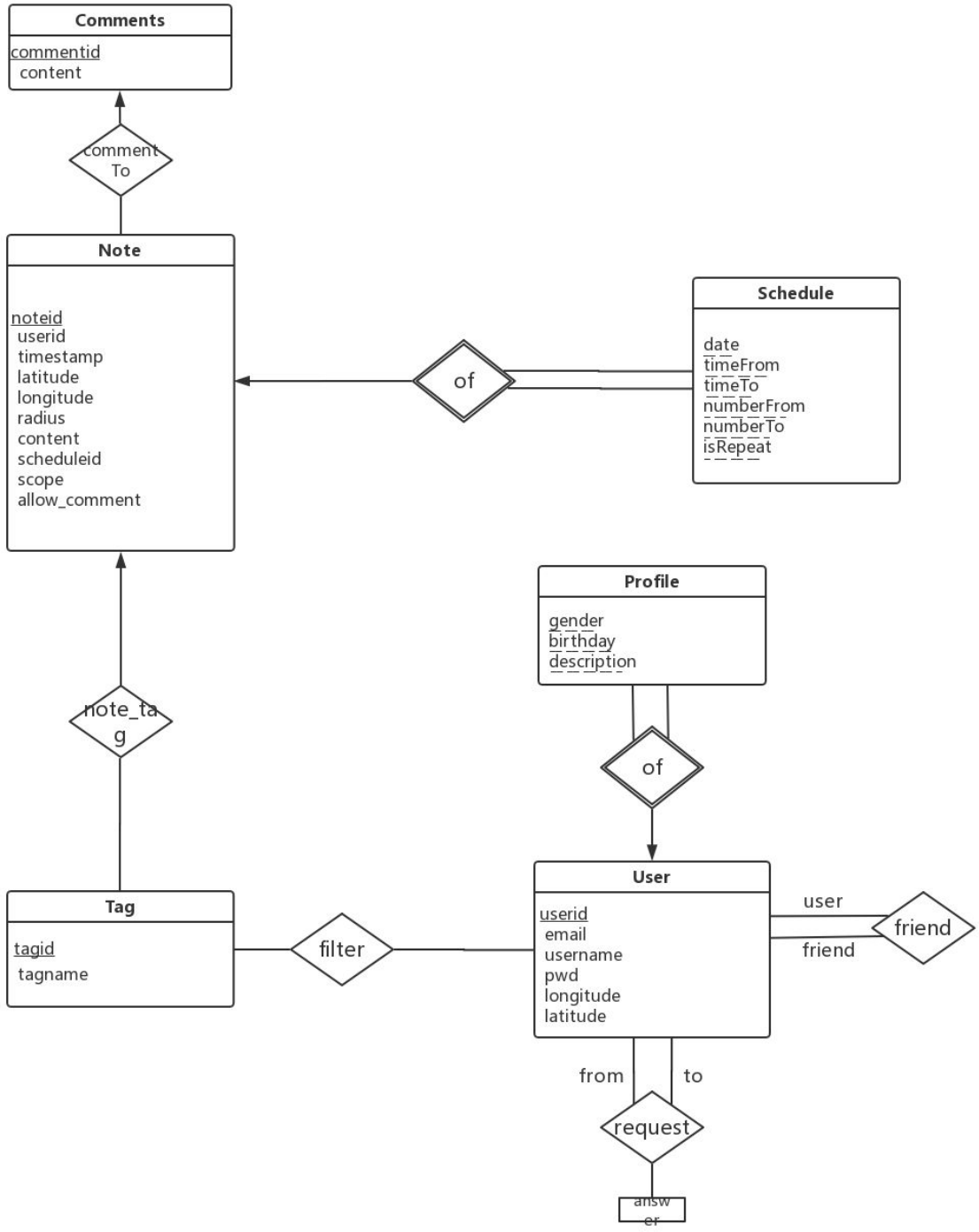
Project 1

Team Member： Wangyue Wang（ww1314), Xiaofan Ni (xn291)

1. ER Model



2. Schema

1. Note (noteid, userid, time, latitude, longitude, radius, content, scheduleid, scope, allow_comment)
2. Schedule (scheduleid, date, timeFrom, timeTo, numberFrom, numberTo, isRepeat)
3. Tag (tagid, tagname)
4. Note2tag (noteid, tagid)
5. Filter (userid, tagid)
6. User (userid, email, username, pwd, longitude, latitude)
7. FriendRequest (fromid, toid, answer)
8. Friendship (userid, friendid)
9. Comments (commentid, commentTo, content)
10. Profile (userid, gender, birthday, description)

3. Foreign Keys

1. Note ---scheduleid---> Schedule
2. Note ---userid---> User
3. Note2Tag ---tagid---> Tag
4. Note2Tag ---noteid---> Note
5. Filter ---noteid---> Note
6. Filter ---tagid---> Tag
7. Friendship ---userid---> User
8. Friendship ---friendid---> User
9. Comments ---commentTo---> Note
10. Profile ---userid---> User

4. Create Database

```
drop schema if exists proj1  ;
create schema proj1;
use proj1;

CREATE TABLE schedule (
 scheduleid INT AUTO_INCREMENT PRIMARY KEY,
 sdate DATE,
 timeFrom datetime,
 timeTo datetime,
 numberFrom INT,
 numberTo INT,
 isRepeat BOOLEAN
);

CREATE TABLE user(
 userid INT AUTO_INCREMENT PRIMARY KEY,
 email VARCHAR(60),
 username VARCHAR(60),
 pwd VARCHAR(60),
 latitude FLOAT,
 longitude FLOAT
```

```
);

CREATE TABLE friendRequest(
 fromid INT,
 toid INT,
 answer BOOLEAN,
 FOREIGN KEY (fromid) REFERENCES user(userid) ON DELETE CASCADE,
 FOREIGN KEY (toid) REFERENCES user(userid) ON DELETE CASCADE
);

CREATE TABLE note (
 noteid INT AUTO_INCREMENT PRIMARY KEY,
 time DATETIME,
 latitude FLOAT,
 longitude FLOAT,
 radius INT,
 content VARCHAR(2000),
 scheduleid INT,
 scope VARCHAR(60),
 allow_comment BOOLEAN,
 userid INT,
 FOREIGN KEY (userid) REFERENCES user(userid) ON DELETE CASCADE,
 FOREIGN KEY (scheduleid) REFERENCES schedule(scheduleid) ON DELETE CASCADE
);


CREATE TABLE tag (
 tagid INT AUTO_INCREMENT PRIMARY KEY,
 tagname VARCHAR(255)
);

CREATE TABLE note2tag(
 noteid INT,
 tagid INT,
 FOREIGN KEY (noteid) REFERENCES note(noteid) ON DELETE CASCADE,
 FOREIGN KEY (tagid) REFERENCES tag(tagid) ON DELETE CASCADE
);

CREATE TABLE filter(
 userid INT,
 tagid INT,
 FOREIGN KEY (userid) REFERENCES user(userid) ON DELETE CASCADE,
 FOREIGN KEY (tagid) REFERENCES tag(tagid) ON DELETE CASCADE
);

CREATE TABLE friendship(
 userid INT,
 friendid INT,
 FOREIGN KEY (userid) REFERENCES  user(userid) ON DELETE CASCADE,
 FOREIGN KEY (friendid) REFERENCES user(userid) ON DELETE CASCADE
```

```
);

CREATE TABLE comments(
 commentid INT AUTO_INCREMENT PRIMARY KEY,
 commentTo INT,
 content VARCHAR(255),
 FOREIGN KEY (commentTo) REFERENCES  note(noteid) ON DELETE CASCADE
);

CREATE TABLE profile(
 userid INT,
 gender VARCHAR(20),
 birthday VARCHAR(60),
 description VARCHAR(200),
 FOREIGN KEY (userid) REFERENCES user(userid) ON DELETE CASCADE
);
```

5. Populate Data

1. Diagram that shows test data

Action	Related Table	Data inserted
Add a user	User	`insert into user(email, username, pwd) values ('xn111@nyu.edu', 'feizhai', '123456');`

Add a note	Note, Schedule, Tag, Note2Tag	`#schedule`
		`insert into schedule(sdate, timeFrom, timeTo, numberFrom, numberTo, isRepeat)`
		`values`
		`('2018-11-28', '2018-11-28 14:00:00', '2018-11-28 18:00:00', 3, 3, true);`
		
		`#tag`
		`insert into tag (tagname) values ('tourism'), ('transportation');`
		
		`#note`
		`insert into note (time, latitude, longitude, radius, content, scheduleid, scope, allow_comment, userid) values`
		`('2018-11-28 12:00:00', 40.729511, -73.996460, 100, 'New York University', 1, 'everyone', true, 1);`
		
		`#note2tag`
		`insert into note2tag values (1, 1), (1, 2);`

Add a few friends for a user	Friendship	`insert into friendship values (1, 2), (1, 3), (1,4);`

Add a filter	Filter, Tag	`#tag`
		`insert into tag (tagname) values ('chill'), ('transportation');`
		`#filter, suppose we have tag 'chill' with id 1, and 'transportation' with id 2'`

		insert into filter **values** (1, 1), (1, 2);

2. Data Structure
(a) Note (noteid, userid, time, latitude, longitude, radius, content, scheduleid, scope, allow_comment)

Column name	Data type
noteid	int
userid	int
time	datetime
longitude	float
radius	float
content	int
scheduleid	int
scope	varchar(60)
allow_comment	boolean

(b) Schedule (scheduleid, date, timeFrom, timeTo, numberFrom, numberTo, isRepeat)

Column name	Data type
scheduleid	int
date	date
timeFrom	datetime
timeTo	datetime
numberFrom	int
numberTo	int
isRepeat	boolean

(c) Tag (tagid, tagname)

Column name	Data type

tagid	int
tagname	varchar(255)

(d) Note2tag (noteid, tagid)

Column name	Data type
noteid	int
tagid	int

(e) Filter (userid, tagid)

Column name	Data type
userid	int
tagid	int

(f) User (userid, email, username, pwd, longitude, latitude)

Column name	Data type
userid	int
email	varchar(60)
username	varchar(60)
pwd	varchar(60)
longitude	float
latitude	float

(g) FriendRequest (fromid, toid, answer)

Column name	Data type
fromid	int
toid	int
answer	boolean

(h) Friendship (userid, friendid)

Column name	Data type
userid	int
friendid	int

(i) Comments (commentid, commentTo, content)

Column name	Data type
commentid	int
commentTo	int
content	carchar(255)

(j) Profile (userid, gender, birthday, description)

Column name	Data type
userid	int
gender	varchar(20)
biethday	varchar(60)
description	varchar(200)

3. Insert Statement

```
insert into user(email, username, pwd) values
        ('xn111@nyu.edu', 'feizhai', '123456'),
        ('ww1111@nnyu.edu', 'crystal', '111'),
        ('cc12@nyu.edu', 'lisa', '0000'),
        ('zz111@nyu.edu', 'patrick', '333');
insert into friendship values (1, 2), (1, 3), (1,4);
insert into schedule(sdate, timeFrom, timeTo, numberFrom, numberTo, isRepeat) values
   ('2018-11-28', '2018-11-28 14:00:00', '2018-11-28 18:00:00', 3, 3, 1);
insert into tag (tagname) values ('tourism'), ('transportation');
insert into note (time, latitude, longitude, radius, content, scheduleid, scope, allow_comment, userid)
        values
        ('2018-11-28 12:00:00', 40.729511, -73.996460, 100, 'New York University', 1, 'everyone', 1, 1);
insert into note2tag values (1, 1), (1, 2);
insert into filter values (1, 1), (1, 2);
```

6. Query

1. Create a new user account, with name, login, and password.

```
insert into user(email, username, pwd) values ('xn111@nyu.edu', 'feizhai', '123456');
```

	userid	email	username	pwd	latitude	longitude
1	1	xn111@nyu.edu	feizhai	123456	<null>	<null>

2. Add a new note to the system, together with tags, and spatial and temporal constraints.

```
insert into schedule(sdate, timeFrom, timeTo, numberFrom, numberTo, isRepeat) values
        ('2018-11-28', '2018-11-28 14:00:00', '2018-11-28 18:00:00', 3, 3, 1);
insert into tag (tagname) values ('tourism'), ('transportation');
insert into note (time, latitude, longitude, radius, content, scheduleid, scope, allow_comment, userid)
values
   ('2018-11-28 12:00:00', 40.729511, -73.996460, 100, 'New York University', 1, 'everyone', 1, 1);
insert into note2tag values (1, 1), (1, 2);
```

	scheduleid	sdate	timeFrom	timeTo	numberFrom	numberTo	isRepeat
1	1	2018-11-28	2018-11-28 14:00:00	2018-11-28 18:00:00	3	3	1

2 rows

	tagid	tagname
1	1	tourism
2	2	transportation

1 row

	noteid	time	latitude	longitude	radius	content	scheduleid	scope	allow_comment	userid
1	1	2018-11-28 12:00:00	40.7295	-73.9965	100	New York University	1	everyone	1	1

2 rows

	noteid	tagid
1	1	1
2	1	2

3. For a given user, list all her friends.

```
#User 'feizhai', whose userid is 1, has friends crystal(id:2), lisa(id:3) and patrick(id:4)
select user.userid, username
from friendship, user
where friendship.userid = 1 and friendid = user.userid;
```

	userid	username
1	2	crystal
2	3	lisa
3	4	patrick

4. Givenauserandhercurrentlocation,currenttime,andcurrentstate,outputallnotesthatsheshouldcurrentlybeable to see given the filters she has set up.

```
# given user(id:1) and currloc(latitude: 40.729511, longitude: -73.996460),
# currtime('2018-11-28 21:00:00'), curr state('tourism'), filters(userid(:1), tagid(:1))
```

```
# Suppose we have pre-processed the currTime and got the dayname for the currTime, which is 3
select note.noteid
from (select distinct noteid from note2tag
    where tagid in (select tagid from filter where userid = 1)) as filtered, note natural join schedule
where note.noteid = filtered.noteid and

(40.729511-note.latitude)*(40.729511-note.latitude)+(-73.996460-note.longitude)*(-73.996460-note.longit
ude)
     < note.radius*note.radius and
    numberFrom<=3 and numberTo>=3 and timeFrom<='2018-11-28 21:00:00' and
timeTo>='2018-11-28 21:00:00' and
     ((isRepeat=false and date('2018-11-28 21:00:00')=sdate)or isRepeat=true) and
    note.noteid in (select filtered2.noteid
                from (select distinct noteid from note2tag
                    where tagid in
                    (select tagid from filter where userid = 1)) as filtered2, note2tag, tag
                where filtered2.noteid = note2tag.noteid and
                    note2tag.tagid = tag.tagid and tag.tagname like ('%tourism%'));
```

5. Given a note (that maybe was just added to the system) and the current time, output all users that should currently be able to see this note based on their filter and their last recorded location.

```
# given a note(noteid: 1), currTime('2018-11-28 21:00:00') and suppose we have pre-processed
# the currTime and got the day name for the curr, filter(userid:1),
# location(latitude: 40.729511, longitude: -73.996460)
select userid
from user
where 1 in (select noteid from note natural join schedule, note2tag
                where (numberFrom<=3 and numberTo>=3 and timeFrom<='2018-11-28 21:00:00'
                    and timeTo>='2018-11-28 21:00:00' and
                     (isRepeat=true or (isRepeat=false and sdate=date('2018-11-28 21:00:00')))) and
                    (user.longitude-note.longitude)*(user.longitude-note.longitude)+
                    (user.latitude-note.latitude)*(user.latitude-note.latitude)<note.radius*note.radius and
                    note.noteid=note2tag.noteid and note2tag.tagid in
                                    (select tagid from filter
                                     where filter.userid = user.userid));
```

6. In some scenarios, in very dense areas or when the user has defined very general filters, there may be a lot of notes that match the current filters for a user. Write a query showing how the user can further filter these notes by inputting one or more keywords that are matched against the text in the notes using the contains operator.

```
CREATE VIEW filteredNotes as (select * from note where ...); #notes based on filter
```

```
select noteid from filteredNotes where content like('%keywords%');
```

7. Function Description of Project

This web app is a social media based on location and time. User can post a note and then link it with spatial and temporal constraints and several tags. User can also view other user's note with customized filter.

8. Pages Design

1. Login page : page used to login.
2. Signup page : page used to add new user.
3. Index page : page used to show notes meet the filter.
4. Add note page : page used to add a note.
5. Profile page : page used to create or update user's profile.
6. Add friend page : page used to search and send friend request to a user.
7. Friend Request page : page used to show all the friend request and allow user to answer a friend request.