SCHOOL OF ENGINEERING AND APPLIED SCIENCE

Snail - Project Report for Cloud Computing

Yin Tan(yt2443) Diyue Xiao(dx2152)
Xianglu Kong(xk2122) Mengting Wu(mw2987)
May 24, 2015

1 Abstract

The prime goal for this project is to learn how to create a mobile application in Android. Also, based on what we have learned this semester, we want to enhance our knowledge in Cloud computing and Google Map API.

In this project, we developed a social network mobile application, *Snail*, which can connect people who share common interest in traveling by adding each other into friend list through the app. The highlight of it is that it can keep track of the routes of every tour as the user requests, similar to the trace behind a snail. Meanwhile, travelers can post onto app what they see and think along the journey while *Snail* will set a marker on the map at that location. When the destination is reached, the information along the journey will be saved as a record for this user into database so that users can see them next time they log in. By viewing other people's postings, travelers will have a more comprehensive knowledge about the journey they are going to face.

None of our team members is familiar with this platform or the topic. We met many difficulties during the project. Most problems are solved after some google search, while others need to be dealt with by searching for alternative solutions. It is much harder than we expected, but we learned a lot and had great fun.

2 Introduction

Android is an Google-developed open-source operating system running on mobile devices. It is primarily designed for touchscreen devices such as smartphones and tablet computers, and is so popular that most mobile app developers choose Android as their developing platform.

This offers us beginners a great deal of resources. We use Java to write inner logic and XML language to design layouts.

Facebook and Twitter are certainly the most popular network applications nowadays, and have domination in mobile market as well. They succeed because people have a desire to know more about what's happening around and to explore specific topics among their friends. These applications, however, are general and not focusing on any specific aspect of life. This gives rise to the boom of similar mobile apps which are designed for people with common interest and offer them a chance to share their stories and views more closely.

Snail is such a mobile app for traveling enthusiasts to form and expand their own group of friends. Users can comment on the routes they have been through, the hotels they once lived in, the restaurants they have enjoyed, and many other things. On the other hand, apart from purely connection between friends, travel agencies can take advantage of this app and approach a more potential client base. This would be the main source of revenue for *Snail* in the future.

The project can be conducted into two parts - front-end and back-end. The front-end is responsible for design of layouts and communication with device, while the back-end part takes care of interaction with Google Play servers as well as database. The details are explained in the next section.

3 Technical Implementation

3.1 Front-end

3.1.1 Login

In our app, every user has a username and a password. For a new user, he needs to firstly sign up and we will store the username and password entered. Then next time he can use this username and password to sign in. Each time an user tries to login, we will visit the database, check whether the database has the username or not, as well as whether the password is correct.

3.1.2 Page jump

We use intents to launch new activities and send messages across different pages. For example, when a user sends a moment on his timeline, the sending page will deliver user name as an identification to homepage so when the user go to homepage next time, the information is already updated with database.

3.1.3 Interact with device

It is absolutely likely that user might want to post a picture from device local disk or take a photo right away for uploading. Fortunately, there is a public class (MediaStore.Images.Media) in Android that support loading image from Gallery. As a result, our application enables users to select image from default Android Gallery application. They may also choose to take picture

with camera. The image will be saved as a file in local disk, and these resources are then uploaded to AWS EC2 for storage to be called.

We use intent with a flag indicating the source folder of image to launch an activity which will response to the selected photo. If user choose to take a photo at the moment, the bitmap stream will be converted into a file in the meantime and saved into gallery. Then the newly launched activity will send the file descriptor to database as a record.

3.2 Back-end

3.2.1 Database

In our project, we maintain a database with MySQL on an Amazon EC2 server. The Client communicates with this remote server via HTTP request, and a batch of php scripts act as the middleware between Client and MySQL Database. The database has five tables: the UserInfo table, the follow table, the moments table, the routePoints table and the routeStartEnd table.

1. UserInfo: This table is used for storing the email(username) and password of each user. The schema of this table is as the following. You can see that the primary key is email as we enforce that no two users can have the same username.

```
mysql> describe UserInfo;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| email    | varchar(30) | NO   | PRI |         |       |
| password | varchar(30) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
```

2. follow: This table is used for storing the friendship relationships of all users using this app. The schema of this table is as follows (the follower and followed are all users of this app):

```
mysql> describe follow;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| follower | varchar(30) | YES  |     | NULL    |       |
| followed | varchar(30) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
```

3. moments: This table is used to store moment information of all users of the app. The schema is as follows:

```
mysql> describe moments;
+---------------+--------------+------+-----+---------+-------+
| Field         | Type         | Null | Key | Default | Extra |
+---------------+--------------+------+-----+---------+-------+
| userId        | varchar(40)  | NO   |     |         |       |
| context       | varchar(200) | YES  |     | NULL    |       |
| time          | datetime     | YES  |     | NULL    |       |
| latitude      | double       | YES  |     | NULL    |       |
| longtitude    | double       | YES  |     | NULL    |       |
| imageLocation | varchar(100) | YES  |     | NULL    |       |
| routeID       | bigint(20)   | YES  |     | NULL    |       |
+---------------+--------------+------+-----+---------+-------+
```

4. routePoints: This table stores the (latitude,longitude) pairs of all the routes. We enforce that there are no same location pairs for one route. The schema of this table is as follows:

```
mysql> describe routePoints;
+-----------+------------+------+-----+---------+-------+
| Field     | Type       | Null | Key | Default | Extra |
+-----------+------------+------+-----+---------+-------+
| routeID   | bigint(20) | NO   | PRI | 0       |       |
| latitude  | double     | NO   | PRI | 0       |       |
| longitude | double     | NO   | PRI | 0       |       |
+-----------+------------+------+-----+---------+-------+
```

5. routeStartEnd: This table stores the start location and end location of each route. In this table, the routeID is the primary key, and is auto-incremented when insertion happens. Whenever a user begins tracking his/her route, we insert his/her intended start and end locations into this table, auto-increment the routeID, and return the routeID to user for following usage.

```
mysql> describe routeStartEnd;
+----------+-------------+------+-----+---------+----------------+
| Field    | Type        | Null | Key | Default | Extra          |
+----------+-------------+------+-----+---------+----------------+
| routeID  | bigint(20)  | NO   | PRI | NULL    | auto_increment |
| sLatt    | double      | YES  |     | NULL    |                |
| sLong    | double      | YES  |     | NULL    |                |
| eLatt    | double      | YES  |     | NULL    |                |
| eLong    | double      | YES  |     | NULL    |                |
| userName | varchar(50) | YES  |     | NULL    |                |
| count    | int(11)     | YES  |     | NULL    |                |
+----------+-------------+------+-----+---------+----------------+
```

For the android clients to communicate with database, the app has to make HTTP requests towards the server. In order not to block the main UI thread, we do this with the Asynctask class and run it as a background thread. Following is a illustrating piece of code on the Android Client:

```
class AttemptInsertPosition extends AsyncTask<ArrayList<String>, String, String> {

    @Override
    protected String doInBackground(ArrayList<String>... passing) {
        ArrayList<String> passed = passing[0];
        try {
            List<NameValuePair> params = new ArrayList<~>();
            params.add(new BasicNameValuePair("routeID", passed.get(0)));
            params.add(new BasicNameValuePair("latt", passed.get(1)));
            params.add(new BasicNameValuePair("long", passed.get(2)));
            JSONObject json = jParser.makeHttpRequest(addPosURL, "GET", params);
            return json.getString(TAG_MESSAGE) + "~";
        } catch (JSONException e) {
            e.printStackTrace();
        }
        return null;
    }

    protected void onPostExecute(String message) {
        if (message != null) {
            Log.d("message",message);
        }
    }
}
```

The following is a piece of code on the EC2 server to handle the corresponding HTTP request:

```
<?php
mysql_connect("localhost","shuai","123");
$db= mysql_select_db("app");
$routeID = $_GET["routeID"];
$latt = $_GET["latt"];
$long = $_GET["long"];
if (!empty($_GET)) {
        if (empty($_GET["routeID"]) || empty($_GET["latt"]) || empty($_GET["long"])) {
                $response["success"] = 0;
                $response["message"] = "One or both of the fields are empty .";
                die(json_encode($response));
        } else {
                $sql = "insert into routePoints values ('$routeID', '$latt', '$long')";
                if (mysql_query($sql) == TRUE) {
                        $response["success"] = 1;
                        $response["message"] = "the location has been inserted.";
                        die(json_encode($response));
                } else {
                        $response["success"] = 0;
                        $response["message"] = "insertion failed";
                        die(json_encode($response));
                }
        }
} else {
        $response["success"] = 0;
        $response["message"] = "empty get";
        die(json_encode($response));
}
mysql_close();
?>
```

3.2.2 Google Map

We used google map to show user's location with photo, and track their locations in time, so users will know their routes, as well as his friends' routes in *Snail*.

1. Get current location
   we use CurLocaTracker component to get user's current location. In this component, we use google's LocationService's FusedLocationApi to get user's latest location, and add marker to the map.

2. Draw trace behind a moving marker
   we use LocaChangeTracker to track user's current location in time. We just connect google map API's locationlistener, and use callback function to collect the locations and show them as polyline on google map.

3. Get recommended route from starting point to destination
   To recommend user's several great routes for him to follow, we design a recommendation algorithm. Assume that user's route is u, and another candidate route is v. u's start position latitude and longtitude are $sl(u)$ and $sg(u)$ respectively, and u's end position's latitude and longitude is $el(u)$ and $eg(u)$. Besides, the count of posted moments on route u is $c(u)$, and the count of posted moments on route v is $c(v)$. Then, the algorithm to calculate the difference D of these two routes is shown as the following:

   $$D = \alpha * ((sl(u)-sl(v))^2+(sg(u)-sg(v))^2+(el(u)-el(v))^2+(eg(u)-eg(v))^2)+\beta*(|c(v)-c(u)|)$$

   In this case, we always choose three candidate routes who have smaller D values, and recommend one to the user.

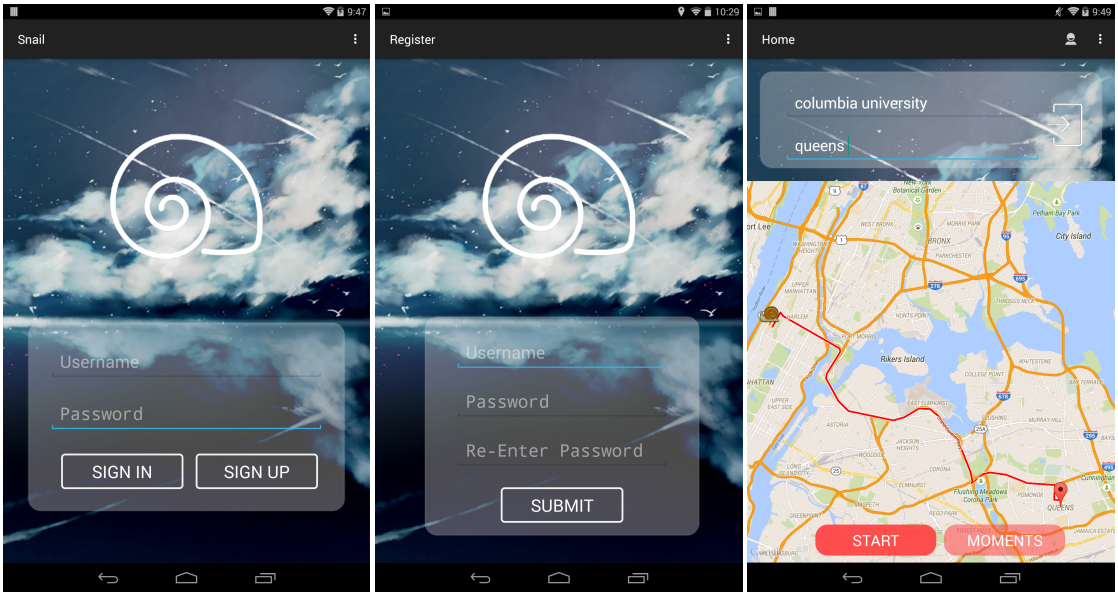
4. Display old routes and friends' previous moments
   To show user's old routes, we store the path and previous moments into database, and some recent path into data structures.

3.3 Highlight

The highlight of our project is idea: We try to track the routes of each member, and track the photos and texts of each route. Moreover, we let users share the trip routes with each other. And we can recommend otherâĂŹs routes to an user according to his/her intended start location and destination.

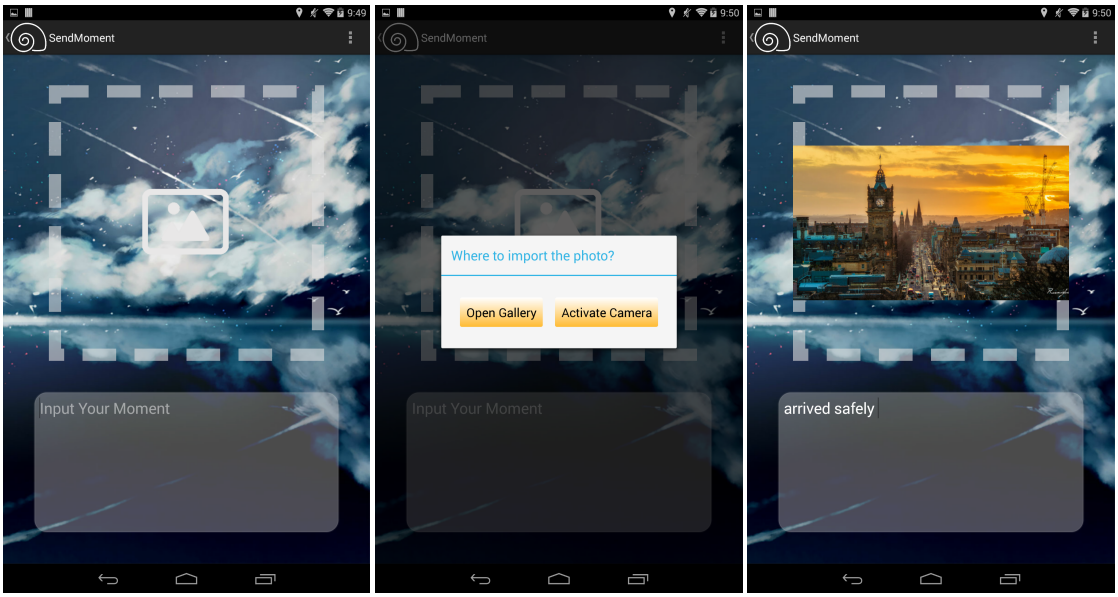
4 Result

Below are some of the screen shots of our application. After logging in, the user needs to input the starting point and destination to activate the tracker. Then he can click on START button

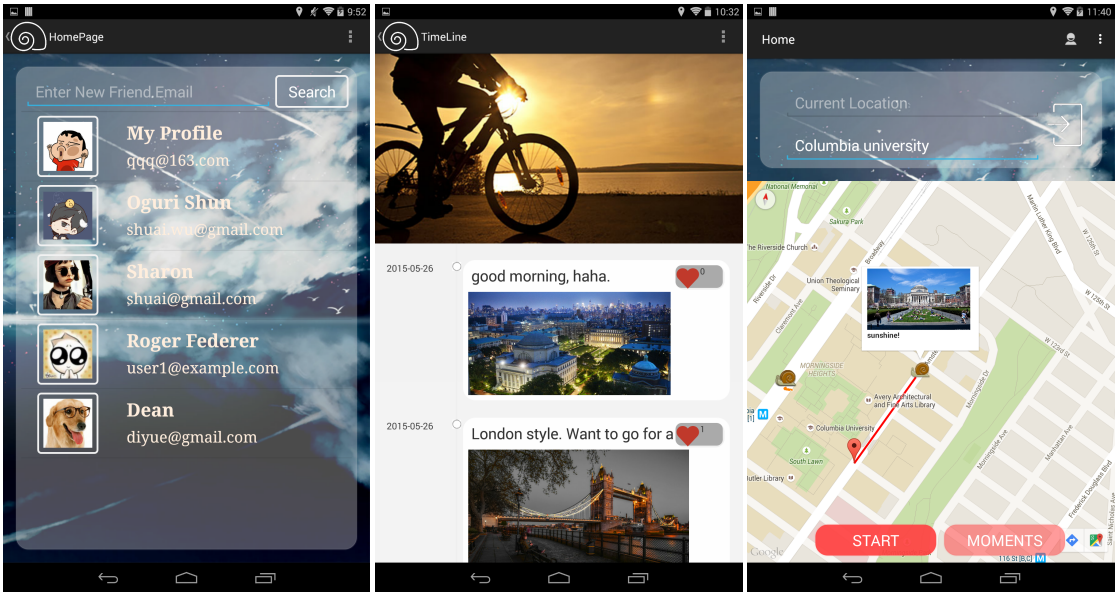


so that a snail icon will appear on the map indicating the current position, which then leaves a trace behind it to record the route.

Along the journey, the user can send moments including pictures and texts to the map. He can also choose to take a photo on the spot and upload it directly.



This new information will be updated on his personal timeline as well as on the map. So when he click the marker, the new post will pop up.



5 Lessons learned

- Lessons learnt as a team

  Working as a team is different to working with oneself. On the one hand, cooperating can be inspiring and promote efficiency. On the other hand, it also introduces some issues. For example, we need to arrange time to accommodate everyoneâĂŹs time schedule. And sometimes we have different ideas about one thing so we need to communicate to decide the best idea. Through this project, we not only learned how to build a real Android app, but also learned how to work as a team to reach a common goal.

  In this project, we have used several development tools and platforms to help us cooperate better. The first is the git. we use git to track project versions, and merge our code together. This greatly speed up our working efficiency. The second is the google doc, we share important documents and articles through google doc. This is very convenient and saved us a lot of time instead of always have to meet in person.

- Lessons learnt as a member

  Diyue Xiao: this is the first time for me to develop a Android app. It is hard since we have to start from the scratch in limited time and have confronted many problems during development. It is also very fun, since we have finally create an app of our ideas.

  Mengting Wu: I have learned how to develop an Android application and how to make an amazing User Interface. And when we do this project, we have used many techniques we learned in the cloud class, such as using EC2, RDS, etc.

  Ying Tan: This is my first time to touch android application, and I learned so much about google map API and how it will be used in android environment. Also, we applied

cloud computing techniques in this project such as RDS and EC2 and combined some recommendation algorithms in this project. It is really a great experience to learn so much about full-stack development. Moreover, I learned that timeline is important, and we should always do planning at first !

Xianglu Kong: I really treasure this experience to build a real Android app with a group of very talented students. I learned the overall structure of Android app as well as many tricks to design a good layout. And I begin to understand why some mobile apps are designed as they are.

6 Future Work

Our application is for tourism pals. Thus, our goal is to make the best service for trip friends to share, communication, and enjoy their trip better.

In the future, we plan to extend our project to support 1. team route, where a team of people can join together as a group, and they can post photos on the same route. 2. peer-to-peer chat, which is for people to communicate more conveniently through our app; and 3. group chat, which is for a team of people to communicate more efficiently when they go out together.

Moreover, we will also refine the details of our app, such as improve the UI design, add more interactive functionality to the timeline, etc.