# DOCUMENTATION OF SOCIAL NETWORK PROJECT

June 27, 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

## 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.

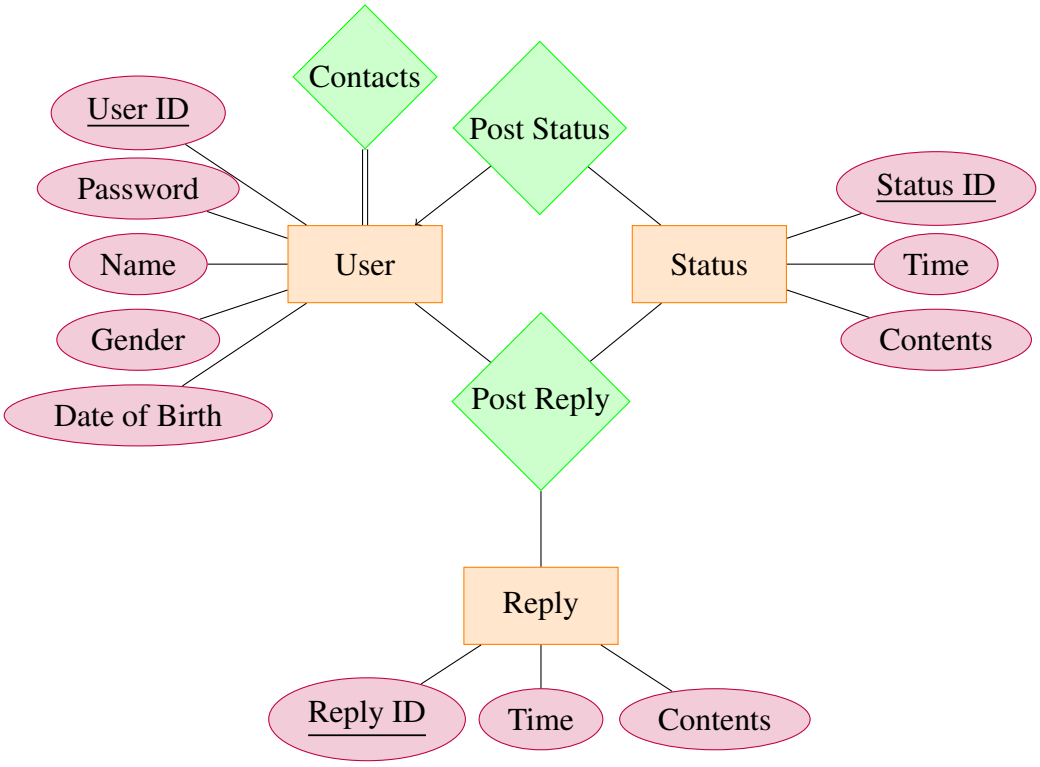


**Figure 1:** Entity-Relationship Diagram

## 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.

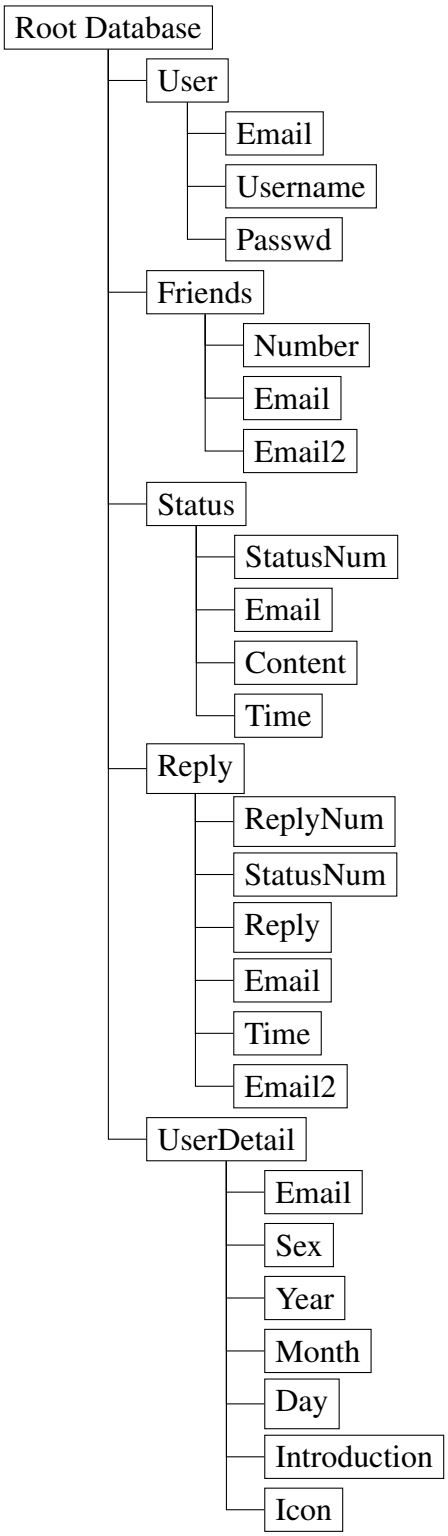


**Figure 2:** MySQL Table Structure

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

```
1  /* Client Side */
2  <div class="g-recaptcha" data-sitekey="6
       LdDNiMUAAAAAHDPfsdqqPKAPFEy5Xi3EoGwJIXi"></div>
3
4  /* Server Side */
5  String gRecaptchaResponse = request.getParameter("g-recaptcha-response");
6  String url = "https://www.google.com/recaptcha/api/siteverify";
7
8  /* Key for reCaptcha validation */
9  String secret = "6LdDNiMUAAAAACVxX9eQOV4ITsc9YApSRpb80Lle";
10
```

```
boolean check = true;
if (!(gRecaptchaResponse == null || "".equals(gRecaptchaResponse))) {
    try{
        URL obj = new URL(url);
        HttpsURLConnection con =
            (HttpsURLConnection) obj.openConnection();
        con.setRequestMethod("POST");
        con.setDoOutput(true);

        String postParams
            = "secret=" + secret + "&response=" + gRecaptchaResponse;

        DataOutputStream wr =
            new DataOutputStream(con.getOutputStream());
        wr.writeBytes(postParams);
        wr.flush();
        wr.close();

        BufferedReader in =
            new BufferedReader(
                new InputStreamReader(
                    con.getInputStream()));
        String inputLine;
        StringBuffer rsps = new StringBuffer();
        while ((inputLine = in.readLine()) != null) {
            rsps.append(inputLine);
        }
        in.close();

        JsonReader jsonReader =
            Json.createReader(
                new StringReader(rsps.toString()));
        JsonObject jsonObject = jsonReader.readObject();
        jsonReader.close();
        check=jsonObject.getBoolean("success");
    } catch(Exception e) {
        e.printStackTrace();
    }
}
...
if (check) {
    ...
```

```
}
```

## 5.2 Two-step friends

```
HashSet<String> frd1 = new HashSet<String>();    // one-step friends
HashSet<String> frd2 = new HashSet<String>();    // two-step friends

/* select friends of own */
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 = '" + 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
}

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);
    while (rs.next()) {
        frd2.add(rs.getString("email2"));    // store 2-step friends in frd2
    }
}

```

```
36 frd2.removeAll(frd1);    // remove 1-step friends in frd2
37 frd2.remove(email); // remove itself in frd2
38 ...
```

# 6 Website Preview

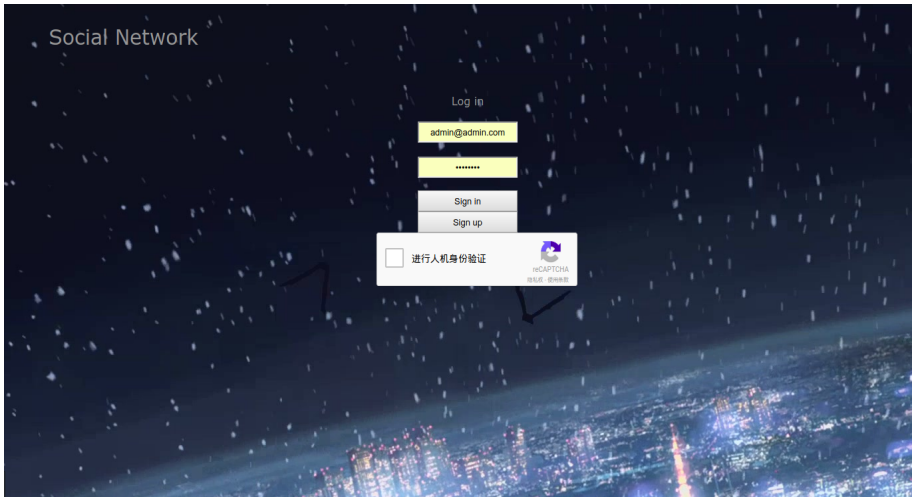


**Figure 3:** Log-in Interface

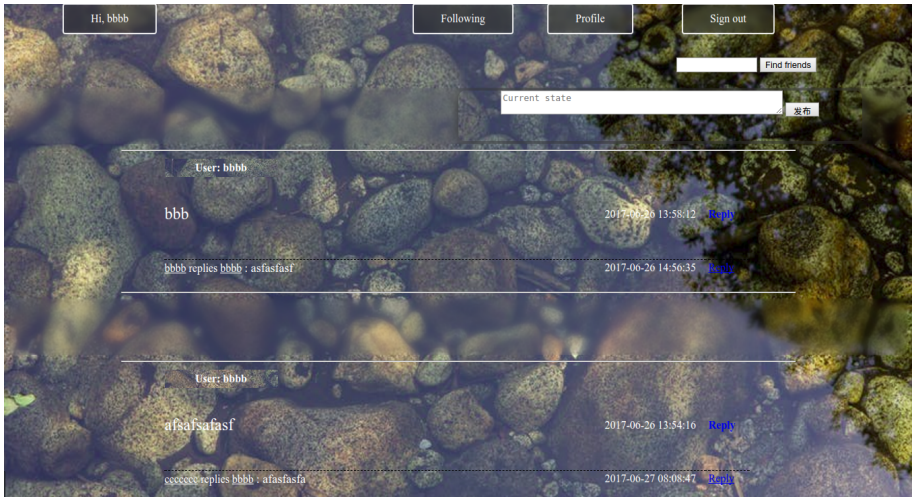


**Figure 4:** User Homepage

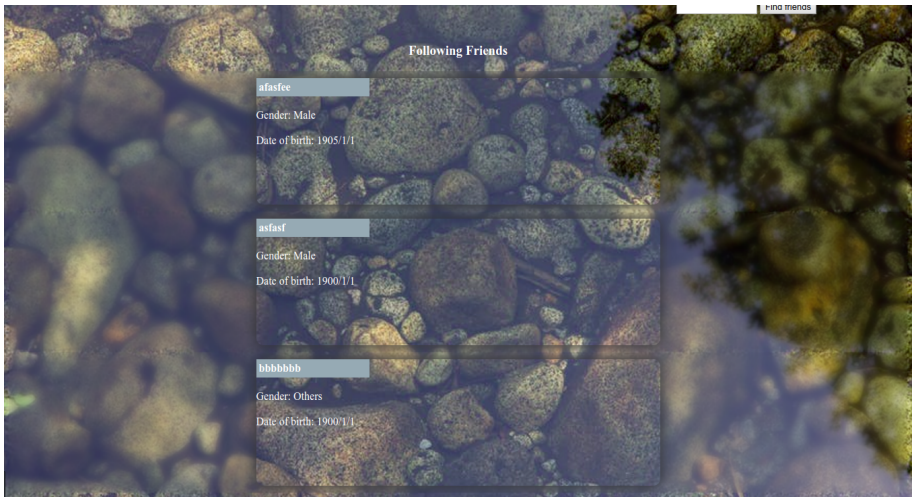


**Figure 5:** Following (1)

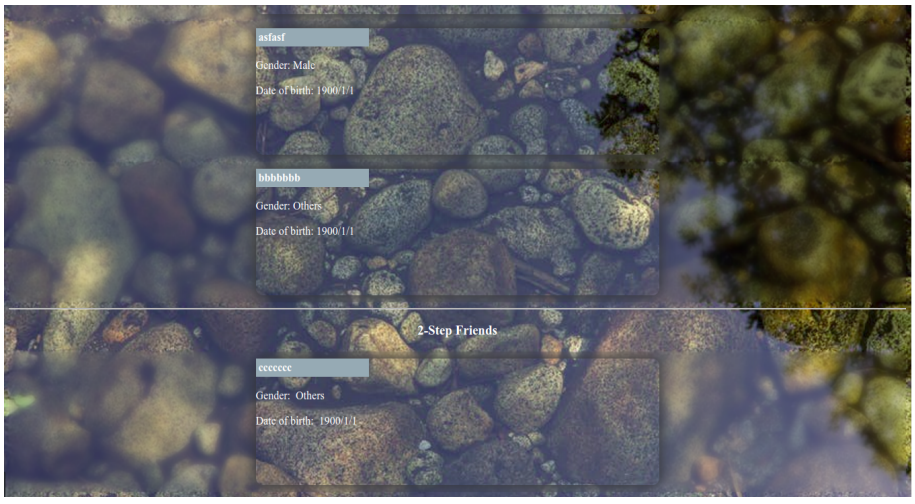


**Figure 6:** Following (2)

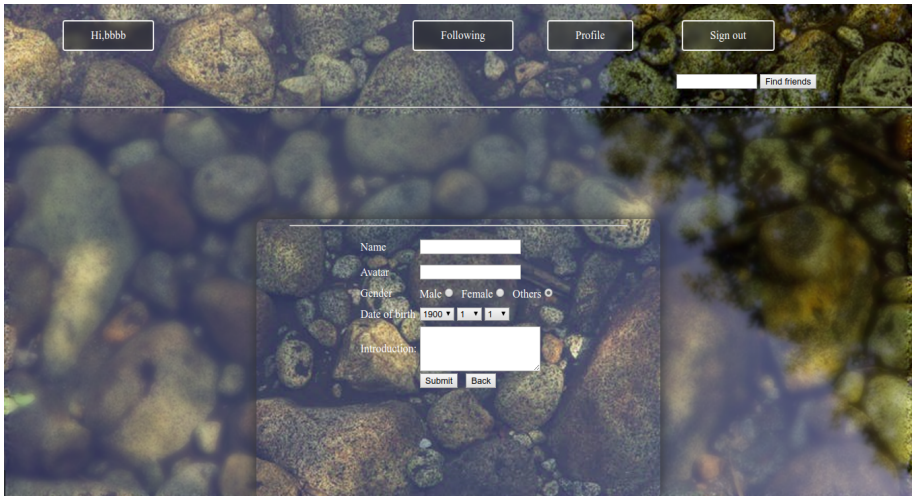


**Figure 7:** Profile

# 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