MD Kabir
Jason Rosenstein

Part 1

Introduction

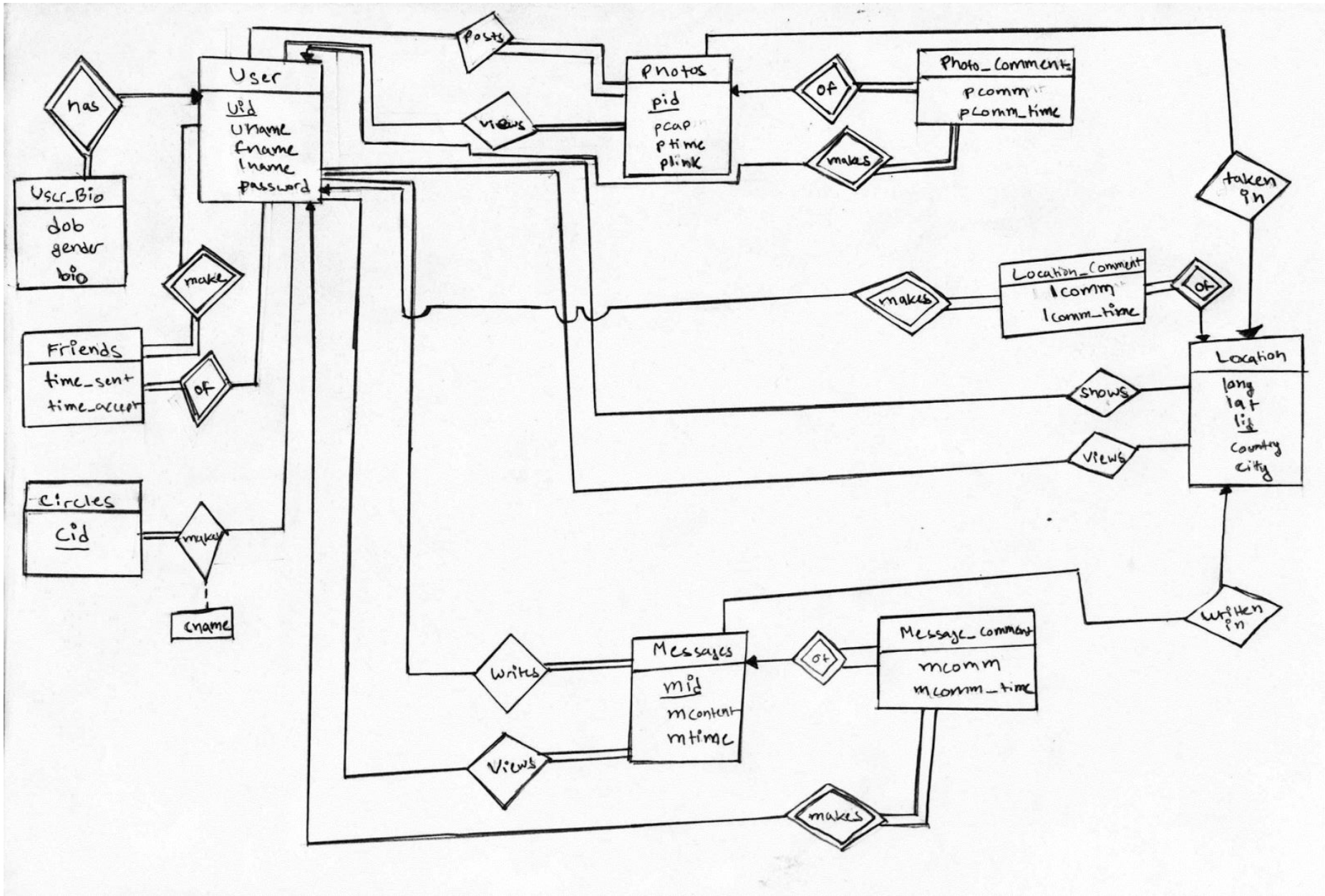
Tourini -> Social Networking Site
-> Traveling Journal
-> Travel oriented photo sharing

People/Users -> can post photos with optional caption, the location and the time
-> can make diary entries with the location and time
-> can post their location
-> can select who sees their posts
-> connect and get advice from other travellers (users)

Entity Relational Model:

The entity relational model we designed is in the following page:

Assumptions:
1) A User is his/her own friend when he/she registers for the site
2) Users can comment on each others location posts, diary entries, or pictures. But, they do not have the ability to message each other.
3) Users are required to enter their first and last name when registering
4) If user provide optional bio, then they must provide their date of birth, short bio, and their gender


User
Uid
Uname
fname
lname
password
has
User_Bio
dob
gender
bio
Posts
views
Photos
pid
pcap
ptime
plink
of
Photo_Comment
pcomm
pcomm_time
makes
taken in
make
Friends
time_sent
time_accept
of
makes
Location_Comment
lcomm
lcomm_time
of
Location
long
lat
lid
country
city
shows
views
Circles
Cid
make
cname
Messages
mid
mcontent
mtime
writes
views
of
Message_Comment
mcomm
mcomm_time
makes
written in

Relational Model:

User(uid, uname, fname, lname, password)
-> uid is the primary key
-> this table is used to keep track of a user's basic information
-> uname stands for username
-> fname stands for first name
-> lname stands for last name
-> users enter username and password to login

User_Bio(uid, dob, gender, bio)
-> this table is used to store user's optional bio that he/she can enter
-> uid is the foreign key that references User
-> dob is date of birth
-> bio is a block of text the user writes to describe himself/herself

Friends(uid1, uid2, time_sent, time_accept)
-> this table is used to keep track of who is friends and friend requests
-> uid1 is a foreign key that references User
-> uid2 is a foreign key that references User
-> time_sent is when the friend request was sent
-> time_accept is when the friend request was accepted,
-> if time_accept null then the friend request was not accepted

Circle_Name(cid, cname)
-> this table is used to track of the names of the circles
-> cid is the primary key

Circle(cid, uid)
-> this table is used to keep track of members of a circle
-> cid is the foreign key that references Circle_Name
-> uid is the foreign key that references User

Location(lid, long, lat, country, city)
-> this table is used to keep record of geo locations
-> lid is the primary key
-> long is the longitude
-> lat is the latitude

Attraction(lid, aname)
-> this table is used to keep a record of attractions

-> lid is the foreign key that references Location

User_Location(lid, uid, ltime, lpid)
-> this table is used to keep a record of locations users post on the site
-> lid is the foreign key that references Location
-> uid is the foreign key that references User
-> ltime is the time the user select or post the location info
-> lpid is the primary key

Location_Access(lpid, uid)
-> this table is used to keep record of who can see the location post made by a user
-> lpid is the foreign key that references User_Location
-> uid is the foreign key that references User

Location_Comment(lpid, uid, lcomm, lcomm_time)
-> this table is used to track all the comments made for a location post
-> lpid is the foreign key that references User_Location
-> uid is the foreign key that references User
-> lcomm is the comment made by a user
-> ltime is the time the comment is made

Photos(pid, pcap, ptime, lid, plink)
-> this table is used to track all the photos posted on the site
-> pid is the primary key
-> pcap is the optional caption users can add to photos
-> ptime is the time the picture was posted or user sets for the picture
-> plink is the link of the image, which will be stores using imgur api
-> lid is the foreign key that references Location

User_Photo(pid, uid)
-> this table is used to track pictures posted by a user
-> pid is the foreign key that references Photos
-> uid is the foreign key that references User

Photo_Access(pid, uid)
-> this table is used to keep record of who can see pictures posted by a user
-> pid is the foreign key that references Photos
-> uid is the foreign key that references User

Photo_Comment(pid, uid, pcomm, pcomm_time)
-> this table is used to track all the comments made for a picture post
-> pid is the foreign key that references Photos
-> uid is the foreign key that references User

-> pcomm is the comment made by a user
-> pcomm_time is the time the comment is made

Messages(<u>mid</u>, mcontent, mtime, lid)
-> this table is to keep track of all the diary entries made on the site
-> mid is the primary key
-> mcontent is the content of the diary entry made
-> mtime is the time the entry was made
-> lid is a foreign key that references Location

User_Message(<u>mid</u>, <u>uid</u>)
-> this table is used to track diary entries made by a user
-> mid is the foreign key that references Messages
-> uid is the foreign key that references User

User_Access(<u>mid</u>, <u>uid</u>)
-> this table is used to keep record of who can see diary entries posted by a user
-> mid is the foreign key that references Messages
-> uid is the foreign key that references User

Message_Comment(<u>mid</u>, <u>uid</u>, mcomm, mcomm_time)
-> this table is used to track all the comments made for a diary entry
-> mid is the foreign key that references Messages
-> uid is the foreign key that references User
-> mcomm is the comment made by a user
-> mcomm_time is the time the comment is made

<u>Explanations:</u>

The User and the User_Bio table are basically to keep track of User's information and their bio if they decide to enter that. If a user does decide to enter optional bio then they must enter their date of birth, short bio, and gender, so these info cannot be NULL in the User_Bio table. The Friends table has two user ids. In this table there will be multiple tuples where the two user ids will be the same, because our design shows that a user is his/her own friend too. It is sort of like Facebook messenger, where you can message yourself. The time_accept column in the Friends table shows if a friend request has been accepted or not. If it is NULL then the friend request between the two users was not accepted. The Access Tables, which are Location_Acess, Photo_Acess, Message_Acess, these tables are used to implement the part where the user gets to chose who gets to see their entries or photos or location posts. It will have the item and the ids of people who can see that item, whether it is a message, or photo, or location post. The Circle table also has similar format. It has a circle id to specify what circle it is

and all the user ids for users who are part of the circle. The comment tables, Photo_Comment, Location_Comment, Message_Comment are all tables that track comments made by users for each entities. The other tables are standard tables needed to keep track of basic information given the website specifications.

Creating and Setting Up the Database

Create the database:

```
mysql> CREATE DATABASE Tourini;
```

Create User table:

```
mysql> CREATE TABLE User(
    -> uid INT UNSIGNED PRIMARY KEY AUTO_INCREMENT,
    -> uname VARCHAR(25) NOT NULL,
    -> password VARCHAR(25) NOT NULL,
    -> fname VARCHAR(35) NOT NULL,
    -> lname VARCHAR(35) NOT NULL);
Query OK, 0 rows affected (1.06 sec)
```

Create User_Bio table:

```
mysql> CREATE TABLE User_Bio(
    -> uid INT UNSIGNED NOT NULL,
    -> dob DATE NOT NULL,
    -> bio VARCHAR(140) NOT NULL,
    -> gender CHAR(1) NOT NULL,
    -> FOREIGN KEY (uid) REFERENCES User(uid) );
Query OK, 0 rows affected (0.81 sec)
```

Create Friends table:

```
mysql> CREATE TABLE Friends(
    -> uid1 INT UNSIGNED NOT NULL,
    -> uid2 INT UNSIGNED NOT NULL,
    -> time_sent DATETIME NOT NULL,
    -> time_accept DATETIME,
    -> FOREIGN KEY(uid1) REFERENCES User(uid),
    -> FOREIGN KEY(uid2) REFERENCES User(uid) );
Query OK, 0 rows affected (0.45 sec)
```

Create Circle_Name table:

```
mysql> CREATE TABLE Circle_Name(
    -> cid INT UNSIGNED NOT NULL PRIMARY KEY AUTO_INCREMENT,
    -> cname VARCHAR(40) NOT NULL);
Query OK, 0 rows affected (0.63 sec)
```

Create Circle table:

```
mysql> CREATE TABLE Circle(
    -> cid INT UNSIGNED NOT NULL,
    -> uid INT UNSIGNED NOT NULL,
    -> FOREIGN KEY(cid) REFERENCES Circle_Name(cid),
    -> FOREIGN KEY(uid) REFERENCES User(uid) );
Query OK, 0 rows affected (1.06 sec)
```

Create Location table:

```
mysql> CREATE TABLE Location(
    -> lid INT UNSIGNED NOT NULL PRIMARY KEY AUTO_INCREMENT,
    -> longitude FLOAT(11,8) NOT NULL,
    -> latitude FLOAT(11,8) NOT NULL,
    -> city VARCHAR(35) NOT NULL,
    -> country VARCHAR(35) NOT NULL );
Query OK, 0 rows affected (0.54 sec)
```

Create Attraction table:

```
mysql> CREATE TABLE Attraction(
    -> lid INT UNSIGNED NOT NULL,
    -> aname VARCHAR(40) NOT NULL,
    -> FOREIGN KEY(lid) REFERENCES Location(lid) );
Query OK, 0 rows affected (0.39 sec)
```

Create User_Location table:

```
mysql> CREATE TABLE User_Location(
    -> lpid INT UNSIGNED NOT NULL PRIMARY KEY AUTO_INCREMENT,
    -> lid INT UNSIGNED NOT NULL,
    -> uid INT UNSIGNED NOT NULL,
    -> ltime DATETIME NOT NULL,
    -> FOREIGN KEY(lid) REFERENCES Location(lid),
    -> FOREIGN KEY(uid) REFERENCES User(uid) );
Query OK, 0 rows affected (0.79 sec)
```

Create Location_Access table:

```
mysql> CREATE TABLE Location_Access(
    -> lpid INT UNSIGNED NOT NULL,
    -> uid INT UNSIGNED NOT NULL,
    -> FOREIGN KEY(lpid) REFERENCES User_Location(lpid),
    -> FOREIGN KEY(uid) REFERENCES User(uid) );
Query OK, 0 rows affected (0.83 sec)
```

Create Location_Comment table:

```
mysql> CREATE TABLE Location_Comment(
    -> lid INT UNSIGNED NOT NULL,
    -> uid INT UNSIGNED NOT NULL,
    -> lcomm VARCHAR(140) NOT NULL,
    -> lcomm_time TIMESTAMP DEFAULT NOW() NOT NULL,
    -> FOREIGN KEY(lid) REFERENCES Location(lid),
    -> FOREIGN KEY(uid) REFERENCES User(uid) );
Query OK, 0 rows affected (0.51 sec)
```

Create Photos table:

```
mysql> CREATE TABLE Photos(
    -> pid INT UNSIGNED NOT NULL PRIMARY KEY AUTO_INCREMENT,
    -> pcap VARCHAR(140),
    -> ptime TIMESTAMP DEFAULT NOW() NOT NULL,
    -> plink VARCHAR(140) NOT NULL,
    -> lid INT UNSIGNED NOT NULL,
    -> FOREIGN KEY(lid) REFERENCES Location(lid) );
Query OK, 0 rows affected (0.89 sec)
```

Create User_Photo table:

```
mysql> CREATE TABLE User_Photo(
    -> pid INT UNSIGNED NOT NULL,
    -> uid INT UNSIGNED NOT NULL,
    -> FOREIGN KEY(pid) REFERENCES Photos(pid),
    -> FOREIGN KEY(uid) REFERENCES User(uid) );
Query OK, 0 rows affected (0.58 sec)
```

Create Photo_Access table:

```
mysql> CREATE TABLE Photo_Access(
    -> pid INT UNSIGNED NOT NULL,
    -> uid INT UNSIGNED NOT NULL,
    -> FOREIGN KEY(pid) REFERENCES Photos(pid),
    -> FOREIGN KEY(uid) REFERENCES User(uid) );
Query OK, 0 rows affected (0.42 sec)
```

Create Photo_Comment table:

```
mysql> CREATE TABLE Photo_Comment(
    -> pid INT UNSIGNED NOT NULL,
    -> uid INT UNSIGNED NOT NULL,
    -> pcomm VARCHAR(140) NOT NULL,
    -> pcomm_time TIMESTAMP DEFAULT NOW() NOT NULL,
    -> FOREIGN KEY(pid) REFERENCES Photos(pid),
    -> FOREIGN KEY(uid) REFERENCES User(uid) );
Query OK, 0 rows affected (0.87 sec)
```

Create Messages table:

```
mysql> CREATE TABLE Messages(
    -> mid INT UNSIGNED NOT NULL PRIMARY KEY AUTO_INCREMENT,
    -> mcontent VARCHAR(140) NOT NULL,
    -> mtime TIMESTAMP DEFAULT NOW() NOT NULL,
    -> lid INT UNSIGNED NOT NULL,
    -> FOREIGN KEY(lid) REFERENCES Location(lid) );
Query OK, 0 rows affected (0.47 sec)
```

Create User_Message table:

```
mysql> CREATE TABLE User_Message(
    -> mid INT UNSIGNED NOT NULL,
    -> uid INT UNSIGNED NOT NULL,
    -> FOREIGN KEY(mid) REFERENCES Messages(mid),
    -> FOREIGN KEY(uid) REFERENCES User(uid) );
Query OK, 0 rows affected (0.52 sec)
```

Create Message_Access table:

```
mysql> CREATE TABLE Message_Access(
    -> mid INT UNSIGNED NOT NULL,
    -> uid INT UNSIGNED NOT NULL,
    -> FOREIGN KEY(mid) REFERENCES Messages(mid),
    -> FOREIGN KEY(uid) REFERENCES User(uid) );
Query OK, 0 rows affected (0.63 sec)
```

Create Message_Comment table:

```
mysql> CREATE TABLE Message_Comment(
    -> mid INT UNSIGNED NOT NULL,
    -> uid INT UNSIGNED NOT NULL,
    -> mcomm VARCHAR(140) NOT NULL,
    -> mcomm_time TIMESTAMP DEFAULT NOW() NOT NULL,
    -> FOREIGN KEY(mid) REFERENCES Messages(mid),
    -> FOREIGN KEY(uid) REFERENCES User(uid) );
Query OK, 0 rows affected (0.76 sec)
```

SQL Queries

Query 1

To create a simple user you would type the following query:

INSERT INTO User(uname, password, fname, lname)
VALUES('username', 'password', 'first', 'last');

Query 2

To view all the friends and their circles

SELECT DISTINCT fname, lname, cname FROM User JOIN Friends ON (User.uid = Friends.uid1 OR User.uid = Friends.uid2) JOIN Circle ON (User.uid = Circle.uid) JOIN Circle_Name ON (Circle.cid = Circle_Name.cid) WHERE Friends.time_accept IS NOT NULL AND Friends.uid1 = (SELECT uid FROM User WHERE uname = 'DoraNYC');

Query 3

Delete friend requests older than three weeks

DELETE FROM Friends WHERE time_sent < NOW()-21 AND time_accept IS NULL;

Query 4

Seeing other users in Paris (Josh1989), First you need to find the longitude and latitude of Paris (assuming there is only one) For that you need to make the following query:

SELECT longitude, latitude FROM Location WHERE city = 'Paris';

After you get the value, you would use it in the following query:

```
SELECT uname FROM User JOIN Location_Access ON (User.uid = Location_Access.uid) JOIN User_Location ON (Location_Access.lpid = User_Location.lpid) JOIN Location ON (User_Location.lid = Location.lid) WHERE User_Location.uid = (SELECT uid FROM User WHERE uname = 'Josh1989') AND (3959 * acos(cos(radians(48.85660172) ) * cos(radians(latitude)) * cos(radians(2.352203) - radians(longitude)) + sin(radians(2.352203)) * sin(radians(latitude)) ) ) > 30
UNION
SELECT uname FROM User JOIN Photo_Access ON (User.uid = Photo_Access.uid) JOIN User_Photo ON (User_Photo.pid = Photo_Access.pid) JOIN Photos ON (User_Photo.pid = Photos.pid) JOIN Location ON (Photos.lid = Location.lid) WHERE User_Photo.uid = (SELECT uid FROM User WHERE uname = 'Josh1989') AND (3959 * acos(cos(radians(48.85660172) ) * cos(radians(latitude)) * cos(radians(2.352203) - radians(longitude)) + sin(radians(2.352203)) * sin(radians(latitude)) ) ) > 30
UNION
SELECT uname FROM User JOIN Message_Access ON (User.uid = Message_Access.uid) JOIN User_Message ON (User_Message.mid = Message_Access.mid) JOIN Messages ON (User_Message.mid = Messages.mid) JOIN Location ON (Messages.lid = Location.lid) WHERE User_Message.uid = (SELECT uid FROM User WHERE uname = 'Josh1989') AND (3959 * acos(cos(radians(48.85660172) ) * cos(radians(latitude)) * cos(radians(2.352203) - radians(longitude)) + sin(radians(2.352203)) * sin(radians(latitude)) ) ) < 20;
```

Query 5

Messages Written By DoraNYC in order of date written:

```
SELECT mcontent FROM User JOIN User_Message ON (User.uid = User_Message.uid) JOIN Messages ON (User_Message.mid = Messages.mid) WHERE uname = 'DoraNYC' ORDER BY mtime;
```

Query 6

Messages Josh1989 has access to that has ‘train’ in it:

```
SELECT mcontent FROM User JOIN Message_Access ON (User.uid = Message_Access.uid) JOIN Messages ON (Message_Access.mid = Messages.mid) WHERE uname = 'Josh1989' AND INSTR(mcontent, 'train') > 0;
```

Query 7

Just Like the previous one we would find all the longitude and latitude of Niagra Falls, which will give the longitude and latitude value, then do the following query:

SELECT pid, pcap, ptime, Photos.lid, plink FROM Photos JOIN Location ON (Photos.lid = Location.lid) WHERE DAY(ptime) = DAY(NOW()) AND (3959 * acos(cos(radians(79.03800201)) * cos(radians(latitude)) * cos(radians(43.09700012) - radians(longitude)) + sin(radians(43.09700012)) * sin(radians(latitude)))) < 30;

Populating Database

The User Table was populated with the following data:

```
mysql> SELECT * FROM User;
+-----+------------+-----------+-----------+----------+
| uid | uname      | password  | fname     | lname    |
+-----+------------+-----------+-----------+----------+
|   1 | DoraNYC    | explorer  | Dora      | Marquez  |
|   2 | Josh1989   | drakejosh | Josh      | Peck     |
|   3 | PirateKing | onepiece  | Gold D.   | Roger    |
|   4 | Luffy_King | strawhat  | Monkey D. | Luffy    |
|   5 | sanjiChef  | food      | Sanji     | Vinsmoke |
+-----+------------+-----------+-----------+----------+
```

The UserBio Table:

```
+-----+------------+-----------------------------+--------+
| uid | dob        | bio                         | gender |
+-----+------------+-----------------------------+--------+
|   1 | 2016-04-24 | I want to Explore the world | F      |
|   3 | 2016-04-24 | I want to be the pirate king| M      |
+-----+------------+-----------------------------+--------+
2 rows in set (0.00 sec)
```

The Friends Table:

uid1	uid2	time_sent	time_accept
1	1	2016-04-24 17:07:39	2016-04-24 17:07:39
2	2	2016-04-24 17:07:39	2016-04-24 17:07:39
3	3	2016-04-24 17:07:39	2016-04-24 17:07:39
4	4	2016-04-24 17:07:39	2016-04-24 17:07:39
5	5	2016-04-24 17:07:39	2016-04-24 17:07:39
1	5	2016-04-24 17:09:00	NULL
1	3	2016-04-24 17:15:49	2016-04-24 17:15:49
1	4	2016-04-24 17:15:49	2016-04-24 17:15:49
2	5	2016-04-24 17:15:49	2016-04-24 17:15:49
2	4	2016-04-24 17:15:49	2016-04-24 17:15:49

10 rows in set (0.00 sec)

The Circle_Name Table:

cid	cname
1	Cool People
2	Pirate Kings
3	Losers

3 rows in set (0.00 sec)

The Circle Table:

cid	uid
1	1
1	3
1	4
2	3
2	4
3	2
3	5

7 rows in set (0.00 sec)

The Location Table:

lid	longitude	latitude	city	country
1	79.03800201	43.09700012	Niagra Falls	Canada
2	73.96540070	40.78290176	New York	USA
3	74.04450226	40.68920135	New York	USA
4	78.04219818	27.17499924	Agra	India
5	2.35220003	48.85660172	Paris	France

5 rows in set (0.00 sec)

The Attraction Table:

```
+-----+-------------------+
| lid | aname             |
+-----+-------------------+
|   1 | Niagra Falls      |
|   2 | Central Park      |
|   3 | Statue of Liberty |
|   4 | Taj Mahal         |
+-----+-------------------+
4 rows in set (0.00 sec)
```

The User_Location Table:

```
+------+-----+-----+---------------------+
| lpid | lid | uid | ltime               |
+------+-----+-----+---------------------+
|    1 |   5 |   2 | 2016-04-24 18:23:29 |
|    2 |   5 |   5 | 2016-04-24 18:23:29 |
+------+-----+-----+---------------------+
2 rows in set (0.00 sec)
```

Location_Access Table:

```
+------+-----+
| lpid | uid |
+------+-----+
|    1 |   2 |
|    1 |   5 |
|    2 |   5 |
+------+-----+
3 rows in set (0.00 sec)
```

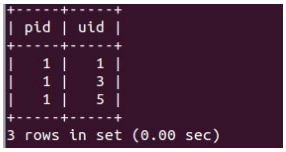
Photos Table:

```
+-----+--------+---------------------+--------------------------------+-----+
| pid | pcap   | ptime               | plink                          | lid |
+-----+--------+---------------------+--------------------------------+-----+
|   1 | OMG <3 | 2016-04-24 18:35:18 | http://i.imgur.com/6KRngnY.png |   1 |
+-----+--------+---------------------+--------------------------------+-----+
1 row in set (0.00 sec)
```

User_Photo Table:

```
+-----+-----+
| pid | uid |
+-----+-----+
|   1 |   1 |
+-----+-----+
1 row in set (0.00 sec)
```

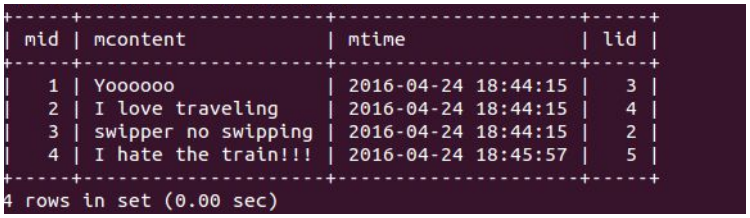
Photo_Access Table:



pid	uid
1	1
1	3
1	5

3 rows in set (0.00 sec)

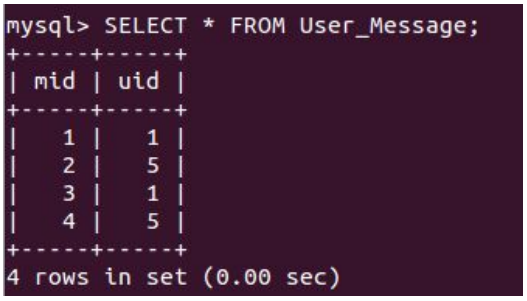
Messages Table:



mid	mcontent	mtime	lid
1	Yoooooo	2016-04-24 18:44:15	3
2	I love traveling	2016-04-24 18:44:15	4
3	swipper no swipping	2016-04-24 18:44:15	2
4	I hate the train!!!	2016-04-24 18:45:57	5

4 rows in set (0.00 sec)

User_Messages Table:

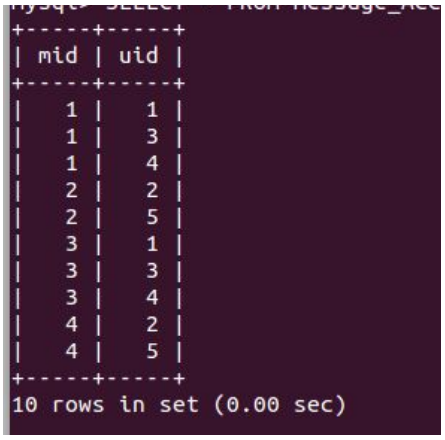


```
mysql> SELECT * FROM User_Message;
```

mid	uid
1	1
2	5
3	1
4	5

4 rows in set (0.00 sec)

Message_Access Table:



mid	uid
1	1
1	3
1	4
2	2
2	5
3	1
3	3
3	4
4	2
4	5

10 rows in set (0.00 sec)

Setup a trigger on User table, every time a user is made, he/she is added as his/her own friend on the Friends table:

```
mysql> DELIMITER $$
mysql> CREATE TRIGGER add_you
    -> AFTER INSERT ON User
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO Friends(uid1, uid2, time_sent, time_accept)
    -> SELECT uid, uid, NOW(), NOW() FROM User
    -> WHERE uid = (SELECT MAX(uid) FROM User);
    -> END $$
Query OK, 0 rows affected (0.39 sec)
```

Testing Queries:

Query 1

The first query performed changes the User and the Friends table:

```
mysql> INSERT INTO User(uname, password, fname, lname)
    -> VALUES('username', 'password', 'first', 'last');
Query OK, 1 row affected (0.51 sec)
```

```
mysql> SELECT * FROM User;
+-----+-----------+----------+----------+---------+
| uid | uname     | password | fname    | lname   |
+-----+-----------+----------+----------+---------+
|   1 | DoraNYC   | explorer | Dora     | Marquez |
|   2 | Josh1989  | drakejosh| Josh     | Peck    |
|   3 | PirateKing| onepiece | Gold D.  | Roger   |
|   4 | Luffy_King| strawhat | Monkey D.| Luffy   |
|   5 | sanjiChef | food     | Sanji    | Vinsmoke|
|   6 | username  | password | first    | last    |
+-----+-----------+----------+----------+---------+
6 rows in set (0.02 sec)
```

```
mysql> SELECT * FROM Friends;
+------+------+---------------------+---------------------+
| uid1 | uid2 | time_sent           | time_accept         |
+------+------+---------------------+---------------------+
|    1 |    1 | 2016-04-24 17:07:39 | 2016-04-24 17:07:39 |
|    2 |    2 | 2016-04-24 17:07:39 | 2016-04-24 17:07:39 |
|    3 |    3 | 2016-04-24 17:07:39 | 2016-04-24 17:07:39 |
|    4 |    4 | 2016-04-24 17:07:39 | 2016-04-24 17:07:39 |
|    5 |    5 | 2016-04-24 17:07:39 | 2016-04-24 17:07:39 |
|    1 |    5 | 2015-12-05 12:30:17 | NULL                |
|    1 |    3 | 2016-04-24 17:15:49 | 2016-04-24 17:15:49 |
|    1 |    4 | 2016-04-24 17:15:49 | 2016-04-24 17:15:49 |
|    2 |    5 | 2016-04-24 17:15:49 | 2016-04-24 17:15:49 |
|    2 |    4 | 2016-04-24 17:15:49 | 2016-04-24 17:15:49 |
|    6 |    6 | 2016-04-24 23:36:22 | 2016-04-24 23:36:22 |
+------+------+---------------------+---------------------+
11 rows in set (0.65 sec)
```

It clearly works as it adds the new user and adds him as his own friend which is done by the trigger.

Query 2

```
mysql> SELECT DISTINCT fname, lname, cname  FROM User JOIN Friends ON (User.uid = Friends.uid1 O
R User.uid = Friends.uid2) JOIN Circle ON (User.uid = Circle.uid) JOIN Circle_Name ON (Circle.ci
d = Circle_Name.cid) WHERE Friends.time_accept IS NOT NULL AND Friends.uid1 = (SELECT uid FROM U
ser WHERE uname = 'DoraNYC');
+-----------+---------+--------------+
| fname     | lname   | cname        |
+-----------+---------+--------------+
| Dora      | Marquez | Cool People  |
| Gold D.   | Roger   | Cool People  |
| Gold D.   | Roger   | Pirate Kings |
| Monkey D. | Luffy   | Cool People  |
| Monkey D. | Luffy   | Pirate Kings |
+-----------+---------+--------------+
5 rows in set (0.11 sec)
```

So Dora is only in one circle ‘Cool People’ with her friends Gold D. and Monkey D. , however, Gold D. and Monkey D. is in a circle without Dora which is ‘Pirate Kings’, since the tasks was to show friends and their circles, this result clearly shows that the query works and of course Dora is her own friend which is why her name is there too.

Query 3

```
mysql> DELETE FROM Friends WHERE time_sent < NOW()-21 AND time_accept IS NULL;
Query OK, 1 row affected (0.23 sec)
```

One row was affected, which should have happen. In the data, Sanji sent Dora a Friend Request last year, so if we look at the friends table now, that should be gone:

```
+------+------+---------------------+---------------------+
| uid1 | uid2 | time_sent           | time_accept         |
+------+------+---------------------+---------------------+
|    1 |    1 | 2016-04-24 17:07:39 | 2016-04-24 17:07:39 |
|    2 |    2 | 2016-04-24 17:07:39 | 2016-04-24 17:07:39 |
|    3 |    3 | 2016-04-24 17:07:39 | 2016-04-24 17:07:39 |
|    4 |    4 | 2016-04-24 17:07:39 | 2016-04-24 17:07:39 |
|    5 |    5 | 2016-04-24 17:07:39 | 2016-04-24 17:07:39 |
|    1 |    3 | 2016-04-24 17:15:49 | 2016-04-24 17:15:49 |
|    1 |    4 | 2016-04-24 17:15:49 | 2016-04-24 17:15:49 |
|    2 |    5 | 2016-04-24 17:15:49 | 2016-04-24 17:15:49 |
|    2 |    4 | 2016-04-24 17:15:49 | 2016-04-24 17:15:49 |
|    6 |    6 | 2016-04-24 23:36:22 | 2016-04-24 23:36:22 |
+------+------+---------------------+---------------------+
10 rows in set (0.00 sec)
```

Since Sanji’s friend request was the only one not accepted, therefore after deleting none of the rows for time_accept is NULL.

Query 4

```
mysql> SELECT uname FROM User JOIN Location_Access ON (User.uid = Location_Access.uid) JOIN User
_Location ON (Location_Access.lpid = User_Location.lpid) JOIN Location ON (User_Location.lid = L
ocation.lid) WHERE User_Location.uid = (SELECT uid FROM User WHERE uname = 'Josh1989') AND (3959
 * acos(cos(radians(48.85660172) ) * cos(radians(latitude)) * cos(radians(2.352203) - radians(lo
ngitude)) + sin(radians(2.352203)) * sin(radians(latitude)) ) )  > 30
    -> UNION
    -> SELECT uname FROM User JOIN Photo_Access ON (User.uid = Photo_Access.uid) JOIN User_Photo
 ON (User_Photo.pid = Photo_Access.pid) JOIN Photos ON (User_Photo.pid = Photos.pid) JOIN Locati
on ON (Photos.lid = Location.lid) WHERE User_Photo.uid = (SELECT uid FROM User WHERE uname = 'Jo
sh1989') AND (3959 * acos(cos(radians(48.85660172) ) * cos(radians(latitude)) * cos(radians(2.35
2203) - radians(longitude)) + sin(radians(2.352203)) * sin(radians(latitude)) ) )  > 30
    -> UNION
    -> SELECT uname FROM User JOIN Message_Access ON (User.uid = Message_Access.uid) JOIN User_M
essage ON (User_Message.mid = Message_Access.mid) JOIN Messages ON (User_Message.mid = Messages.
mid) JOIN Location ON (Messages.lid = Location.lid) WHERE User_Message.uid = (SELECT uid FROM Us
er WHERE uname = 'Josh1989') AND (3959 * acos(cos(radians(48.85660172) ) * cos(radians(latitude)
) * cos(radians(2.352203) - radians(longitude)) + sin(radians(2.352203)) * sin(radians(latitude)
```

So sanjiChef and Josh1989 are friends, they have both posted a location as Paris and sanjiChef does not have access to Josh1989's location data, but Josh1989 does have access to sanjiChef's location data, so the result should print Josh1989 and sanjiChef, since no other pictures or messages anywhere near Paris, which is does, so the query works:

```
+-----------+
| uname     |
+-----------+
| Josh1989  |
| sanjiChef |
+-----------+
2 rows in set (0.00 sec)
```

Query 5

```
mysql> SELECT mcontent FROM User JOIN User_Message ON (User.uid = User_Message.uid) JOIN Message
s ON (User_Message.mid = Messages.mid) WHERE uname = 'DoraNYC' ORDER BY mtime;
+--------------------+
| mcontent           |
+--------------------+
| swipper no swipping |
| Yoooooo            |
+--------------------+
2 rows in set (0.00 sec)
```

Dora wrote two messages, which were shown in the result by the order she wrote them. So the query works as it is supposed to.

Query 6

```
mysql> SELECT mcontent FROM User JOIN Message_Access ON (User.uid = Message_Access.uid) JOIN Mes
sages ON (Message_Access.mid = Messages.mid) WHERE uname = 'Josh1989' AND INSTR(mcontent, 'train
') > 0;
+--------------------+
| mcontent           |
+--------------------+
| I hate the train!!! |
+--------------------+
1 row in set (0.00 sec)
```

So, Josh1989 did not write any messages, however, his friend sanjiChef wrote two messages out of which Josh1989 has access to one, and that one has the word train in it, so the query clearly works.

Query 7

```
mysql> SELECT pid, pcap, ptime, Photos.lid, plink FROM Photos JOIN Location ON (Photos.lid = L
ation.lid) WHERE DAY(ptime) = DAY(NOW()) AND (3959 * acos(cos(radians(79.03800201) ) * cos(rad
ns(latitude)) * cos(radians(43.09700012 ) - radians(longitude)) + sin(radians(43.09700012 )) *
in(radians(latitude)) ) )  < 30;
Empty set (0.00 sec)
```

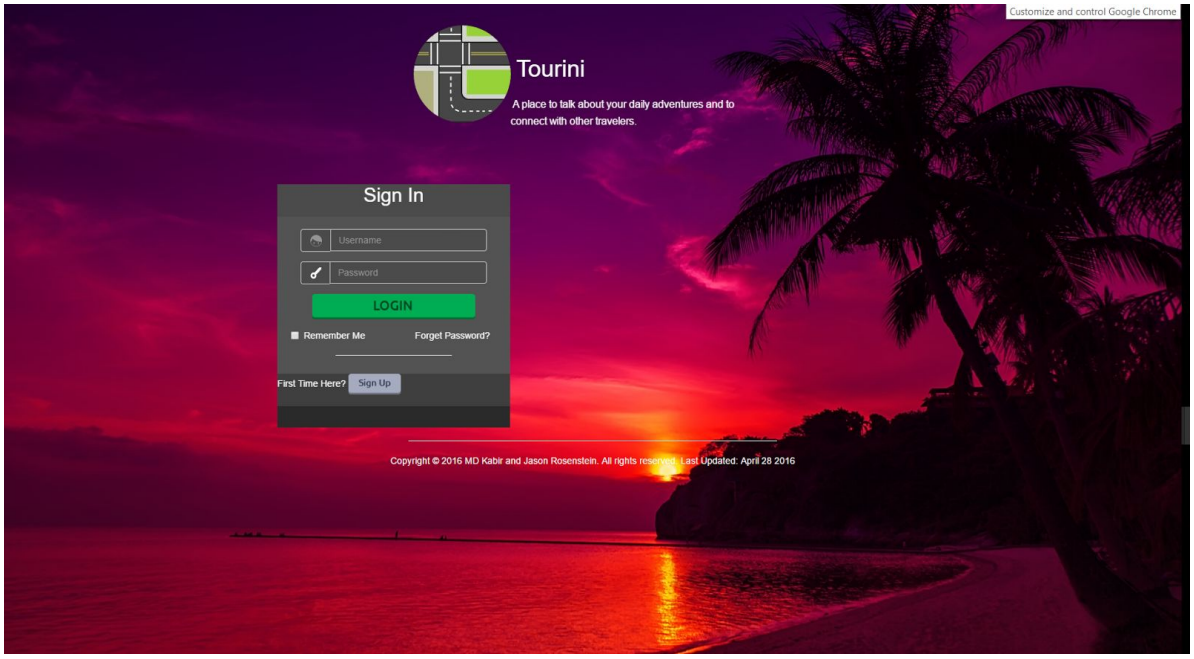
The reason the result in empty set, because there is only one photo in the database, which was taken in Niagra Falls however, it was posted yesterday, which is why technically there is not even any pictures posted today, so the query should return empty set, which it did, so it works.

Part 2

The tables that were modified during part two where the **message** table and the **circle** table to better accommodate our website design:

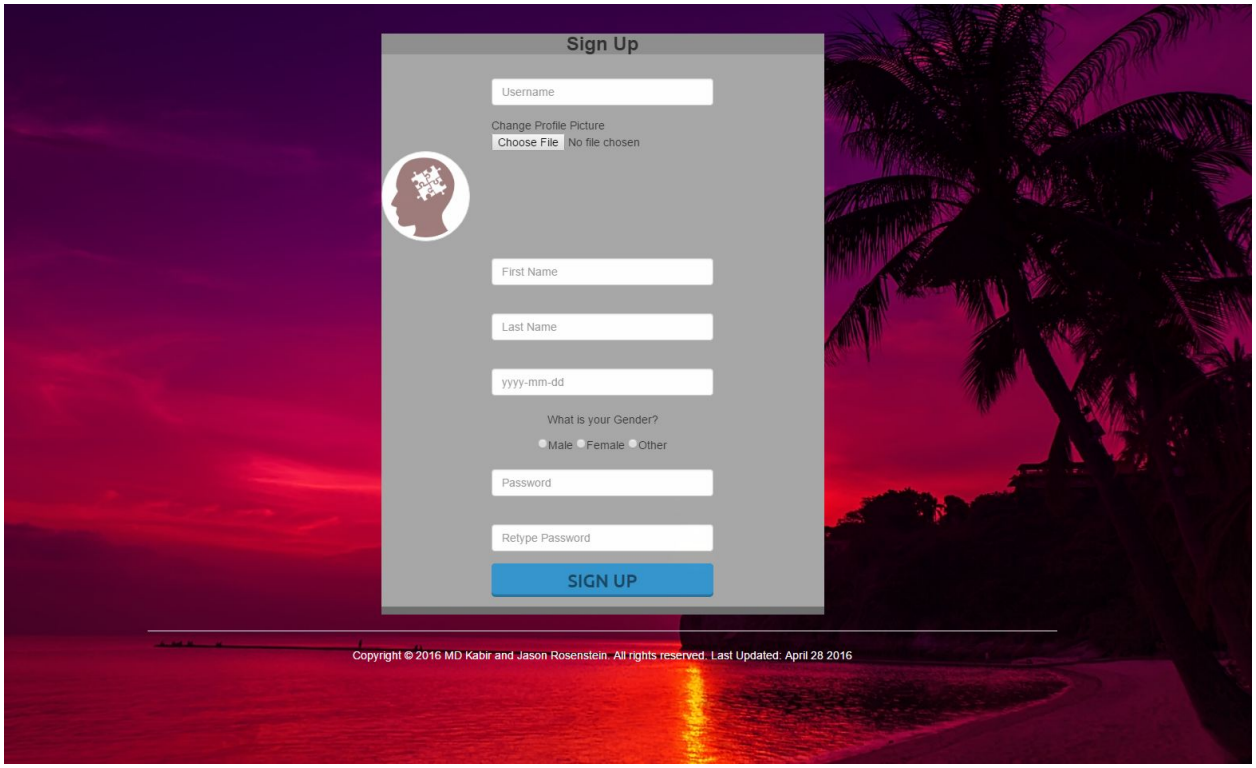
```
mysql> desc message;
+----------+--------------+------+-----+---------+----------------+
| Field    | Type         | Null | Key | Default | Extra          |
+----------+--------------+------+-----+---------+----------------+
| mid      | int(11)      | NO   | PRI | NULL    | auto_increment |
| uid      | int(11)      | NO   |     | NULL    |                |
| message  | varchar(145) | NO   |     | NULL    |                |
| timeSent | timestamp    | YES  |     | NULL    |                |
| lid      | int(11)      | YES  | MUL | NULL    |                |
+----------+--------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)

mysql> desc circle;
ERROR 1146 (42S02): Table 'tourini.circle' doesn't exist
mysql> desc circles;
+-------------+-------------+------+-----+---------+----------------+
| Field       | Type        | Null | Key | Default | Extra          |
+-------------+-------------+------+-----+---------+----------------+
| cid         | int(11)     | NO   | PRI | NULL    | auto_increment |
| uid1        | int(11)     | NO   | PRI | NULL    |                |
| uid2        | int(11)     | NO   | PRI | NULL    |                |
| circle_name | varchar(45) | YES  |     | NULL    |                |
+-------------+-------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

Front Page:



A simple front page with sign in box with a signup button that sends users to signup page, copyright footer, a nice scenery background.

Signup Page:



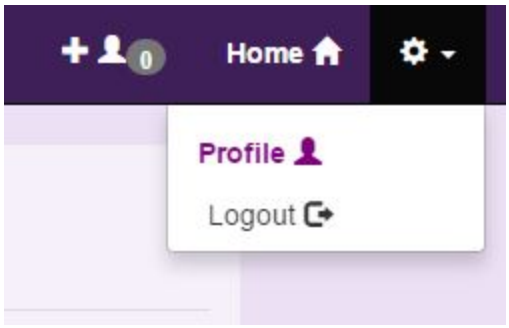
The signup page includes a username, firstname, lastname, date of birth, gender, password, and confirming the password. This page also implements Facebook's new module React.js to validate the form input as the user types in the input boxes. It also sets a default profile picture for the user as shown above.

Main Page Design Implementation:

For our main page, we decided to keep everything in a single page for a nice user interface, so when you search something, or add a friend, or view your or your friends' profile, it shows you in Bootstrap modals. That way you never leave your home page and able to do everything with a nice UI.

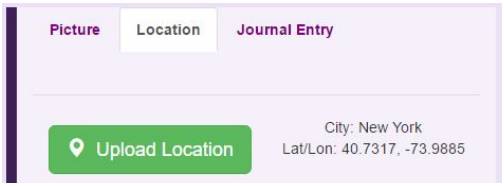
Main Page: Navigation

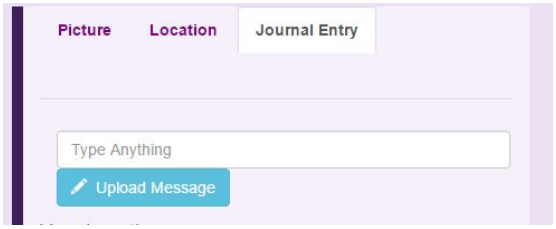
It has Tourini Logo on the side as a Brand Logo. It also has search box to search for anything. It also has to the right the friend requests you received, a home button to refresh the page, and setting to view your profile or logout.



Main Page: Posting

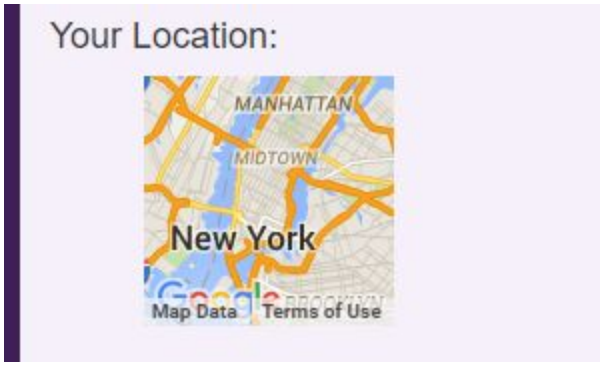
It has a tab for different types of posting, the one on top shows the tab for image, the pictures below show it for location and journal entry:





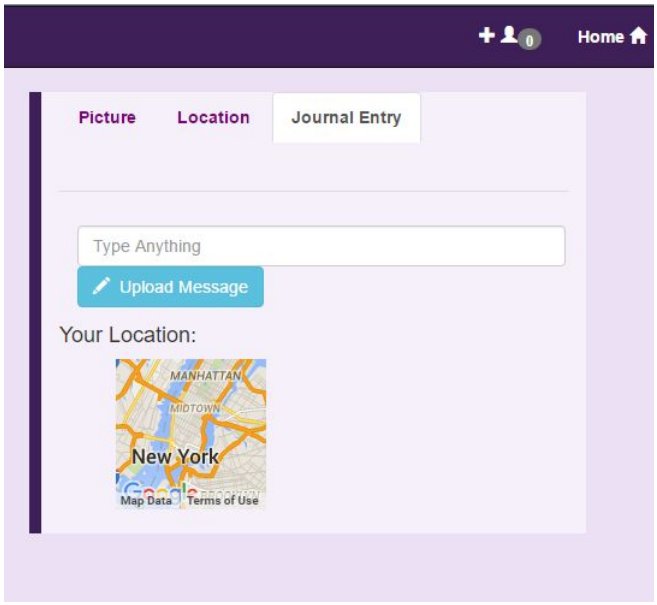
Main Page: Location

The main page also shows you the current location you are in:



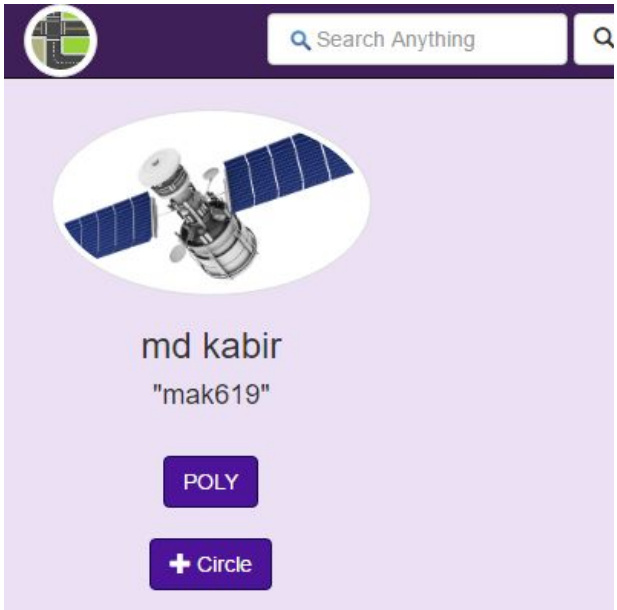
Main Page: Right Navigation

The right side of the page essentially consists of all those components talked about the posting and your current location:

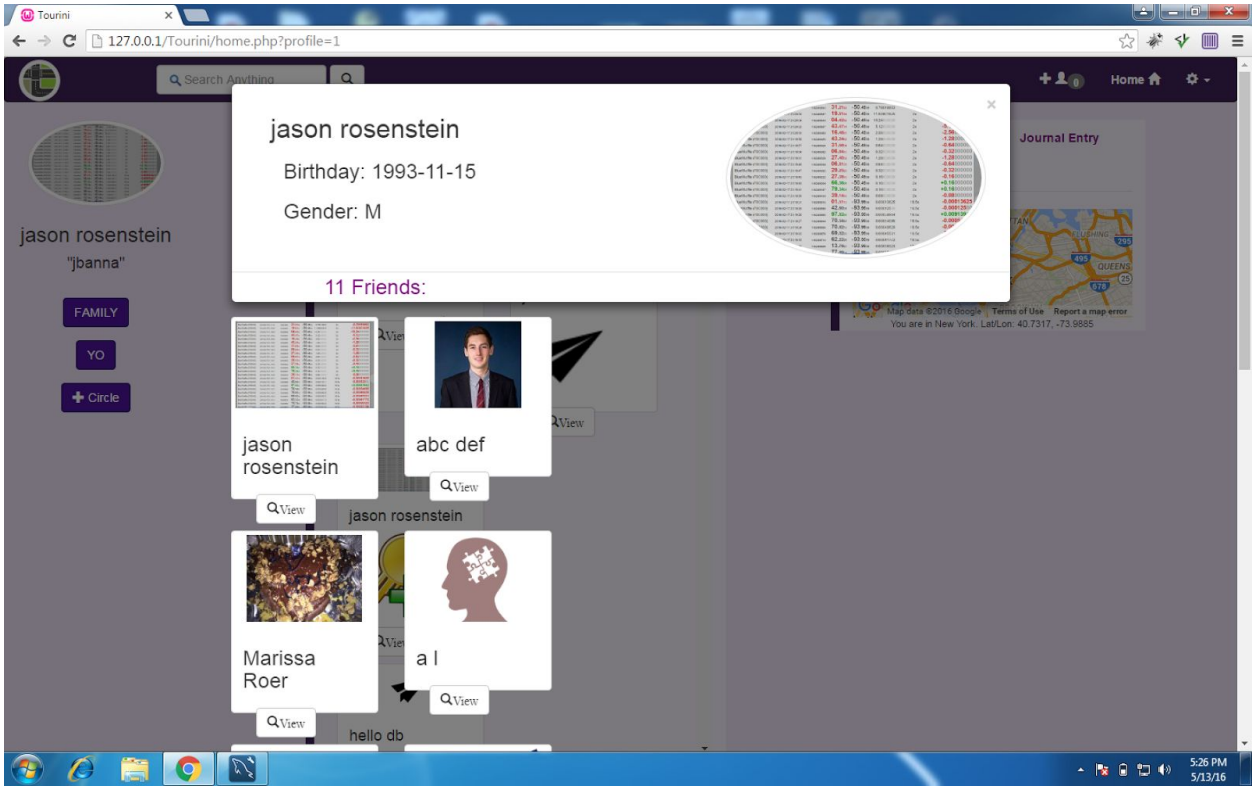


Main Page: Left Navigation

The left side of the page essentially consists of the User's profile picture, full name, username, circles you are in, and a button to add circle:

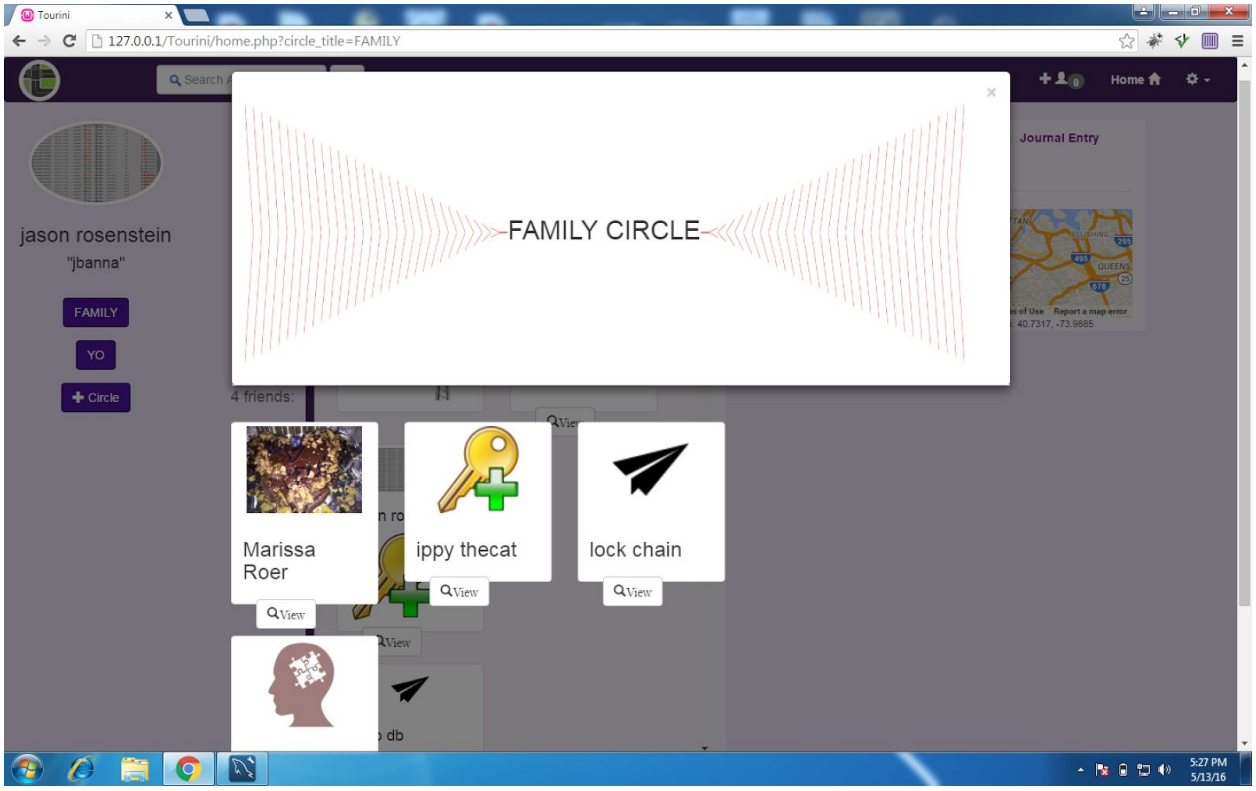


Profile Page:



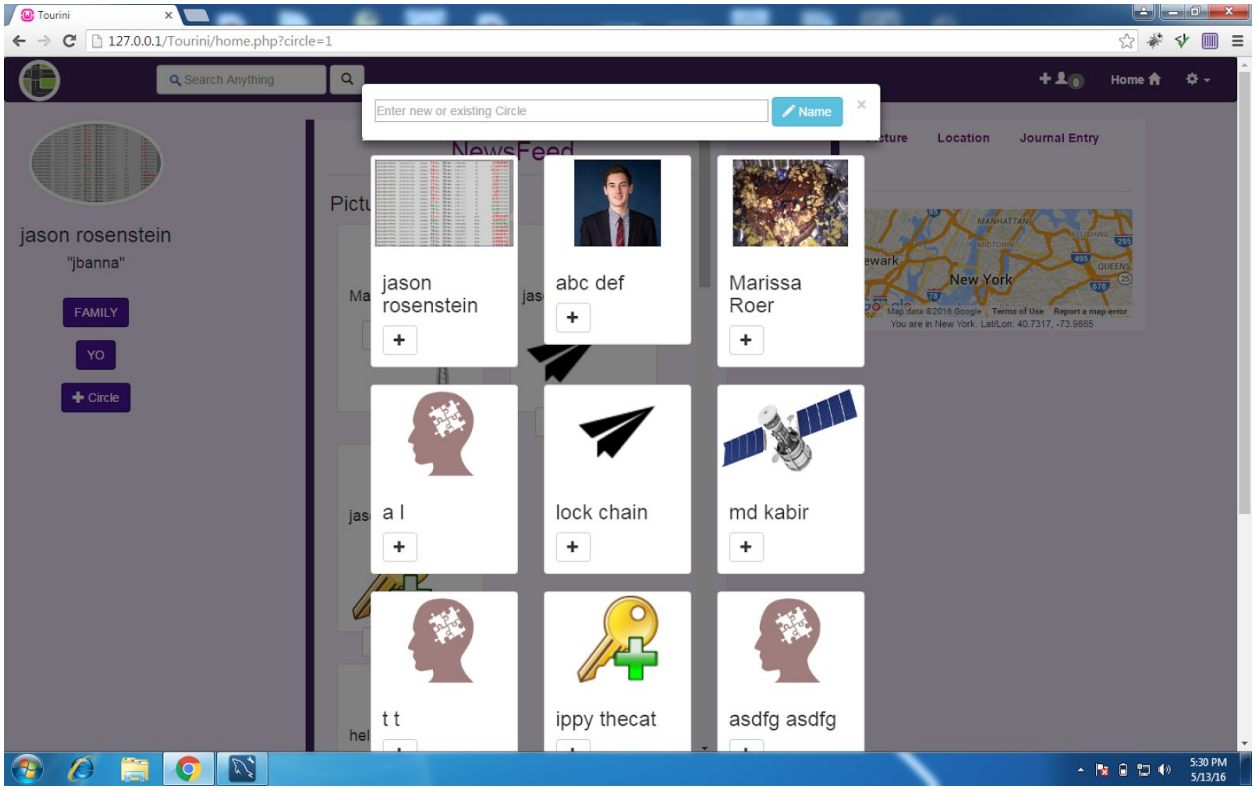
Displays Profile in a Modal so you don't have to leave the page and it has a nice User Interface. Just like this page, everything is displayed in Modals and it still displays a background. So the dark contrast from background to the Modal make it pop out and look incredibly great. By clicking on view besides the friends, we can see the profiles of our friends. We cannot see the profiles of our friends, friends. The ability to request friends, friends is available.

Circle Page:



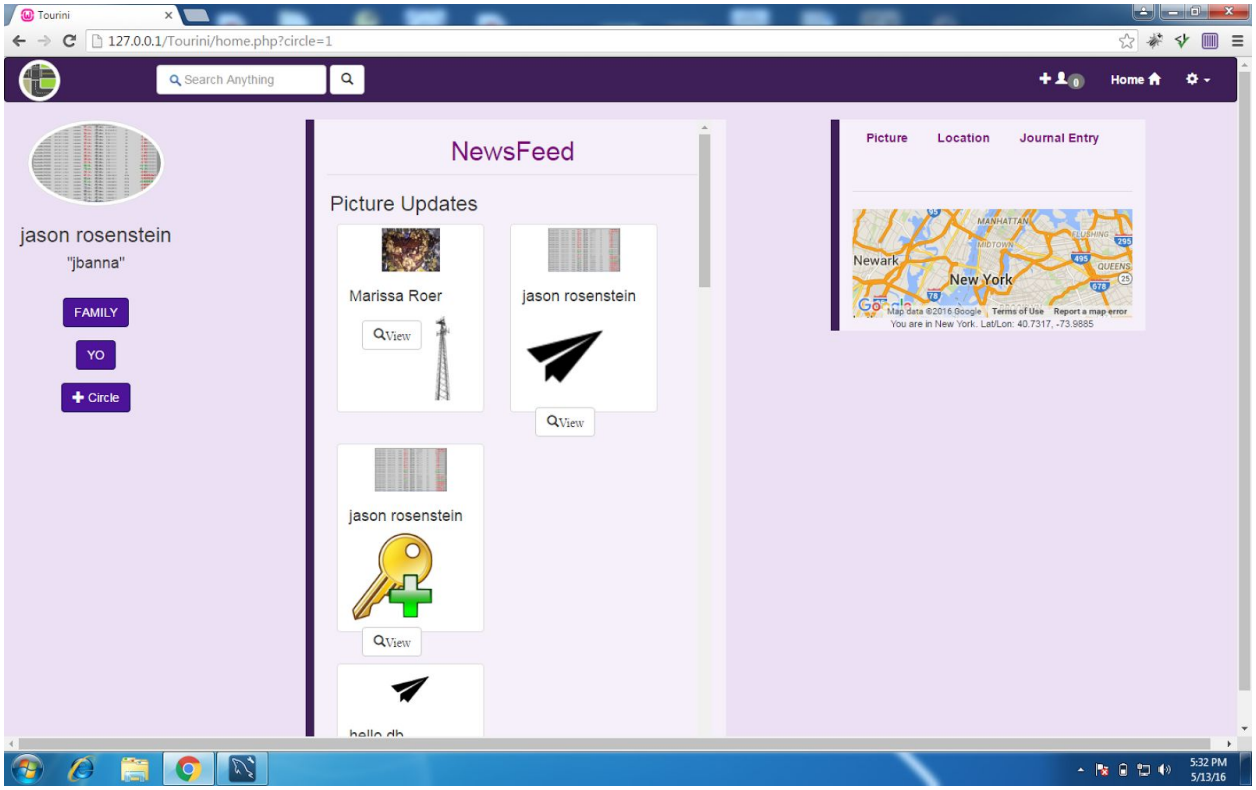
Displays unique circle name with proper friends floating below it.

Add or Edit Circles



This modal allows the user to create a new circle and subsequently add friends. Also the ability to add new users to present circles is possible from here.

So this is our site:



y

The NewsFeed allows you to see location, picture, and message updates done by users you've authenticated. It is an easy to use interface with a beautiful custom made layout.