DOCUMENTATION OF SOCIAL NETWORK PROJECT

June 29, 2017

He Yan

Contents

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[1] validation
  - Two-step friends (search name & display relation path)

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

[1] Completely Automated Public Turing test to tell Computers and Humans Apart

3. link Apache and Tomcat with Apache Tomcat Connector [2]

4. install MySQL[3] 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 LaTeX.

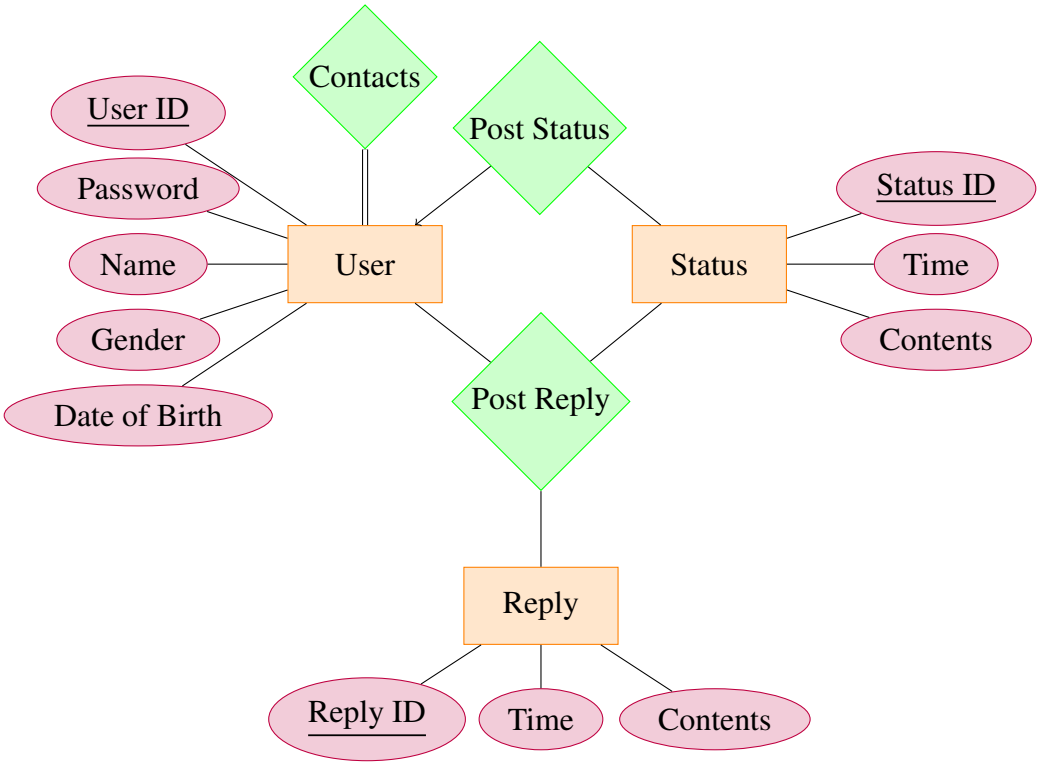

Figure 1: Entity-Relationship Diagram

3.2 MySQL Table

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

[2]This makes it possible to run static web pages on Apache and dynamic ones on Tomcat.
[3]MySQL is case insensitive for Windows and MacOS, but that's not true for Linux.

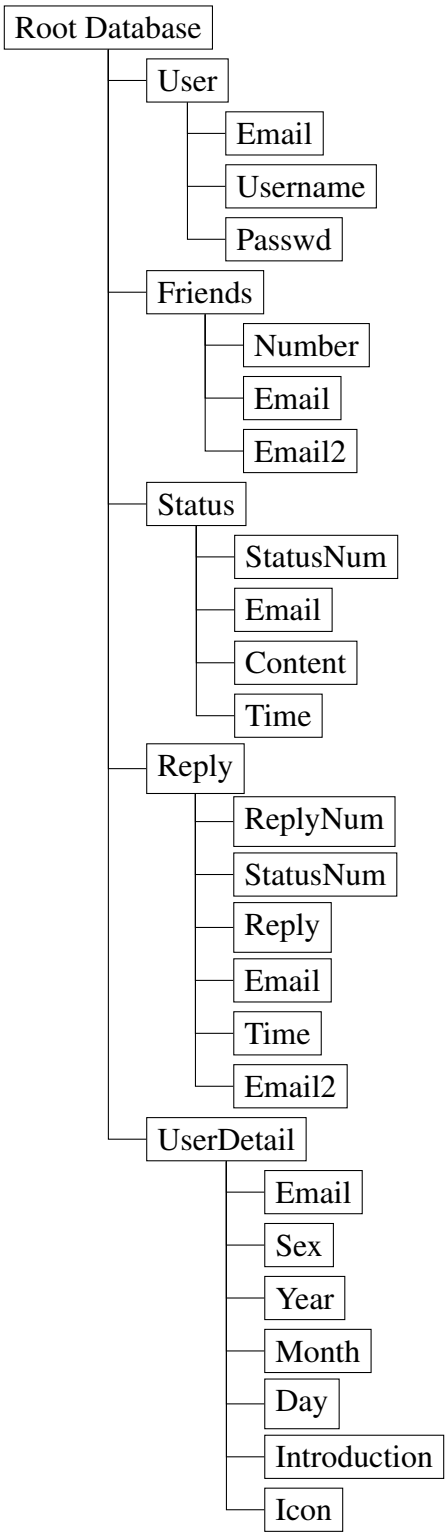

Figure 2: MySQL Table Structure

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	heyan@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

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

```
12          con.setDoOutput(true);
13
14          String postParams = "secret=" + secret + "&response=" +
                gRecaptchaResponse;
15          DataOutputStream wr = new DataOutputStream(con.getOutputStream());
16          wr.writeBytes(postParams);
17          wr.flush();
18          wr.close();
19
20          BufferedReader in = new BufferedReader(new InputStreamReader(con.
                getInputStream()));
21          String inputLine;
22          StringBuffer rsps = new StringBuffer();
23          while ((inputLine = in.readLine()) != null) {
24              rsps.append(inputLine);
25          }
26          in.close();
27
28          JsonReader jsonReader = Json.createReader(new StringReader(rsps.
                toString()));
29          JsonObject jsonObject = jsonReader.readObject();
30          jsonReader.close();
31          check=jsonObject.getBoolean("success");
32      } catch(Exception e) {
33          e.printStackTrace();
34      }
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  sql = "SELECT * FROM
4          `working`.`friends` as a,
5          `working`.`user` as b,
6          `working`.`userdetail` as c
```

```
    WHERE
        a.email2 = c.email
        and a.email2 = b.email
        and a.email = '" + email + "'"; // email variable stores its own
            email String
rs = stmt.executeQuery(sql);     // get result set
while (rs.next()) {
    frd1.add(rs.getString("email2"));    // store 1-step friends in frd1
}
... // output 1-step friends

HashMap<String, HashSet<String>> frd2 =
    new HashMap<String, HashSet<String>>(); // two-step friends
for (String frd : frd1) {    // for each 1-step friend
    /* get the friends of 1-step friends */
    sql = "SELECT * FROM
        `working`.`friends` as a,
        `working`.`user` as b,
        `working`.`userdetail` as c
        WHERE
            a.email2 = c.email
            and a.email2 = b.email
            and a.email = '" + frd + "'";
    rs = stmt.executeQuery(sql);

    /* NOTE that a 2-step friend may have multiple intermediaries */
    while (rs.next()) {
        String frd2eml = rs.getString("email2");     // 2-step friend email
        if(!frd2.containsKey(frd2eml))  // if new, create a new entry
            frd2.put(frd2eml, new HashSet<String>());
        frd2.get(frd2eml).add(frd); // add intermediary to record set
    }
}
frd2.keySet().removeAll(frd1);  // exclude 1-step friends
frd2.keySet().remove(email);     // exclude itself
/*
 * Here you get emails of 2-step friends and all
 * their intermediaries, so just select username
 * from database with the emails and output
 */
...
```

6 Website Preview

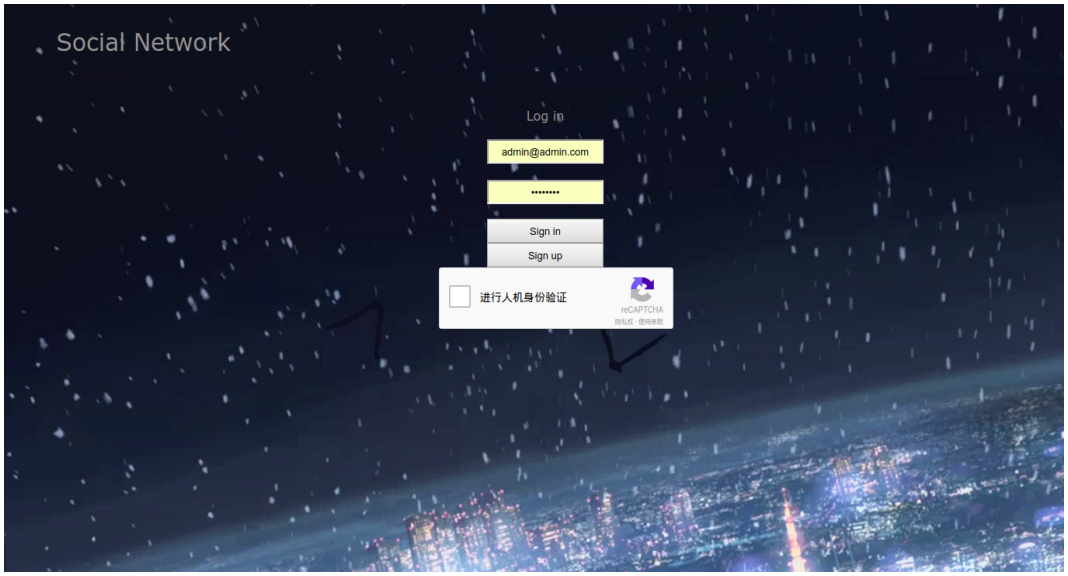

Figure 3: Log-in Interface

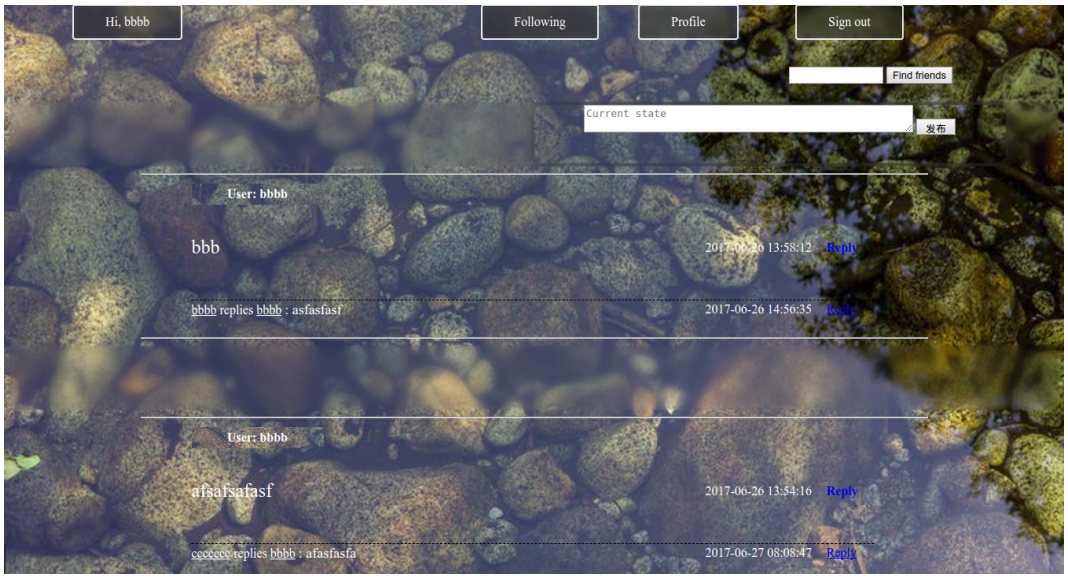

Figure 4: User Homepage

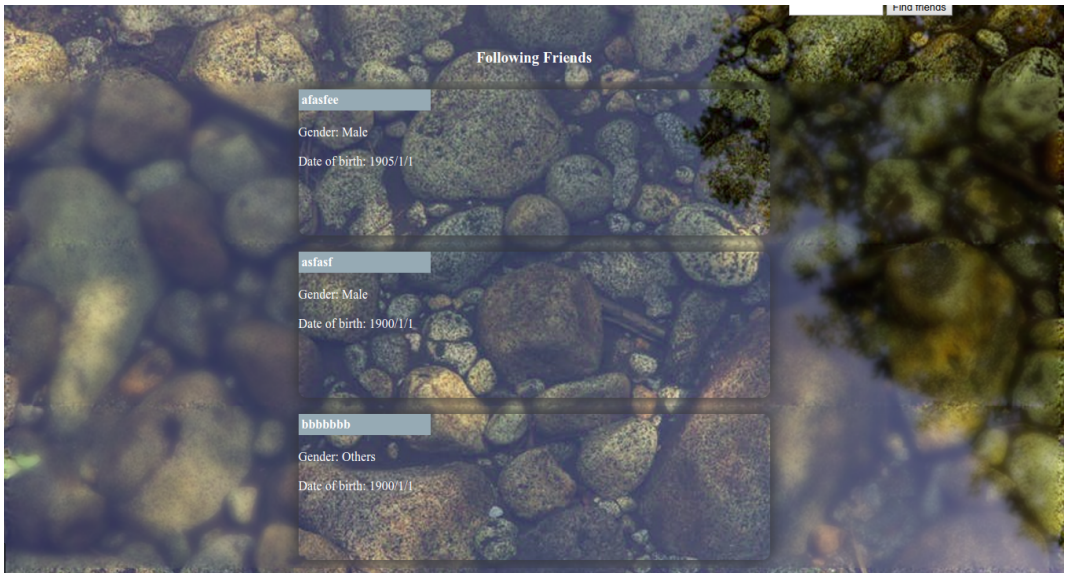

Figure 5: Following (1)

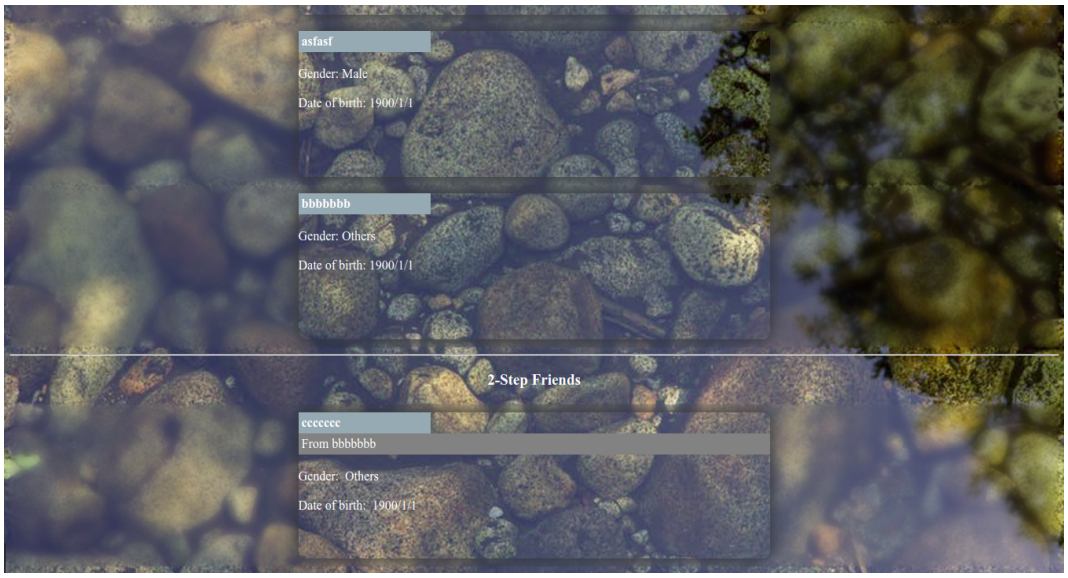

Figure 6: Following (2)

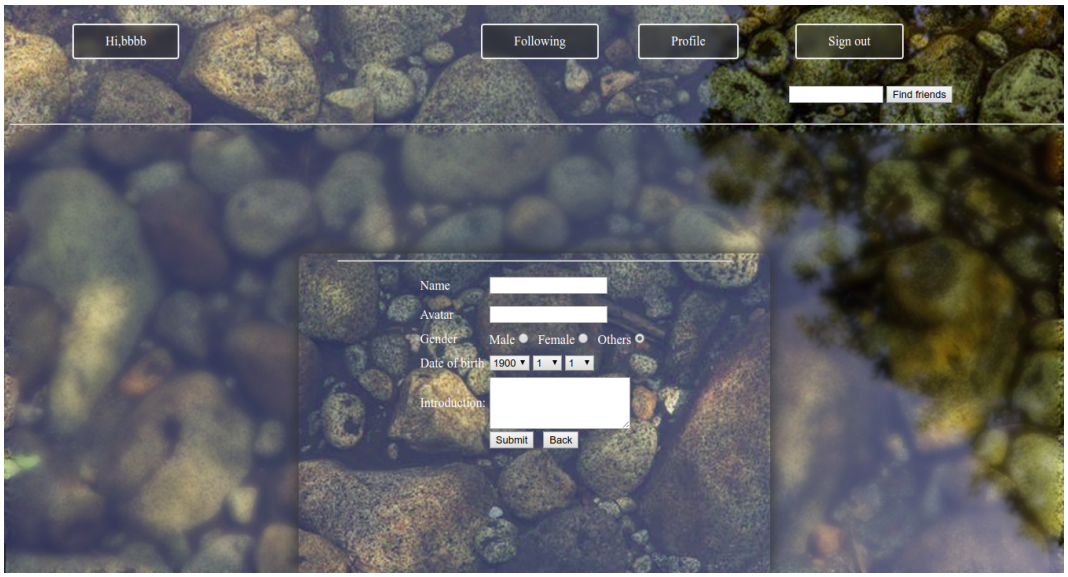

Figure 7: Profile

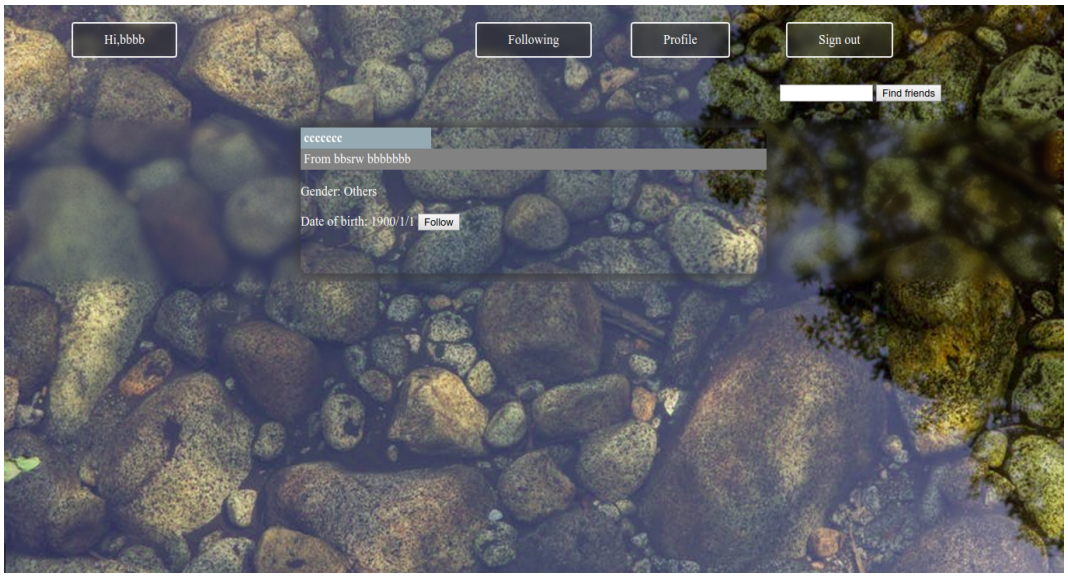

Figure 8: Search for Friends

7 References

- Guidebook, installers and demo provided at `course.pku.edu.cn`
- LaTeX template provided by Overleaf
- mysql-connector-java-5.1.42-bin.jar
- javax.json-api-1.1.jar