple introduction to the user
  - Icon: user's avatar

## 4 Division of Labor

Our group members:

Name	Student ID	Mobile	Email
He Yan	1400015464	15910670278	heyuan@pku.edu.cn
Sun Meng	1500012867	15010189739	1400017665@pku.edu.cn
Wu Chuchuan	1500062802	18811788416	wuchuchuan@pku.edu.cn

**Table 1:** Group Members

Division of labor:

- He Yan: add reCaptcha & add 2-step friend & write documentation
- Sun Meng: design of website appearance (CSS & image resources)
- Wu Chuchuan: website framework & database statement implementation

## 5 Kernel Codes

### 5.1 Google reCaptcha

```

1 <div class="g-recaptcha" data-sitekey="...">></div> // Client side
2 /* Server side, send post request to Google to verify */
3 String gRecaptchaResponse = request.getParameter("g-recaptcha-response");
4 String url = "https://www.google.com/recaptcha/api/siteverify";
5 String secret = "..."; // key for reCaptcha validation
6 boolean check = true; // whether reCaptcha passed
7 if (!(gRecaptchaResponse == null || "" .equals(gRecaptchaResponse))) {
8     try {
9         URL obj = new URL(url);
10        HttpsURLConnection con = (HttpsURLConnection) obj.openConnection();
11        con.setRequestMethod("POST");

```

---

```

12     con.setDoOutput(true);

13
14     String postParams = "secret=" + secret + "&response=" +
15         gRecaptchaResponse;
16     DataOutputStream wr = new DataOutputStream(con.getOutputStream());
17     wr.writeBytes(postParams);
18     wr.flush();
19     wr.close();

20     BufferedReader in = new BufferedReader(new InputStreamReader(con.
21         getInputStream()));
22     String inputLine;
23     StringBuffer rsps = new StringBuffer();
24     while ((inputLine = in.readLine()) != null) {
25         rsps.append(inputLine);
26     }
27     in.close();

28     JsonReader jsonReader = Json.createReader(new StringReader(rsps.
29         toString()));
30     JsonObject jsonObject = jsonReader.readObject();
31     jsonReader.close();
32     check=jsonObject.getBoolean("success");
33 } catch(Exception e) {
34     e.printStackTrace();
35 }
36 ...
37 if (check) { // only if reCaptcha verification passed
38     ...
39 }
```

## 5.2 Two-step friends

1	HashSet<String> frd1 = new HashSet<String>(); // one-step friends
2	/* select friends of own */
3	sq1 = "SELECT * FROM
4	'working'. 'friends' as a,
5	'working'. 'user' as b,
6	'working'. 'userdetail' as c

---

```

7 WHERE
8     a.email2 = c.email
9     and a.email2 = b.email
10    and a.email = '" + email + "'"; // email variable stores its own
11   email String
12 rs = stmt.executeQuery(sql);      // get result set
13 while (rs.next()) {
14     frd1.add(rs.getString("email2")); // store 1-step friends in frd1
15 }
16 ...
17 ... // output 1-step friends
18
19 HashMap<String , HashSet<String>> frd2 =
20     new HashMap<String , HashSet<String>>(); // two-step friends
21 for (String frd : frd1) { // for each 1-step friend
22     /* get the friends of 1-step friends */
23     sql = "SELECT * FROM
24         'working'.'friends' as a,
25         'working'.'user' as b,
26         'working'.'userdetail' as c
27     WHERE
28         a.email2 = c.email
29         and a.email2 = b.email
30         and a.email = '" + frd + "'";
31     rs = stmt.executeQuery(sql);
32
33     /* NOTE that a 2-step friend may have multiple intermediaries */
34     while (rs.next()) {
35         String frd2em1 = rs.getString("email2"); // 2-step friend email
36         if (!frd2.containsKey(frd2em1)) // if new, create a new entry
37             frd2.put(frd2em1, new HashSet<String>());
38         frd2.get(frd2em1).add(frd); // add intermediary to record set
39     }
40 }
41 frd2.keySet().removeAll(frd1); // exclude 1-step friends
42 frd2.keySet().remove(email); // exclude itself
43 /*
44 * Here you get emails of 2-step friends and all
45 * their intermediaries , so just select username
46 * from database with the emails and output
47 */
48 ...

```

---

## 6 Website Preview

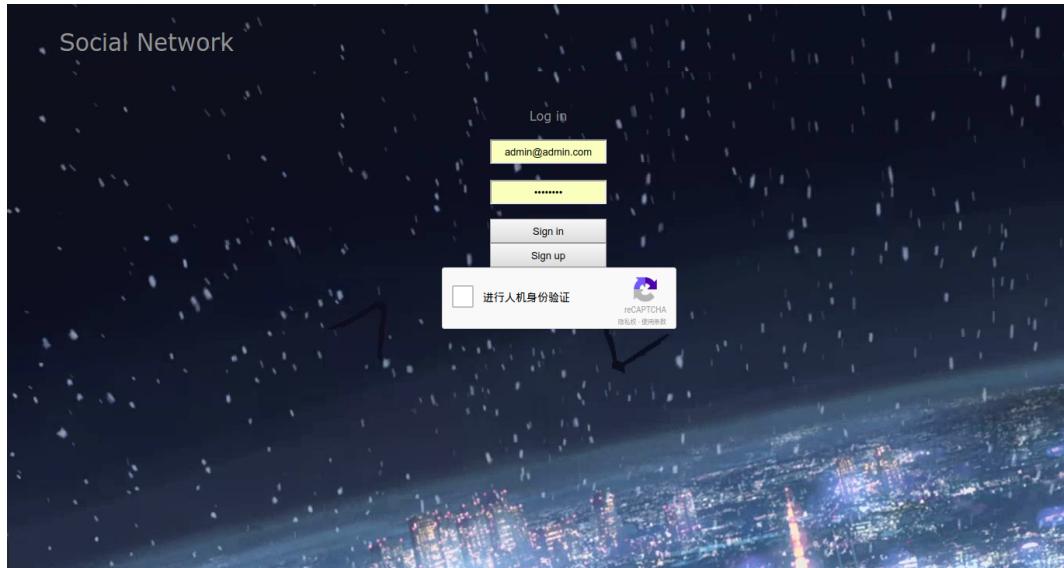


Figure 3: Log-in Interface

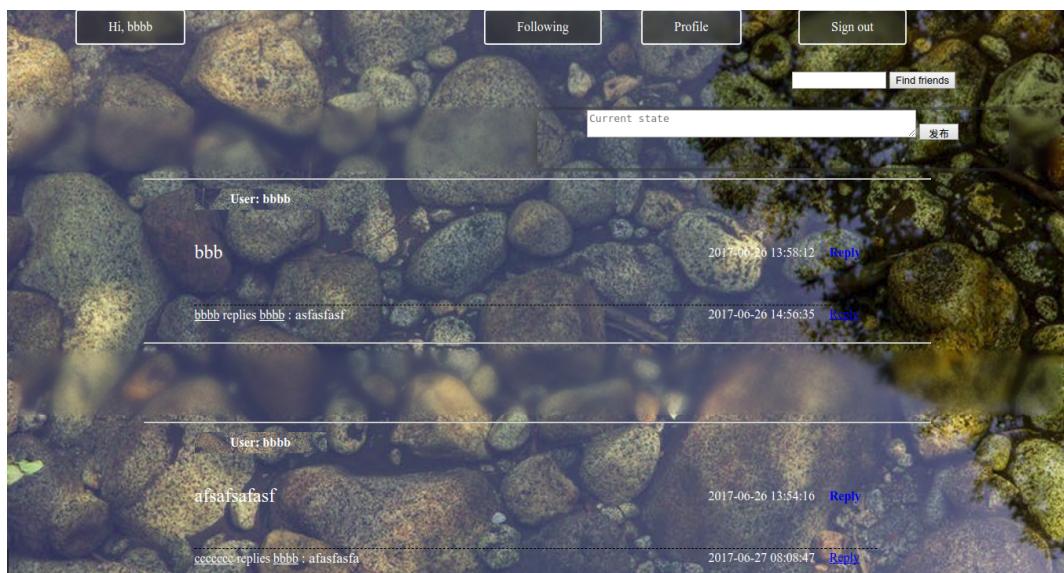
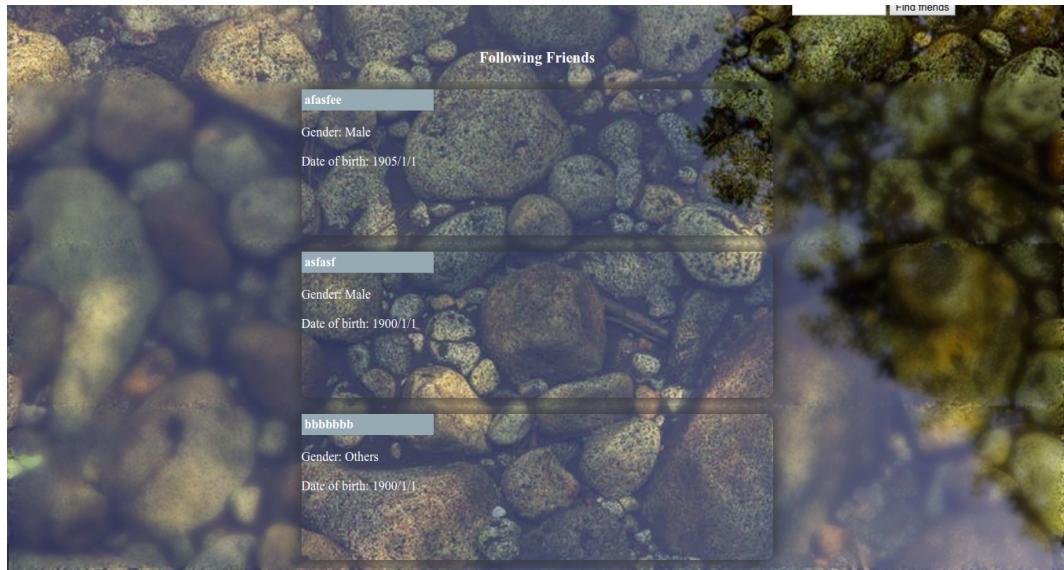
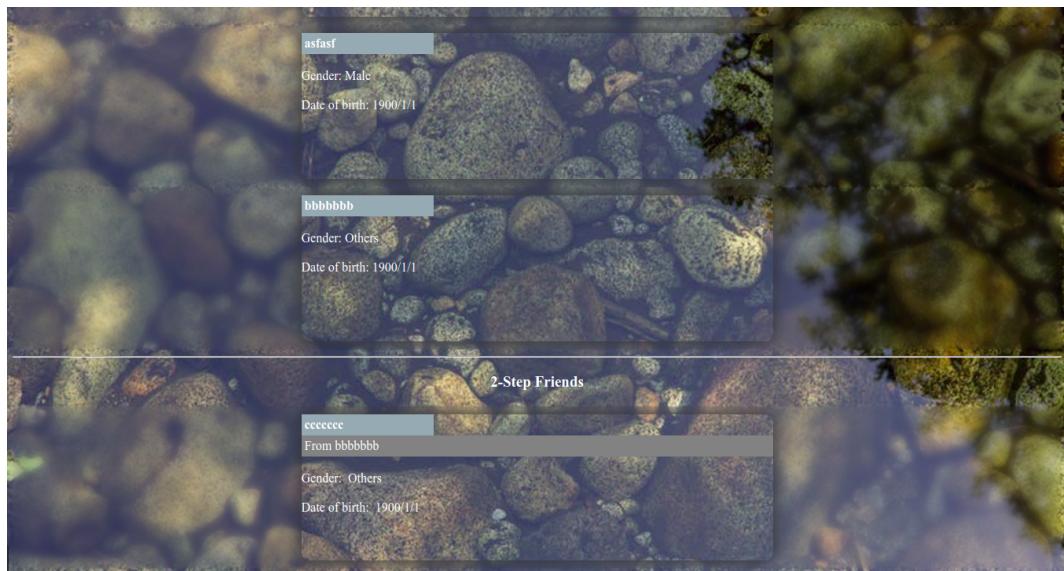


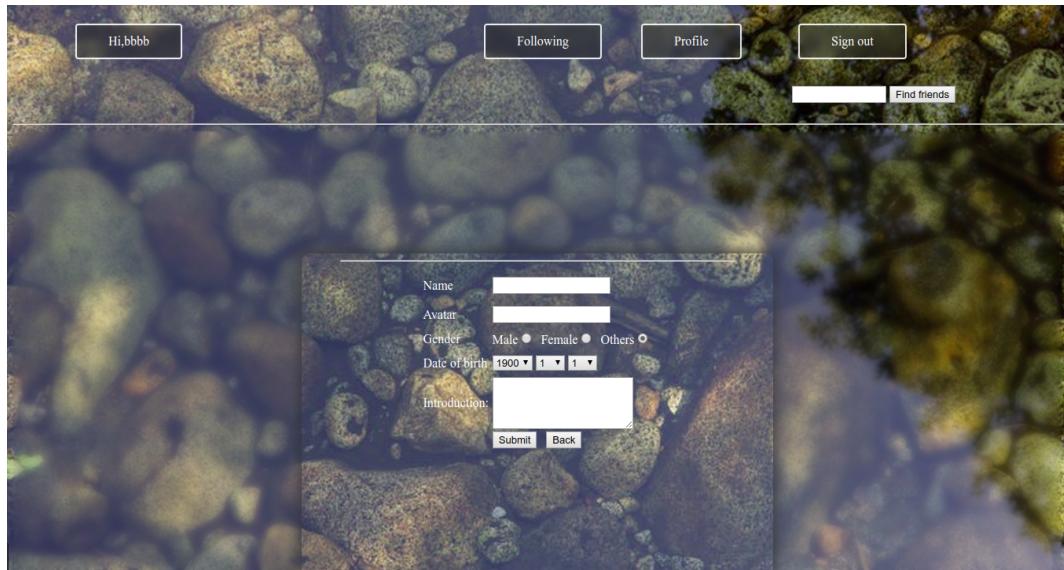
Figure 4: User Homepage



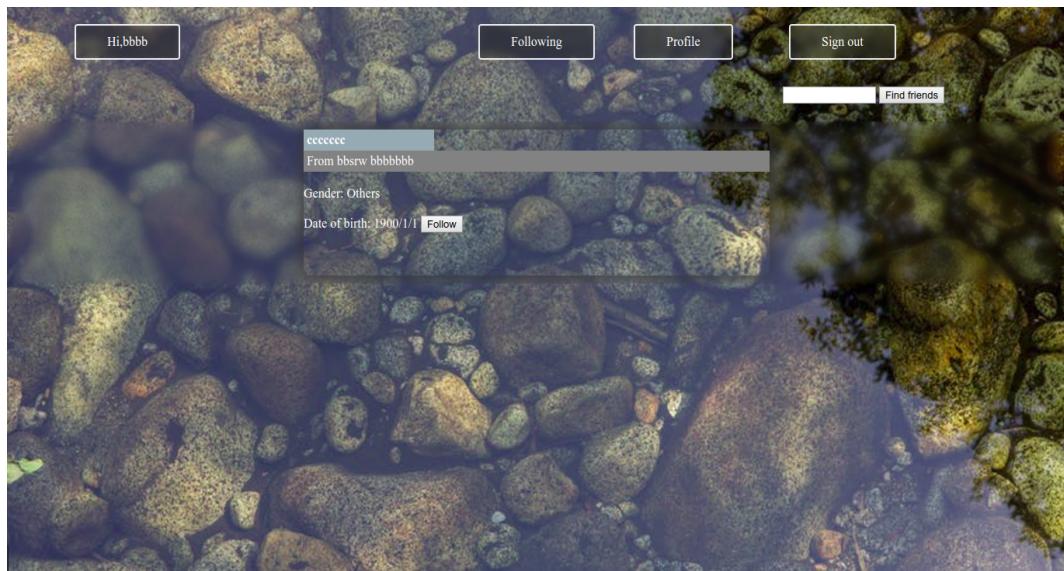
**Figure 5:** Following (1)



**Figure 6:** Following (2)



**Figure 7:** Profile



**Figure 8:** Search for Friends

---

## 7 References

- Guidebook, installers and demo provided at [course.pku.edu.cn](http://course.pku.edu.cn)
- L<sup>A</sup>T<sub>E</sub>X template provided by [Overleaf](#)
- [mysql-connector-java-5.1.42-bin.jar](#)
- [javax.json-api-1.1.jar](#)