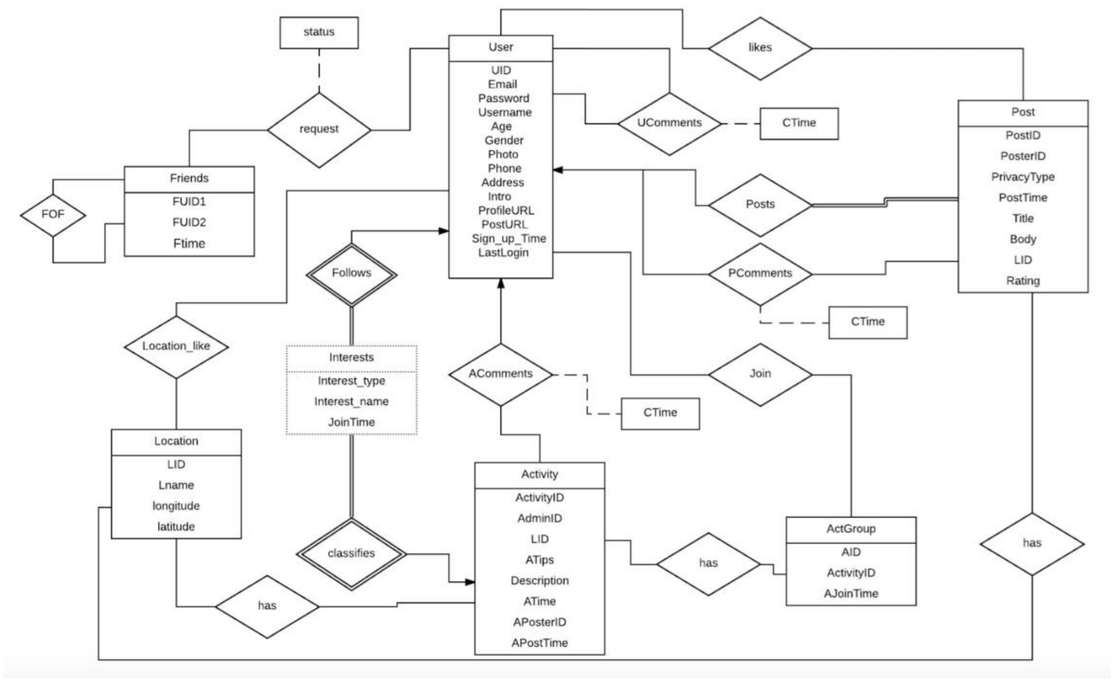
Ke Yang N17391687 ky935
Yixuan Li N15773578 yl3768

All of the project were finished by group work, all the solutions were generated after discussion by both of us .

1.E-R Diagram



2. Table Design

Table	**User(UID, Email, Password, Username, Age, Gender, Photo, Phone, Address Intro ProfileURL, PostURL, Sign_up_Time, LastLogin)**
Description	This table stores the basic User profiles. UID, Email, Username are unique and cannot be changed after signing up. The Intro is one user's description, PostURL links to user's list of post entries.

```
CREATE TABLE `User` (
  `UID` int(10) UNSIGNED NOT NULL,
  `Email` varchar(45) NOT NULL,
  `Password` varchar(45) NOT NULL,
  `Username` varchar(45) NOT NULL,
  `Age` int(11) DEFAULT NULL,
  `Gender` varchar(45) DEFAULT NULL,
  `Photo` blob,
  `Phone` bigint(11) DEFAULT NULL,
  `Address` varchar(45) DEFAULT NULL,
  `Intro` longtext,
  `ProfileURL` text,
  `PostURL` text,
  `Sign_up_Time` datetime NOT NULL,
  `LastLogin` datetime NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Table	**Post(PostID, PosterID, PrivacyType, PostTime, Title, Body, Location, Rating)**
Description	This table stores information of User's post. PrivacyType should be one of the "public, private, visible to friends, visible to friends and FoFs". Body is the main part of the post, it could include text and media. Location should be LID in Location table.

```
CREATE TABLE `Post` (
    `PostID` int(11) NOT NULL,
    `PosterID` int(11) NOT NULL,
    `PrivacyType` varchar(45) DEFAULT NULL,
    `PostTime` datetime NOT NULL,
    `Title` text,
    `Body` blob,
    `Location` varchar(45) DEFAULT NULL,
    `Rating` int(11) DEFAULT NULL,
    PRIMARY KEY (`PostID`));
```

Table	**Interests_UID(UID, interest_type,interest_name,jointime)**
Description	This table stores information of Users' interest type and name. Interest types are like such "sports, travel, food", interest_names are like such "hiking, basketball, bbq"

```
CREATE TABLE `user`.`Interests_UID` (
`UID` INT(11) NOT NULL ,
`interest_type` VARCHAR(45) NOT NULL ,
`interest_name` VARCHAR(45) NOT NULL ,
`JoinTime` DATETIME NOT NULL ,
PRIMARY KEY (`UID`, `interest_type`, `interest_name`));
```

Table	**Interests_Act(ActivityID,interest_type,interest_name,jointime)**

Description	This table stores information of Activities' interest type and name. Interest types are like such"sports, travel, food"

```
CREATE TABLE `user`.`Interests_Act` (
`ActivityID` INT(11) NULL ,
`interest_type` VARCHAR(45) NULL ,
`interest_name` VARCHAR(45) NULL ,
`JoinTime` DATETIME NOT NULL );
```

Table	**Activity(ActivityID,APosterID,ALocation,ATips,Description,ATime,APostTime)**
Description	Activity table stores information of Activity itself. I think this table is pretty straight forward.

```
CREATE TABLE `user`.`Activity` (
  `ActivityID` INT NOT NULL ,
`APosterID` INT NOT NULL ,
`ALocation` VARCHAR(45) NULL ,
`ATips` TEXT NULL ,
`Description` TEXT NULL ,
  `ATime` DATETIME NOT NULL ,
`APostTime` DATETIME NOT NULL ,
PRIMARY KEY (`ActivityID`))
```

Table	**ActGroup(UID,ActivityID,AJoinTime)**
Description	ActGroup table stores information of people who will follow and join the activity. Activity Groups.

```
CREATE TABLE `user`.`ActGroup` (
`UID` INT(11) NOT NULL ,
`ActivityID` INT(11) NOT NULL ,
`AJoinTime` DATETIME NOT NULL ,
PRIMARY KEY (`UID`, `ActivityID`)
CONSTRAINT `UID`
    FOREIGN KEY (`UID`)
    REFERENCES `user`.`User` (`UID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `ActivityID`
    FOREIGN KEY (‘ActivityID’)
    REFERENCES `user`.`Activity`(`ActivityID`)
    ON DELETE NO ACTION
     ON UPDATE NO ACTION));
```

Table	**Friends(FUID1,FUID2)**

Description	This table are the relation between two users. The information is stored bidirectional, that is , in the table, if there exists FUID1=1,FUID2=2, then there must exist a tuple that FUID1=2,FUID2=1.

```
CREATE TABLE `user`.`Friends` (
`FUID1` INT(11) NOT NULL ,
`FUID2` INT(11) NOT NULL ,
`FTime` DATETIME NOT NULL ,
PRIMARY KEY (`FUID1`, `FUID2`)
CONSTRAINT `FUID1`
    FOREIGN KEY (`FUID1`)
    REFERENCES `user`.`User` (`UID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `FUID2`
    FOREIGN KEY (`FUID2`)
    REFERENCES `user`.`User`(`UID`)
    ON DELETE NO ACTION
ON UPDATE NO ACTION));
```

Table	**request(UID1,UID2,status,ReqTime)**
Description	Request table stores information when user asks for friends with other users. Status stores the information of the request, if the user approved the request or not. ReqTime is the request time when user sends the friends invitation.

```
CREATE TABLE `user`.`request` (
`UID1` INT(11) NOT NULL ,
`UID2` INT(11) NOT NULL ,
`status` VARCHAR(45) NOT NULL ,
`ReqTime` DATETIME NOT NULL ,
PRIMARY KEY (`UID1`, `UID2`));
```

Table	**UComments(UCID,UID1,UID2,CTime,Body)**
Description	UComments stores information of the comments from one user to others. UCID is used to uniquely specify each comments, UID1,UID2 are foreign keys reference with User, UID. CTime represents the comment time. Body stores comments context or images,etc.

```
CREATE TABLE `user`.`UComments` (
`UCID` INT NOT NULL ,
`UID1` INT NOT NULL ,
`UID2` INT NOT NULL ,
`CTime` DATETIME NOT NULL ,
`Body` BLOB NOT NULL ,
PRIMARY KEY (`UCID`)
CONSTRAINT `UID1`
    FOREIGN KEY (`UID1`)
    REFERENCES `user`.`User` (`UID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `UID2`
    FOREIGN KEY (`UID2`)
    REFERENCES `user`.`User`(`UID`)
    ON DELETE NO ACTION
ON UPDATE NO ACTION));
```

Table	PComments(PCID,UID,PostID,CTime,Body)
Description	PComments stores information of the comments from one user to posts. PCID is used to uniquely specify each post comment. CTime represents the post comment time. Body stores comments context or images,etc.

```
CREATE TABLE `user`.`PComments` (
`PCID` INT NOT NULL ,
`UID` INT NOT NULL ,
`PostID` INT NOT NULL ,
`CTime` DATETIME NOT NULL ,
`Body` BLOB NOT NULL ,
PRIMARY KEY (`PCID`)
CONSTRAINT `UID`
    FOREIGN KEY (`UID`)
    REFERENCES `user`.`User` (`UID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `PostID`
    FOREIGN KEY (`PostID`)
    REFERENCES `user`.`Post`(`PostID`)
    ON DELETE NO ACTION
ON UPDATE NO ACTION));
```

Table	AComments(ACID,UID,ActivityID,Body,CTime)
Description	AComments stores information of the comments from one user to each activity. ACID is used to uniquely specify each activity comment. CTime represents the post comment time. Body stores comments context or images,etc.

```
CREATE TABLE `user`.`AComments` (
`ACID` INT NOT NULL ,
`UID` INT NOT NULL ,
`ActivityID` INT NOT NULL ,
`Body` BLOB NOT NULL ,
`CTime` DATETIME NOT NULL ,
PRIMARY KEY (`ACID`)
CONSTRAINT `UID`
    FOREIGN KEY (`UID`)
    REFERENCES `user`.`User` (`UID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `ActivityID`
    FOREIGN KEY (`ActivityID `)
    REFERENCES `user`.`Activity`(`ActivityID`)
    ON DELETE NO ACTION
ON UPDATE NO ACTION));
```

Table	**Location(LID, Lname, longitude, latitude)**
Description	This table stores location information.

```
CREATE TABLE `user`.`Location` (
`LID` INT(11) NOT NULL ,
`Lname` VARCHAR(45) NOT NULL ,
`longitude` VARCHAR(45) NOT NULL ,
`latitude` VARCHAR(45) NOT NULL ,
PRIMARY KEY (`LID`))
```

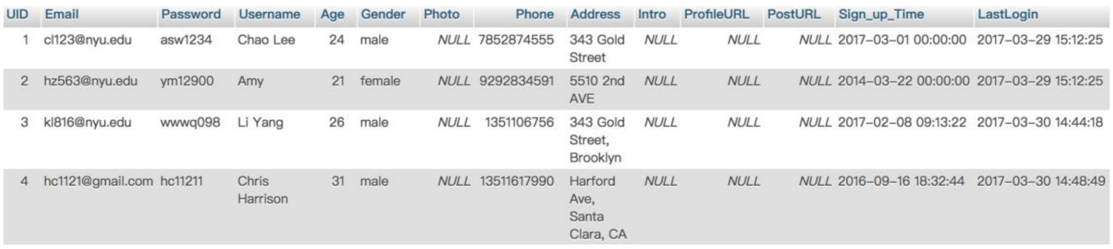
Table	**Location_like(UID, LID)**
Description	This table stores who like the Location.

```
CREATE TABLE `user`.`Location_like` (
`UID` INT(11) NOT NULL ,
`LID` INT(11) NOT NULL ,
PRIMARY KEY (`UID`, `LID`))
```

3. SQL queries:
3.1 Content Posting
(1).User sign up:

```
INSERT INTO `User` (`UID`, `Email`, `Password`, `Username`, `Age`, `Gender`,
`Photo`, `Phone`, `Address`, `Intro`, `ProfileURL`, `PostURL`, `Sign_up_Time`,
`LastLogin`) VALUES
(1, 'cl123@nyu.edu', 'asw123', 'Chao Lee', 24, 'male', NULL, 7852874555, '343
Gold Street', NULL, NULL, NULL, '2017-03-01 00:00:00', '2017-03-28 17:44:57'),
```

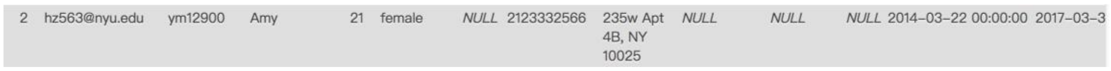
(2, 'hz563@nyu.edu', 'ym12900', 'Amy', 21, 'female', NULL, 9292834591, '5510 2nd AVE', NULL, NULL, NULL, '2014-03-22 00:00:00', '2017-03-28 17:46:29');



UID	Email	Password	Username	Age	Gender	Photo	Phone	Address	Intro	ProfileURL	PostURL	Sign_up_Time	LastLogin
1	cl123@nyu.edu	asw1234	Chao Lee	24	male	NULL	7852874555	343 Gold Street	NULL	NULL	NULL	2017-03-01 00:00:00	2017-03-29 15:12:25
2	hz563@nyu.edu	ym12900	Amy	21	female	NULL	9292834591	5510 2nd AVE	NULL	NULL	NULL	2014-03-22 00:00:00	2017-03-29 15:12:25
3	kl816@nyu.edu	wwwq098	Li Yang	26	male	NULL	1351106756	343 Gold Street, Brooklyn	NULL	NULL	NULL	2017-02-08 09:13:22	2017-03-30 14:44:18
4	hc1121@gmail.com	hc11211	Chris Harrison	31	male	NULL	13511617990	Harford Ave, Santa Clara, CA	NULL	NULL	NULL	2016-09-16 18:32:44	2017-03-30 14:48:49

(2). Create or edit profiles: (Profile is some attributes in the user table.)
UPDATE `User` SET
`Password`='ym12900',`LastLogin`=CURRENT_TIMESTAMP   WHERE UID='2'

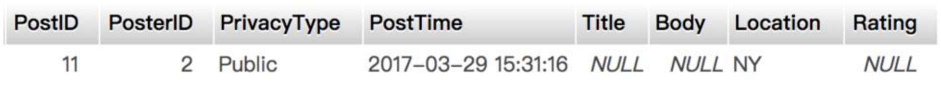
UPDATE `User` SET `Phone`='2123332566',`Address`='235w Apt 4B, NY 10025',`LastLogin`=CURRENT_TIMESTAMP   WHERE UID='2'



2	hz563@nyu.edu	ym12900	Amy	21	female	NULL	2123332566	235w Apt 4B, NY 10025	NULL	NULL	NULL	2014-03-22 00:00:00	2017-03-3

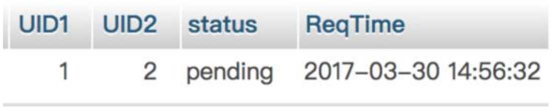
(3). Post an image:
INSERT INTO `Post` (`PostID`, `PosterID`, `PrivacyType`, `PostTime`, `Title`, `Body`, `Location`, `Rating`) VALUES ('1', '2', 'Public', CURRENT_TIME(), NULL, image binary, NY, NULL)

(4). Add new entry to the diaries:
INSERT INTO `Post` (`PostID`, `PosterID`, `PrivacyType`, `PostTime`, `Title`, `Body`, `Location`, `Rating`) VALUES ('11', '2', 'Public', CURRENT_TIME(),`My first post`, NULL, Alaska, NULL)



PostID	PosterID	PrivacyType	PostTime	Title	Body	Location	Rating
11	2	Public	2017-03-29 15:31:16	NULL	NULL	NY	NULL

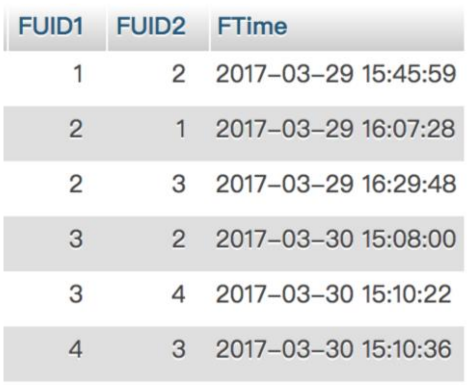
3.2 Friendship:
(1). Send friend request to someone:
INSERT INTO `request` (`UID1`, `UID2`, `status`, `ReqTime`) VALUES ('1', '2', 'pending ', CURRENT_TIME());



UID1	UID2	status	ReqTime
1	2	pending	2017-03-30 14:56:32

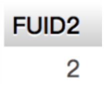
(2). Accept friend request from someone: (When become friends, each user add the other one at the same time, it is a mutual relationship.)
UPDATE `request` SET `status`='Approved' WHERE UID1 = '1' and UID2='2'
INSERT INTO `Friends` (`FUID1`, `FUID2`, `FTime`) VALUES ('1', '2', CURRENT_TIME());
INSERT INTO `Friends` (`FUID1`, `FUID2`, `FTime`) VALUES ('2', '1',

CURRENT_TIME());



FUID1	FUID2	FTime
1	2	2017-03-29 15:45:59
2	1	2017-03-29 16:07:28
2	3	2017-03-29 16:29:48
3	2	2017-03-30 15:08:00
3	4	2017-03-30 15:10:22
4	3	2017-03-30 15:10:36

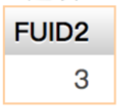
(3). List all current friends:
SELECT FUID2 FROM `Friends` WHERE FUID1='1'



FUID2
2

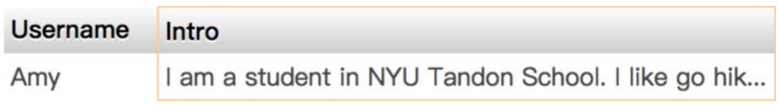
(4). List all friends of friends:

```
SELECT DISTINCT FUID2
FROM `Friends`
WHERE FUID1 in ( SELECT FUID2
                 FROM Friends
                 WHERE FUID1='1') AND FUID1 != '1'
```



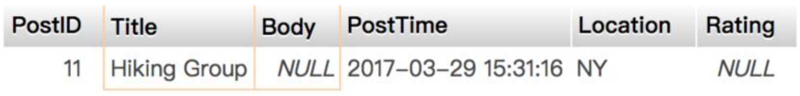
FUID2
3

3-3. Browse/Search Queries:
(1). A user accesses all the queries by his friends:
a. A user accesses profile introduction contains keyword “NYU” of his friends:
SELECT Username, Intro
FROM `User`
WHERE UID in (SELECT FUID2 FROM `Friends` WHERE FUID1='1') and Intro Like '%NYU%'



Username	Intro
Amy	I am a student in NYU Tandon School. I like go hik...

b. A user accesses diaries contains keywords “Hiking” of his friends:
SELECT PostID, Title, Body, PostTime, Location, Rating
FROM `Post` natural join `User`
WHERE UID in (SELECT FUID2 FROM `Friends` WHERE FUID1='1') and Title Like '%Hiking%' or Body Like '%Hiking%'

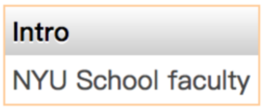


PostID	Title	Body	PostTime	Location	Rating
11	Hiking Group	NULL	2017-03-29 15:31:16	NY	NULL

(2). A user accesses all the queries by his friends of friends:

a. A user accesses profile introduction contains keyword “NYU” of his friends of friends:

```
SELECT Intro
FROM `User`
WHERE UID in (   SELECT DISTINCT FUID2
                 FROM `Friends`
                 WHERE FUID1 in ( SELECT FUID2
                                  FROM Friends
              WHERE FUID1='1') AND FUID2 != 1)
and Intro Like '%NYU%'
```

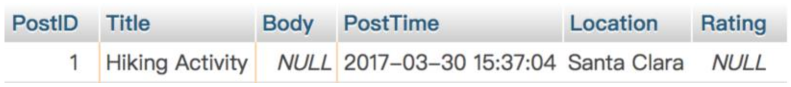


Intro
NYU School faculty

In this example, FUID1’s friend of friends is FUID3, whose introduction contains keyword “NYU”.

b. A user accesses diaries contains keywords “Hiking” of his friends of friends:

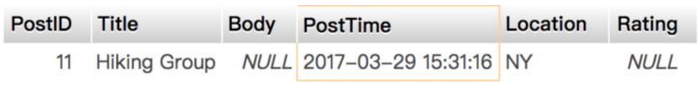
```
SELECT PostID, Title, Body, PostTime, Location, Rating
FROM `Post` natural join `User`
WHERE UID in (   SELECT DISTINCT FUID2
                 FROM `Friends`
                 WHERE FUID1 in ( SELECT FUID2
                                  FROM Friends
              WHERE FUID1='1') AND FUID2 != 1)
and Title Like '%Hiking%' or Body Like '%Hiking%'
```



PostID	Title	Body	PostTime	Location	Rating
1	Hiking Activity	NULL	2017-03-30 15:37:04	Santa Clara	NULL

c. A user want to list all diary entries by his friends during the last week.

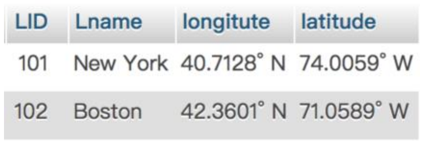
```
SELECT PostID, Title, Body, PostTime, Location, Rating
FROM `Post` natural join `User`
WHERE UID in (SELECT FUID2 FROM `Friends` WHERE FUID1='1') and
PostTime between date_sub(now(),INTERVAL 1 WEEK) and now();
```



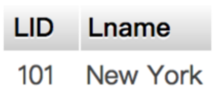
PostID	Title	Body	PostTime	Location	Rating
11	Hiking Group	NULL	2017-03-29 15:31:16	NY	NULL

d. A user want to see all location liked by friends

```
SELECT Distinct LID, Lname
FROM Location natural join Location_like natural join User
WHERE UID in (SELECT FUID2 FROM `Friends` WHERE FUID1='1')
```

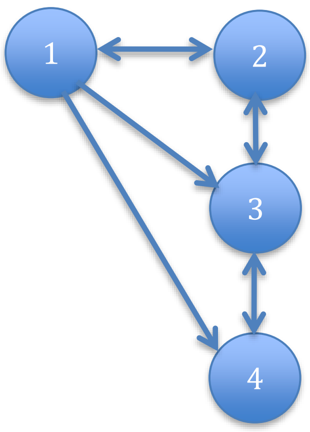


LID	Lname	longitute	latitude
101	New York	40.7128° N	74.0059° W
102	Boston	42.3601° N	71.0589° W



LID	Lname
101	New York

4. Testing map:

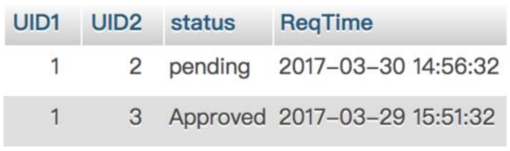


$$\begin{tikzcd}
1 \arrow[r, leftrightarrow] \arrow[dr] \arrow[ddr] & 2 \arrow[d, leftrightarrow] \\
 & 3 \arrow[d, leftrightarrow] \\
 & 4
\end{tikzcd}$$

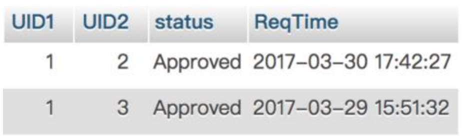
1) Chao Li
2) Amy
3) Li Yang
4) Chris Harrison

We designed a test map shows that 1 and 2 are friends, 2 and 3 are friends, 3 and 4 are friends, 1 sends friend request to 3 and 4, status is pending. Once, 3 or 4 accept the request. The status will change into approved. 3 is a friend of friend of 1.

In the request table:

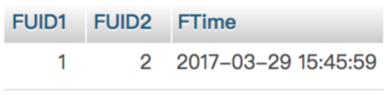


UID1	UID2	status	ReqTime
1	2	pending	2017–03–30 14:56:32
1	3	Approved	2017–03–29 15:51:32



UID1	UID2	status	ReqTime
1	2	Approved	2017–03–30 17:42:27
1	3	Approved	2017–03–29 15:51:32

In the Friends table:



FUID1	FUID2	FTime
1	2	2017–03–29 15:45:59

5. Stored Procedures:

(1). Update password:

```
CREATE PROCEDURE update_profile ( IN Newpassword VARCHAR(45), UID
INT(11))
    BEGIN
    UPDATE `User`
    SET User.password= Newpassword
    WHERE User.UID = UID;
    END
```

(2). Update profile:

```
CREATE PROCEDURE update_profile ( IN Age INT(11), Gender VARCHAR(45),
Photo LONGBLOB, Phone BIGINT, Address VARCHAR(45), Intro Text(100), UID
INT(11))
    BEGIN
    UPDATE `User`
    SET User.Age= Age, User.Gender=Gender, User.Photo=Photo,
User.phone=phone, User.Address=Address, User.Intro=Intro
      WHERE User.UID=UID;
     END
```

(3). Send friendship request:

```
CREATE PROCEDURE friend_request(IN UID1 INT(11), UID2 INT(11))
BEGIN
INSERT INTO `request`
VALUE(UID1, UID2, `pending`, CURRENT_TIME());
END
```

(4). Accept friend request:

```
CREATE PROCEDURE friend_approved(IN UID1 INT(11), UID2 INT(11))
BEGIN
UPDATE `request`
SET STATUS = `approved`
WHERE request.UID1=UID1 and request.UID2=UID2;
INSERT INTO `Friends`
VALUE(UID1, UID2, CURRENT_TIME());
INSERT INTO `Friends`
VALUE(UID2, UID1, CURRENT_TIME());
END
```

(5). Decline friend request:

```
CREATE PROCEDURE friend_decline(IN UID1 INT(11), UID2 INT(11))
BEGIN
UPDATE `request`
SET status = `declined`
WHERE request.UID1=UID1 and request.UID2=UID2;
END
```

(6). Delete friend:

```
CREATE PROCEDURE friend_delete(IN UID1 INT(11), UID2 INT(11))
BEGIN
DELETE FROM `Friends`
```

```
WHERE (Friends.UID1 =UID1 AND Friends.UID2=UID2) or (Friends.UID1 =UID2
AND Friends.UID2=UID1);
END
```

(7). List all my friends name:

```
CREATE PROCEDURE list_friend_name (IN MyUID INT(11), OUT Fname
VARCHAR(45))
BEGIN
SELECT User.Username as Fname
From User
Where User.UID in (SELECT Friends.FUID2
                   From Friends join User on Friends.FUID1 = MyUID);
END
```

(8). Create a Post:

```
CREATE PROCEDURE create_post(IN PostID INT(11), PosterID INT(11),
PrivacyType VARCHAR(45), PostTime DATETIME, Title TEXT, Body
LONGBLOB, Location VARCHAR(45))
BEGIN
INSERT INTO `Post`
VALUE(PostID, PosterID, PrivacyType, PostTime, Title, Body, Location, 0);
END
```