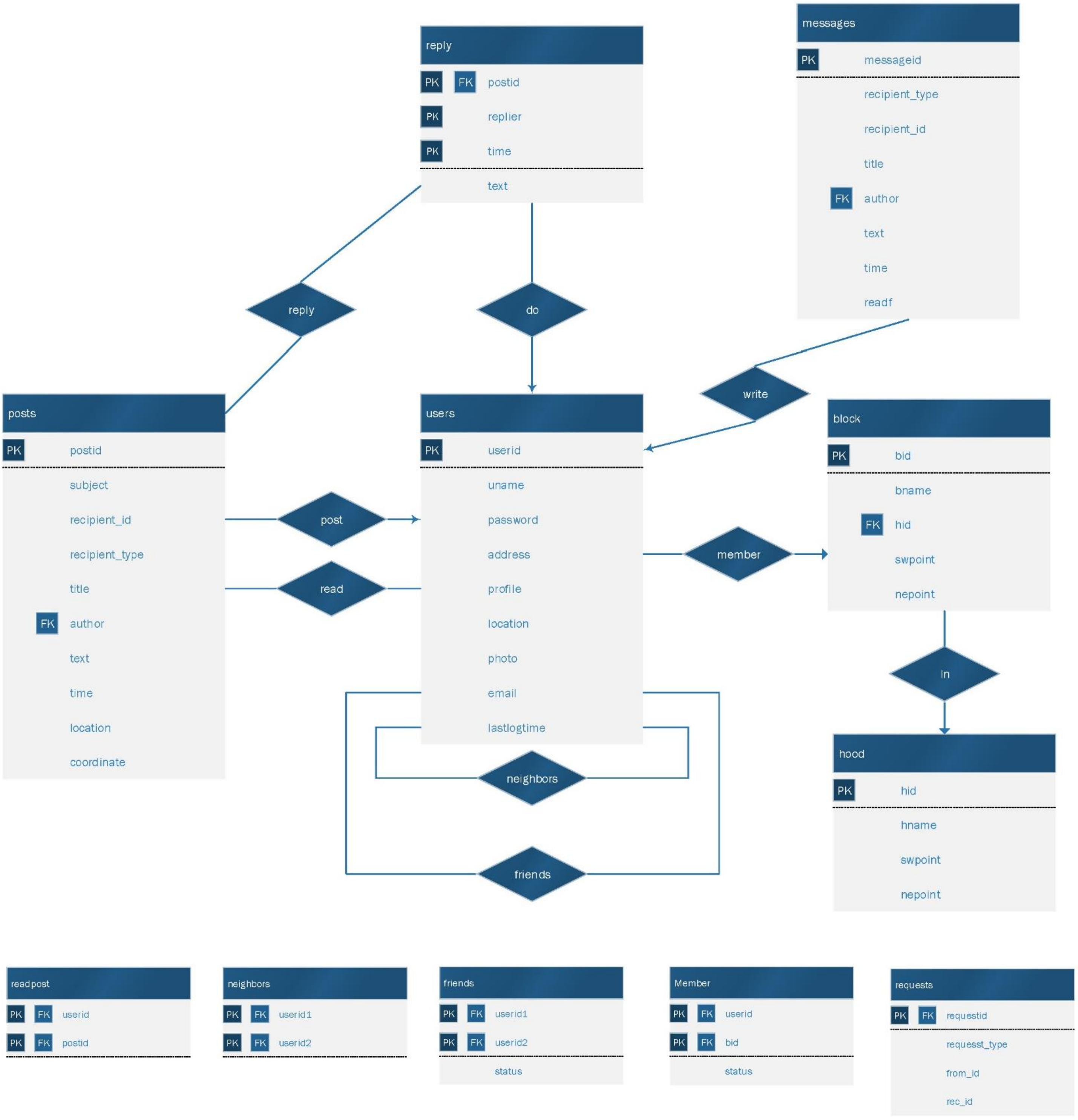
Database Systems

Project

content

Schema

ER diagram



relational schema

users (`userid`, `uname`, `password`, `address`, `profile`, `photo`, `email`, `lastlogtime`)
PRIMARY KEY (`userid`)

This table stores the information of the users. Password is md5 encrypted. lastlogtime is updated every time a user logs out. Users use email and password to log in.

`hood` (`hid`,`hname`,`swpoint`,`nepoint`)
PRIMARY KEY (`hid`)

This table stores the name and the id of the hood. A hood is a rectangle defined by its southwest point and northeast point.

`block` (`bid`,`hid`,`bname`,`swpoint`,`nepoint`)
PRIMARY KEY (`bid`)
FOREIGN KEY (`hid`) REFERENCES `hood` (`hid`)

This table stores the name and the id of the block as well as the id of the hood it belongs to. A block is a rectangle defined by its southwest point and northeast point.

`member` (`userid`,`bid`,`status`)
PRIMARY KEY (`userid`)
FOREIGN KEY (`userid`) REFERENCES `users` (`userid`)
FOREIGN KEY (`bid`) REFERENCES `block` (`bid`)

Member table stores the id of the user and the id of the block he belongs to. The status has four values, empty,1,2,Y. This indicates whether the user is admitted as a member.

`friends` (`userid1`,`userid2`,`status`)
PRIMARY KEY (`userid1`,`userid2`)
FOREIGN KEY (`userid1`) REFERENCES `users` (`userid`)
FOREIGN KEY (`userid2`) REFERENCES `users` (`userid`)

Friends table stores the friend relation between users. There's a constraint that userid1 must be less than userid2. Status has two value, pending and Y which indicates whether the friend request are accepted.

`neighbors` (`userid1`,`userid2`)
PRIMARY KEY (`userid1`,`userid2`)
FOREIGN KEY (`userid1`) REFERENCES `users` (`userid`)
FOREIGN KEY (`userid2`) REFERENCES `users` (`userid`)

Neighbors table stores the neighbor relation between users. Userid1 has added userid2 as a neighbor.

`messages` (`messageid`,`recipient_id`,`title`,`author`,`text`,`time`,`readf`)
PRIMARY KEY (`messageid`)
FOREIGN KEY (`author`) REFERENCES `users` (`userid`)
FOREIGN KEY (`recipient_id`) REFERENCES `users` (`userid`)

A message is private. Author is the sender id. Readf is the read flag indicating whether this message has been read.

```
`requests` (`requestid`,`request_type`,`from_id`,`rec_id`)
PRIMARY KEY (`requestid`)
FOREIGN KEY (`from_id`) REFERENCES `users` (`userid`)
FOREIGN KEY (`rec_id`) REFERENCES `users` (`userid`)
```

Requests stores all the requests. There are two types, block member request and friend request. From_id is the sender id and rec_id is the id of the user who are required to respond to the request.

```
`posts` (`postid`,`subject`,`recipient_type`,`recipient_id`,`title`,`author`,`text`,`time`,`coordinate`)
PRIMARY KEY (`postid`)
FOREIGN KEY (`author`) REFERENCES `users` (`userid`)
```

Posts are group oriented. There are four type of the group which is indicated in recipient_type field, hood, block, friends and neighbors. Recipient_id are set to the block id and hood id if the type are hood and block. While the types are friends and neighbors, the recipient_id is NULL. The subject are predefined. There is an attribute coordinate representing the location of what the post is about if any.

```
`readpost` (`userid`,`postid`)
PRIMARY KEY (`userid`,`postid`)
FOREIGN KEY (`userid`) REFERENCES `users` (`userid`)
FOREIGN KEY (`postid`) REFERENCES `posts` (`postid`)
```

This table includes the information of whether a user has already read the post and the replies of it. Whenever there’s a new reply, the record of that post is deleted except the one of the replier. That is a post with new reply is considered unread to all other users except the user who post that reply.

```
`reply` (`postid`,`replier`,`text`,`time`)
PRIMARY KEY (`postid`,`replier`,`time`)
FOREIGN KEY (`replier`) REFERENCES `users` (`userid`)
FOREIGN KEY (`postid`) REFERENCES `posts` (`postid`)
```

This table stores all replies to all the posts.

Trigger and procedures

-- Trigger on friends

-- Every time a tuple is inserted on friends table, check if userid1 < userid2. If not, switch userid1 and userid2.

```
drop trigger if exists friends_check1;
DELIMITER //
create trigger friends_check1 before insert on `friends`
for each row
begin
    DECLARE userid INT;
    set userid = new.userid2;
    if new.userid1 > new.userid2 then
        set new.userid2 = new.userid1, new.userid1 = userid;
    end if;
end //
DELIMITER ;
```

-- Trigger on member

-- Member_check1 on update. When updating a member, a member approved the request, the status is set to the next state(From '' to 1, from 1 to 2, from 2 to Y). If the number of existing members is less than 3, then check if all the member has approved the request, the status is set to Y. Upon the admission of a new member, insert a new tuple in posts on the block feeds automatically.

```
drop trigger if exists member_check1;
DELIMITER //
create trigger member_check1 before update on `member`
for each row
begin
  if (select count(*)
from member
where bid = new.bid and status = 'Y' ) = new.status + 0 then
set new.status = 'Y';
  if new.status = 'Y' then
    insert into posts (subject,recipient_type,recipient_id,title,author,text) VALUES ('General', 'block',new.bid, 'New member in the block', new.userid,'Hi, I\'m a new member in the block.');
  end if;
end if;
end //
DELIMITER ;
```

-- Member_check2 on insert. When inserting a new tuple in the member table, sending a member

request to all the existing members in the block.

```
use neighborhood;
DELIMITER //
create trigger member_check2 before insert on `member`
for each row
begin
    DECLARE done INT DEFAULT FALSE;
    DECLARE ids INT;
DECLARE cur CURSOR FOR SELECT userid FROM member where bid = new.bid and status = 'Y';
DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    if (select count(*)
from member
where bid = new.bid) < 1 then
set new.status = 'Y';
    else
        OPEN cur;
        ins_loop: LOOP
              FETCH cur INTO ids;
              IF done THEN
                  LEAVE ins_loop;
              END IF;
              INSERT INTO requests (request_type,from_id,rec_id) VALUES ('block',new.userid,ids);
END LOOP;
CLOSE cur;
    end if;

end //
DELIMITER ;
```

-- Member_check3 on delete. When deleting a member (move), delete all the neighbor relation of the user while keep his friends.

```
DELIMITER //
create trigger member_check3 after delete on `member`
for each row
begin
    delete from neighbors where userid1 = old.userid or userid2 = old.userid;
end //
DELIMITER ;
```

-- Trigger on reply

-- Whenever there's new reply, the corresponding post is considered as unread to everyone except the

replier.

```
DELIMITER //
create trigger reply_check1 before insert on `reply`
for each row
begin
    delete from readpost
    where postid = new.postid and userid <> new.replier;
end //
delimiter ;
```

-- Procedure for searching.
-- Search the keywords through all the accessible posts and output the posts containing the keywords. User can specify the field to search, such as subject, tile, text or all the above.

```
drop procedure IF EXISTS postsSearcher;
DELIMITER //
CREATE PROCEDURE postsSearcher (IN uid int(10), searchType enum('subject','title','text','location','all'),
keywords varchar(25))
BEGIN
case searchType

      when 'subject' then
      BEGIN
      select postid,subject,title,uname,time,recipient_type
      from posts,users
      where `subject` like CONCAT('%',keywords,'%') and users.userid = author and ((recipient_type = 'block'
and recipient_id = (select bid from member where userid = uid and status = 'Y'))
            or (recipient_type = 'hood' and recipient_id = (select hid from member natural join block where
userid = uid and status = 'Y'))
           or (recipient_type = 'neighbors' and (uid in (select userid2 from neighbors where userid1 = author)
or uid = author))
            or (recipient_type = 'friends' and (uid in (select userid1 from friends where userid2 = author and
status = 'Y') or uid in (select userid2 from friends where userid1 = author and status = 'Y') or uid = author)));
      END;

      when 'title' then
      BEGIN
      select postid,subject,title,uname,time,recipient_type
      from posts,users
      where `title` like CONCAT('%',keywords,'%') and users.userid = author and ((recipient_type = 'block'
and recipient_id = (select bid from member where userid = uid and status = 'Y'))
            or (recipient_type = 'hood' and recipient_id = (select hid from member natural join block where
userid = uid and status = 'Y'))
```

```
        or (recipient_type = 'neighbors' and (uid in (select userid2 from neighbors where userid1 = author)
or uid = author))
         or (recipient_type = 'friends' and (uid in (select userid1 from friends where userid2 = author and
status = 'Y') or uid in (select userid2 from friends where userid1 = author and status = 'Y') or uid = author)));
    END;

    when 'text' then
    BEGIN
    select postid,subject,title,uname,time,recipient_type
    from posts,users
    where `text` like CONCAT('%',keywords,'%') and users.userid = author and ((recipient_type = 'block'
and recipient_id = (select bid from member where userid = uid and status = 'Y'))
         or (recipient_type = 'hood' and recipient_id = (select hid from member natural join block where
userid = uid and status = 'Y'))
        or (recipient_type = 'neighbors' and (uid in (select userid2 from neighbors where userid1 = author)
or uid = author))
         or (recipient_type = 'friends' and (uid in (select userid1 from friends where userid2 = author and
status = 'Y') or uid in (select userid2 from friends where userid1 = author and status = 'Y') or uid = author)));
    END;

    when 'location' then
    BEGIN
    select postid,subject,title,uname,time,recipient_type
    from posts,users
    where `location` like CONCAT('%',keywords,'%') and users.userid = author and ((recipient_type =
'block' and recipient_id = (select bid from member where userid = uid and status = 'Y'))
         or (recipient_type = 'hood' and recipient_id = (select hid from member natural join block where
userid = uid and status = 'Y'))
        or (recipient_type = 'neighbors' and (uid in (select userid2 from neighbors where userid1 = author)
or uid = author))
         or (recipient_type = 'friends' and (uid in (select userid1 from friends where userid2 = author and
status = 'Y') or uid in (select userid2 from friends where userid1 = author and status = 'Y') or uid = author)));
    END;

    when 'all' then
    BEGIN
    select postid,subject,title,uname,time,recipient_type
    from posts,users
    where (`subject` like CONCAT('%',keywords,'%') or `title` like CONCAT('%',keywords,'%') or `text` like
CONCAT('%',keywords,'%')) or `location` like CONCAT('%',keywords,'%') and users.userid = author and
((recipient_type = 'block' and recipient_id = (select bid from member where userid = uid and status = 'Y'))
         or (recipient_type = 'hood' and recipient_id = (select hid from member natural join block where
userid = uid and status = 'Y'))
        or (recipient_type = 'neighbors' and (uid in (select userid2 from neighbors where userid1 = author)
or uid = author))
                or (recipient_type = 'friends' and (uid in (select userid1 from friends where userid2 = author and status = 'Y') or uid in (select userid2 from friends where userid1 = author and status = 'Y') or uid = author)));
        END;
        END CASE;
END //
DELIMITER ;
```

-- Procedure for registering a new user. Check if the email is already registered.

```
use neighborhood;
drop procedure IF EXISTS addUser;
DELIMITER //
CREATE PROCEDURE addUser (IN inEmail varchar(32), inName varchar(32), inPassword varchar(32), inAddress varchar(32), inLat float, inLong float, inProfile text )
BEGIN
        declare userAdd bool;
        set @inLocation =POINT(inLat, inLong);
        if((select userid from users where email = inEmail) is NULL) then
                BEGIN
                        insert into users (`uname`,`password`,`address`,`profile`,`email`,`locationpoint`) values(inName,inPassword,inAddress,inProfile,inEmail,@inLocation);
                        select userid from users where email = inEmail;
                END;
        else
                BEGIN
                        select FALSE;
                END;
        end if;

END //
DELIMITER ;
```

-- Procedure for sending friend request. Insert a tuple into friends table and a request into the request table.

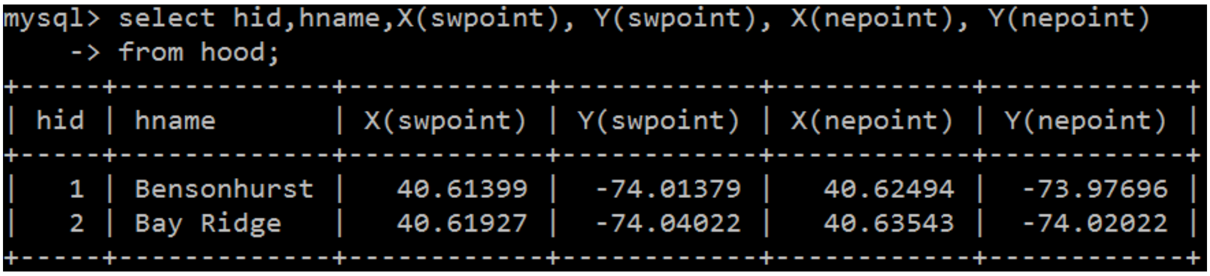
```
DELIMITER //
CREATE PROCEDURE addfriends (IN userid1 INT, IN userid2 INT)
BEGIN
        insert into `friends` (`userid1`,`userid2`) VALUES (userid1,userid2);
        insert into `requests` (`request_type`,`from_id`,`rec_id`) VALUES ('friends',userid1,userid2);
END //
DELIMITER ;
```

-- Procedure for adding neighbors. Insert a tuple into neighbors table and a message into messages table to inform the user that he has been added as a neighbor.

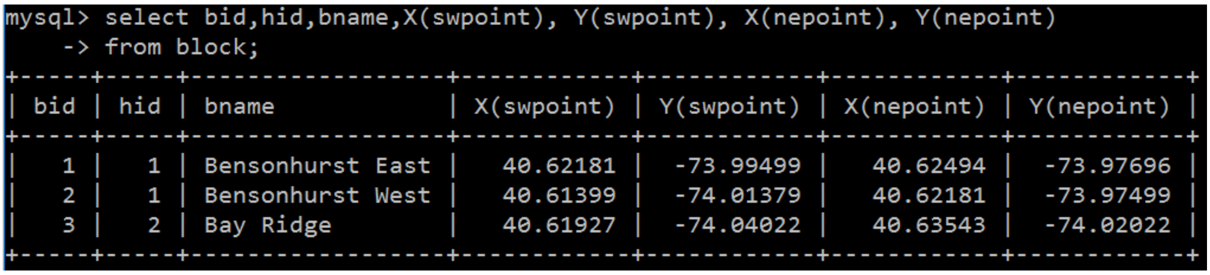
```
drop procedure IF EXISTS addneighbors;
DELIMITER //
CREATE PROCEDURE addneighbors (IN userid1 INT, IN userid2 INT)
BEGIN
    insert into `neighbors` (`userid1`,`userid2`) VALUES (userid1,userid2);
    insert into `messages` (`recipient_id`,`title`,`author`,`text`) VALUES (userid2, 'New Neighbor', userid1,
 'I have added you as my neighbor.');
END //
DELIMITER ;
```

Description

hood and block



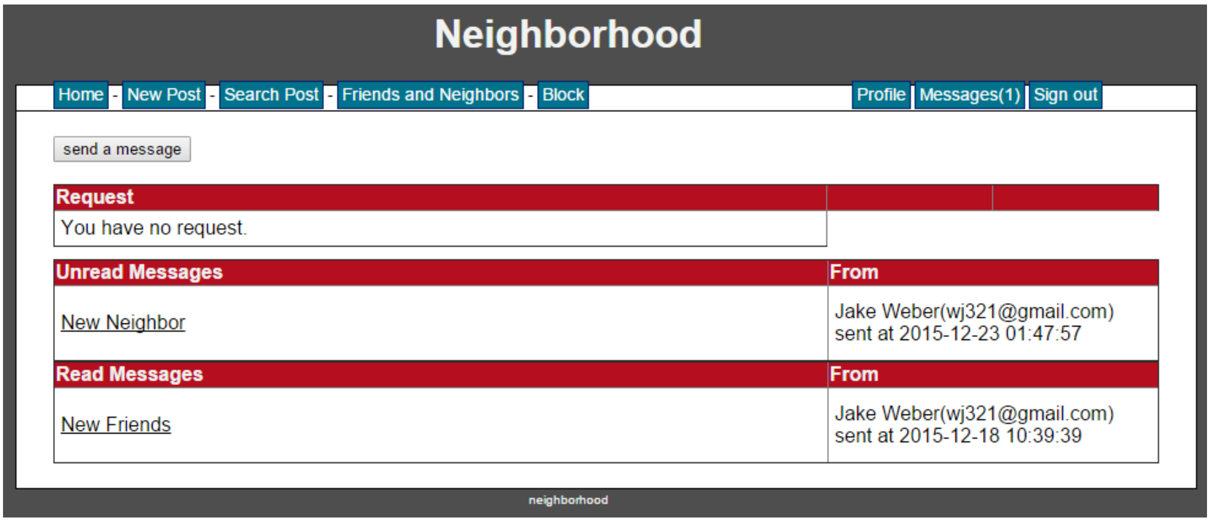
```
mysql> select hid,hname,X(swpoint), Y(swpoint), X(nepoint), Y(nepoint)
    -> from hood;
+-----+-------------+------------+------------+------------+------------+
| hid | hname       | X(swpoint) | Y(swpoint) | X(nepoint) | Y(nepoint) |
+-----+-------------+------------+------------+------------+------------+
|   1 | Bensonhurst |   40.61399 |  -74.01379 |   40.62494 |  -73.97696 |
|   2 | Bay Ridge   |   40.61927 |  -74.04022 |   40.63543 |  -74.02022 |
+-----+-------------+------------+------------+------------+------------+
```



```
mysql> select bid,hid,bname,X(swpoint), Y(swpoint), X(nepoint), Y(nepoint)
    -> from block;
+-----+-----+------------------+------------+------------+------------+------------+
| bid | hid | bname            | X(swpoint) | Y(swpoint) | X(nepoint) | Y(nepoint) |
+-----+-----+------------------+------------+------------+------------+------------+
|   1 |   1 | Bensonhurst East |   40.62181 |  -73.99499 |   40.62494 |  -73.97696 |
|   2 |   1 | Bensonhurst West |   40.61399 |  -74.01379 |   40.62181 |  -73.97499 |
|   3 |   2 | Bay Ridge        |   40.61927 |  -74.04022 |   40.63543 |  -74.02022 |
+-----+-----+------------------+------------+------------+------------+------------+
```

In our schema, the blocks and hoods are predefined and assigned to a unique id. Every block belongs to a hood. Blocks and hoods are rectangles defined by their southwest points and northeast points.

message,post,reply

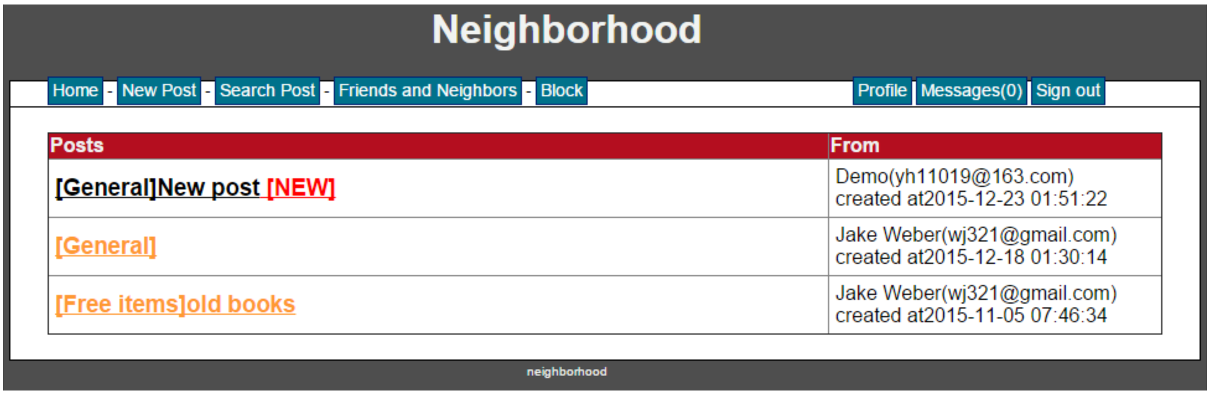


Request		
You have no request.		

Unread Messages	From
New Neighbor	Jake Weber(wj321@gmail.com) sent at 2015-12-23 01:47:57

Read Messages	From
New Friends	Jake Weber(wj321@gmail.com) sent at 2015-12-18 10:39:39

Messages is categorized into request, unread messages, read messages.

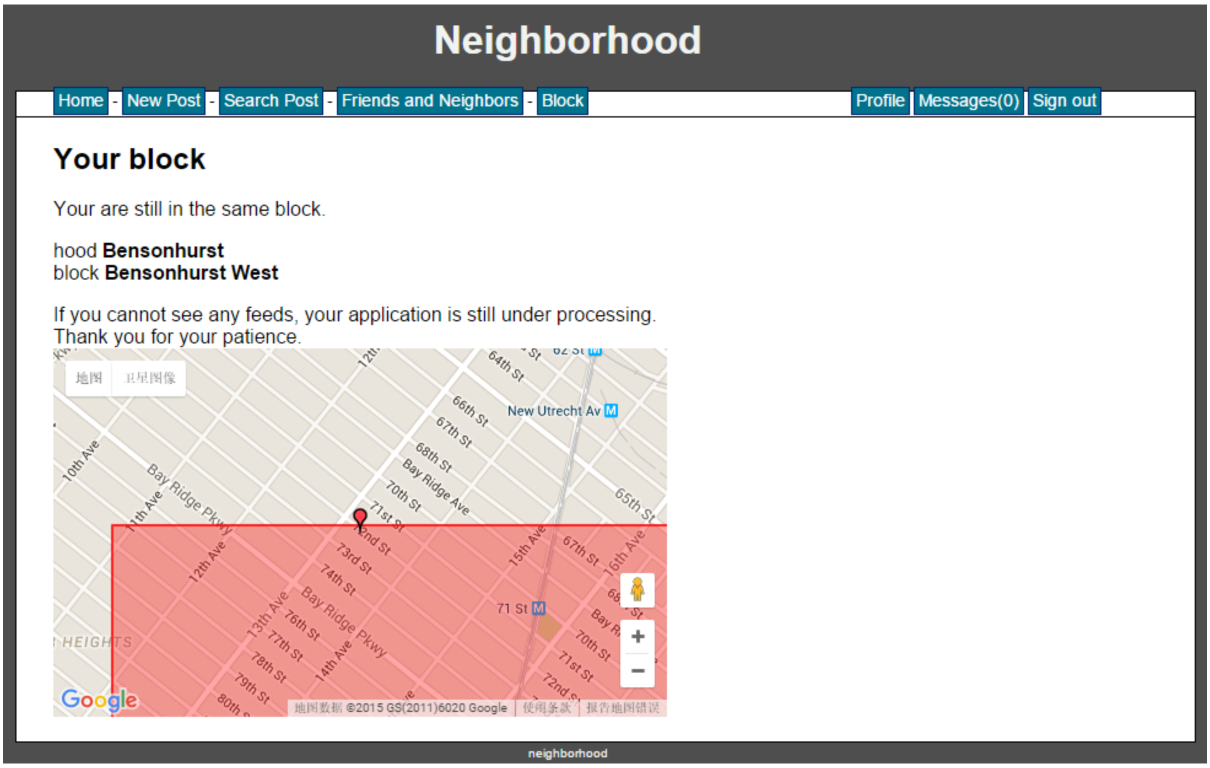


Posts	From
[General]New post [NEW]	Demo(yh11019@163.com) created at2015-12-23 01:51:22
[General]	Jake Weber(wj321@gmail.com) created at2015-12-18 01:30:14
[Free items]old books	Jake Weber(wj321@gmail.com) created at2015-11-05 07:46:34

There are new posts, old posts when the creation time is concerned. There are read posts and unread posts when the action of the user is concerned.

The messages in our system are categorized into 3 types. A post is posted to a group people (hood, block, friends, or neighbors). A reply is a reply to the post, which can be seen by all the people specified by the original post. A message is private message between two users while a sender and unlike a post reply, a reply to a message is also a message. So the post is assigned a postid while reply only have the original postid and a time so as to be displayed in order of time. When a user click on a message or a post, the message or the post is considered as read (orange colored title). If a new reply is added to the post, the post is considered as unread to everyone except the replier. And a post created after the user's last logout time is considered as a new post.

member



Your block

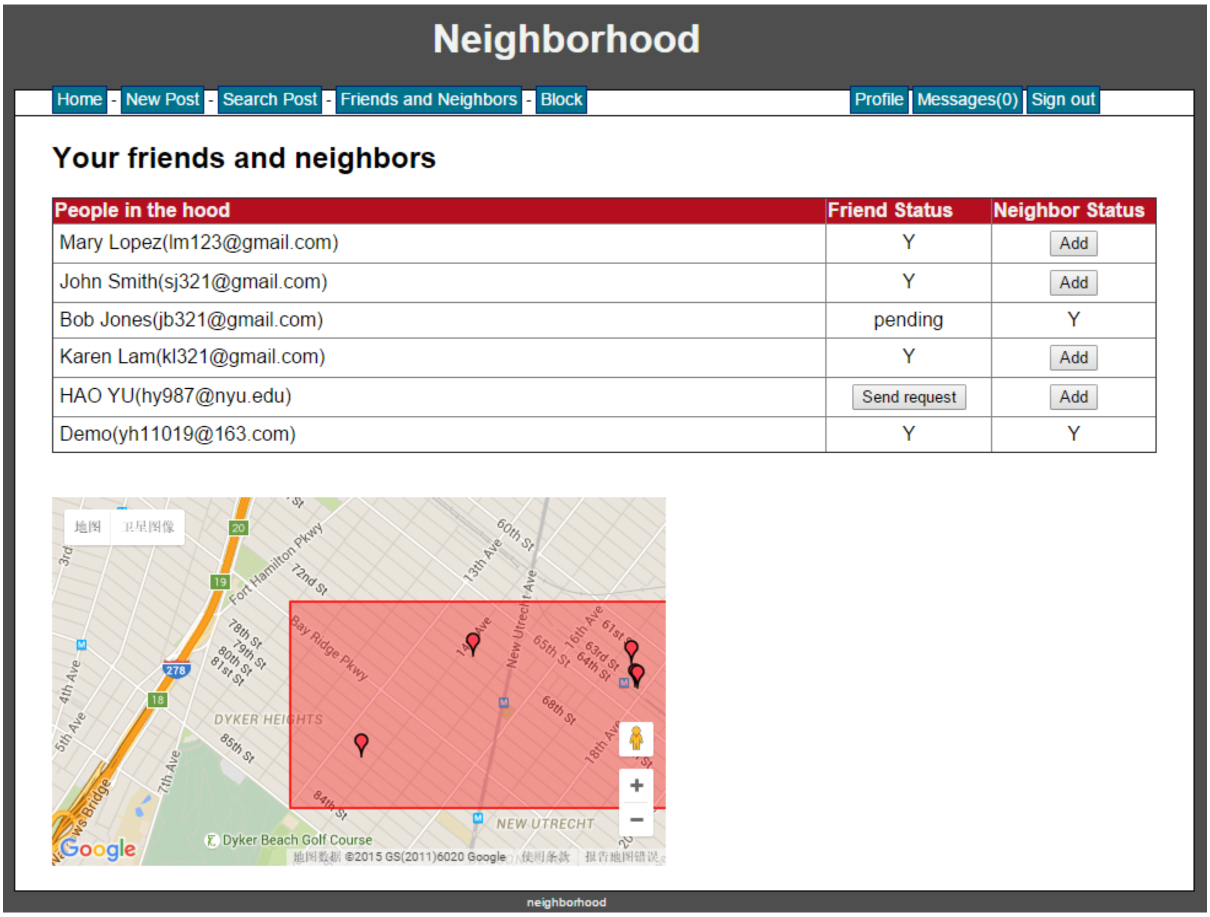
Your are still in the same block.

hood **Bensonhurst**
block **Bensonhurst West**

If you cannot see any feeds, your application is still under processing.
Thank you for your patience.

When a user registers in our system, the name, address, password, and a unique email are required. And a unique user id will be assigned to the user automatically. The user is not a member of any block at first. And according to the address the user provide, we'll recommend a block for the user. Then after the user apply for the membership of the block, requests will be sent to all the current members in the block. And a tuple with the user id and block id is inserted into member table but with status set to ``. A member then choose yes or no to indicate whether to accept the user as a member. When there are 3 approval or all the members in a block in the case that there are less than 3 members in the block, the status of the corresponding tuple in the member table will be updated to ‘Y’, which suggests the membership is established. Upon the admission, we will send a post on the block feeds whose author is set as the new member. This post functions as a notification of a new member.

relationship



Neighborhood

Home - New Post - Search Post - Friends and Neighbors - Block
Profile
Messages(0)
Sign out

Your friends and neighbors

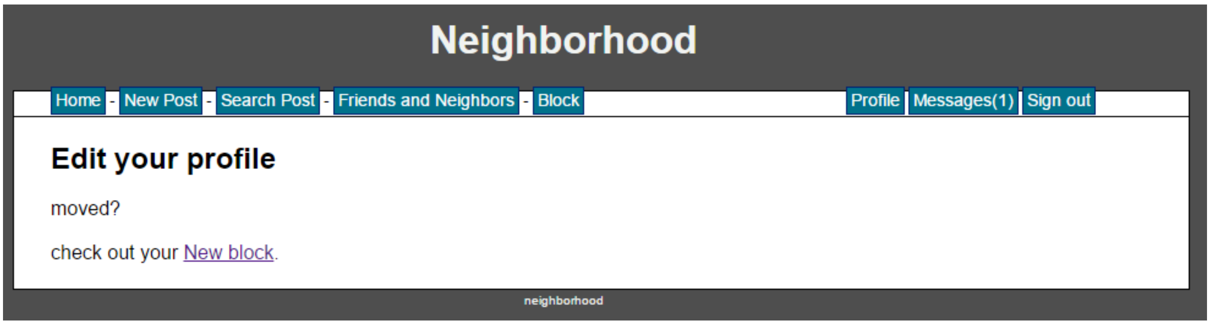
People in the hood	Friend Status	Neighbor Status
Mary Lopez(lm123@gmail.com)	Y	Add
John Smith(sj321@gmail.com)	Y	Add
Bob Jones(jb321@gmail.com)	pending	Y
Karen Lam(kl321@gmail.com)	Y	Add
HAO YU(hy987@nyu.edu)	Send request	Add
Demo(yh11019@163.com)	Y	Y

neighborhood

A user with membership can friend other users in the same hood and specify neighbors in the same hood. To friend another user, the user select the person from a list which he can friend and then a friend request will be sent to ask the person whether or not to accept the request. And a tuple of the pair of user ids is inserted into the friends table and the status is defaulted to ‘pending’. Once the request is approved, the status is updated to ‘Y’ which indicates the friendship is valid and a private message will be sent to the user who initiate the request to inform him of the approval. Otherwise, the tuple will be deleted. As for neighbors, the relationship is chosen unilaterally. A user can specify any user in the same hood as a neighbor. After the user add a neighbor, a message will be sent to the person added by the user to notify him the action.

After a user moved to another hood, the neighbors of the user are lost while the user still keeps his friends.

moving



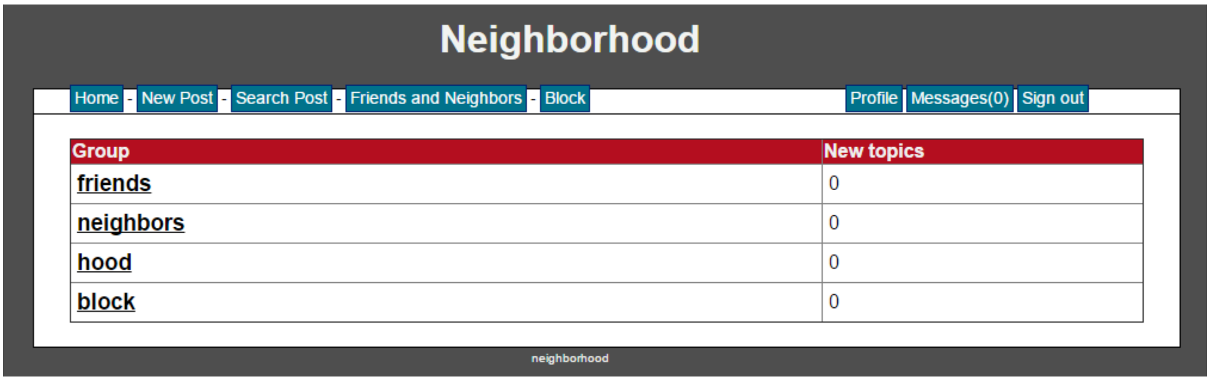
Edit your profile

moved?

check out your New block.

When user changes his address in his profile, the system will check if the user are still in the same block. If user is still in the same block, everything remains unchanged. If the user moves to another block, the messages are kept while the previous posts cannot be accessed any more. Also the user lost all his neighbor relations while keeping the friend relations. So the user can still see the previous friend feeds. But not until he gets the admission to the new block. Because we only give content access to the user who has a block membership. But the user can still send and receive messages.

general use case



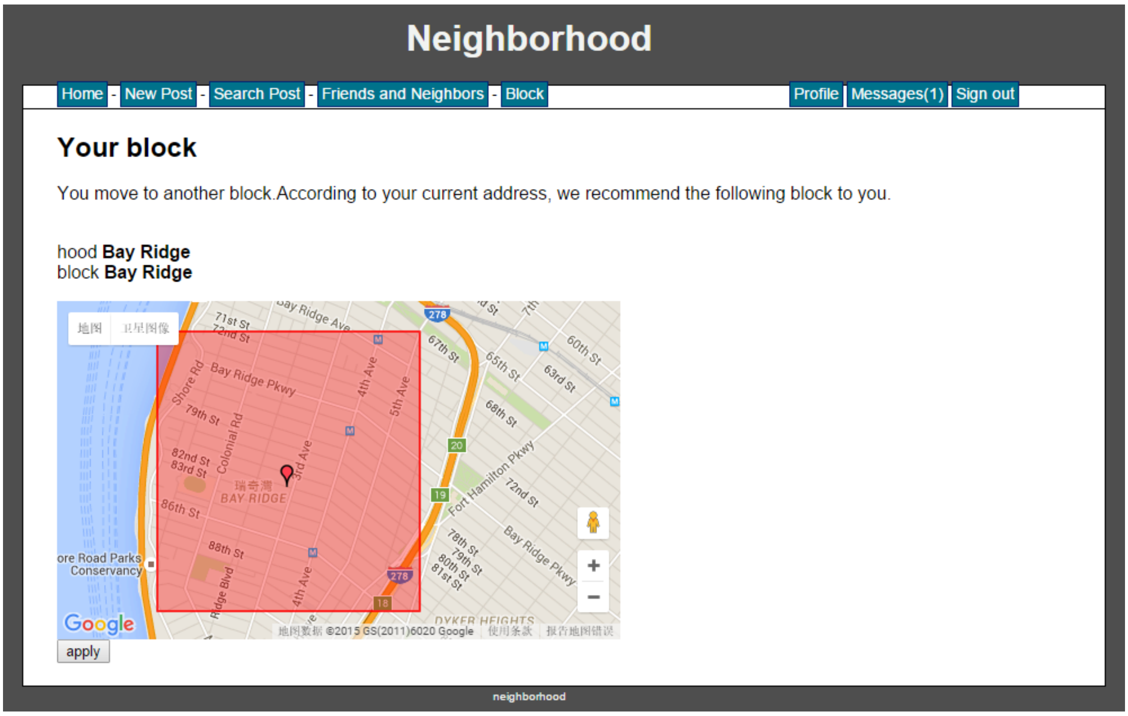
Group	New topics
friends	0
neighbors	0
hood	0
block	0

A user uses his email and password to log in. After that, the user is directed to a page presenting group category of the posts. A user can start a new post after choose a subject from a predefined set of subjects and specify a title and add content in text. Also user can review a post and reply. A user can do some operations on the posts, like search the keywords. Messages are kept in a mailbox. Every time a user logs out, the lastlogtime is updated in the users table.

Improvement

1. move

We have a bug during demo that a user move to another block. But still retain its membership in previous block. We have fixed that. After a user changes the address and moves out of the block (check that the new address not in the same block), the membership in the previous block is deleted from the member table and at this time the user is same as a newly registered user except that he keeps his friend relations.



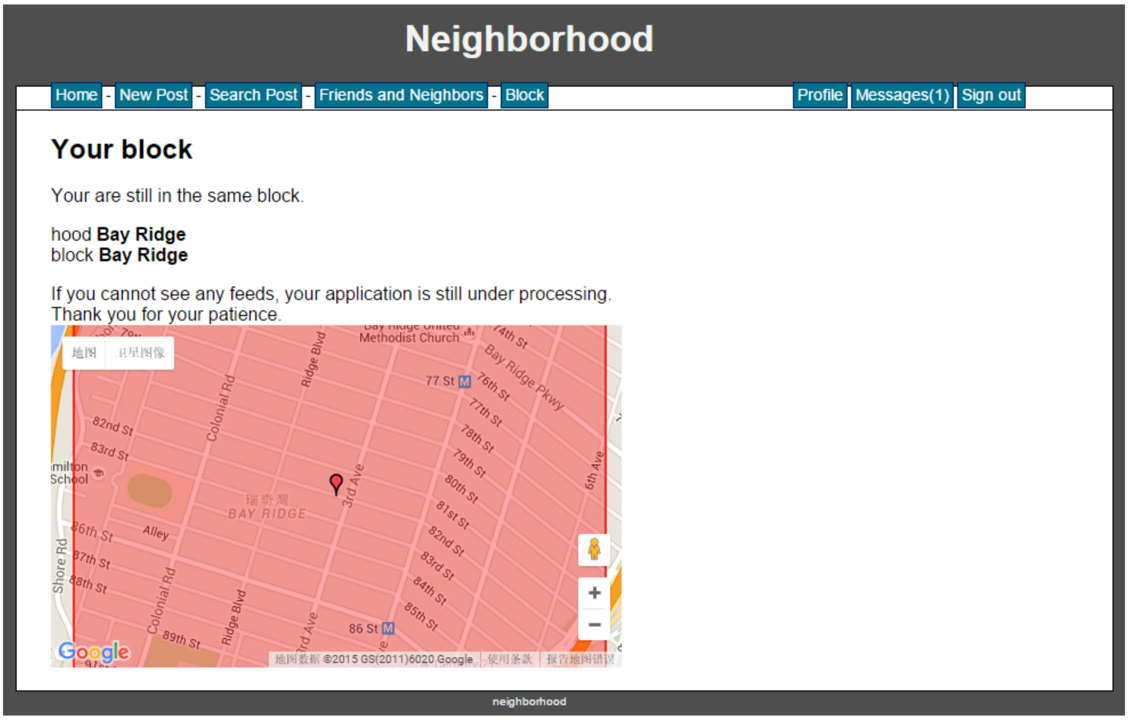
Neighborhood

Your block

You move to another block.According to your current address, we recommend the following block to you.

hood **Bay Ridge**
block **Bay Ridge**

neighborhood



Neighborhood

Your block

Your are still in the same block.

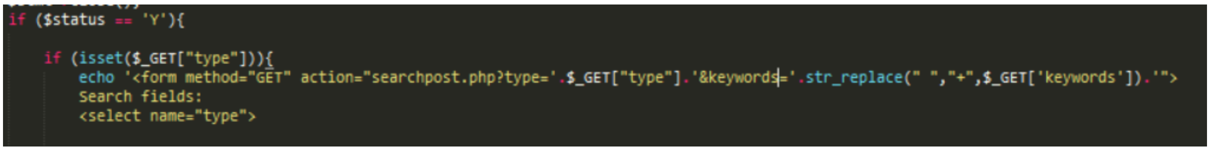
hood **Bay Ridge**
block **Bay Ridge**

If you cannot see any feeds, your application is still under processing.
Thank you for your patience.

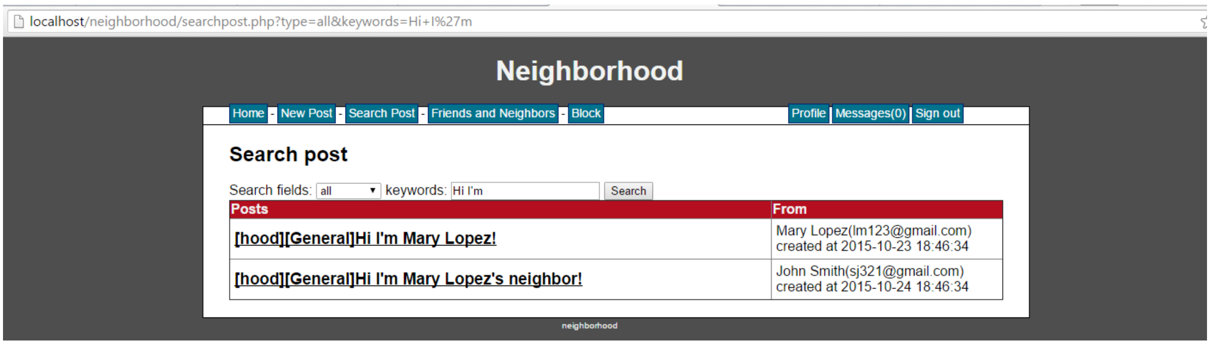
neighborhood

2. search page bookmark

We assign a url to every search result page.   Now a user can bookmark a search result!



```
if ($status == 'Y'){

    if (isset($_GET["type"])){
        echo '<form method="GET" action="searchpost.php?type='.$_GET["type"].'&keywords='.str_replace(" ","+",$_GET['keywords']).'">
        Search fields:
        <select name="type">
```



localhost/neighborhood/searchpost.php?type=all&keywords=Hi+I%27m

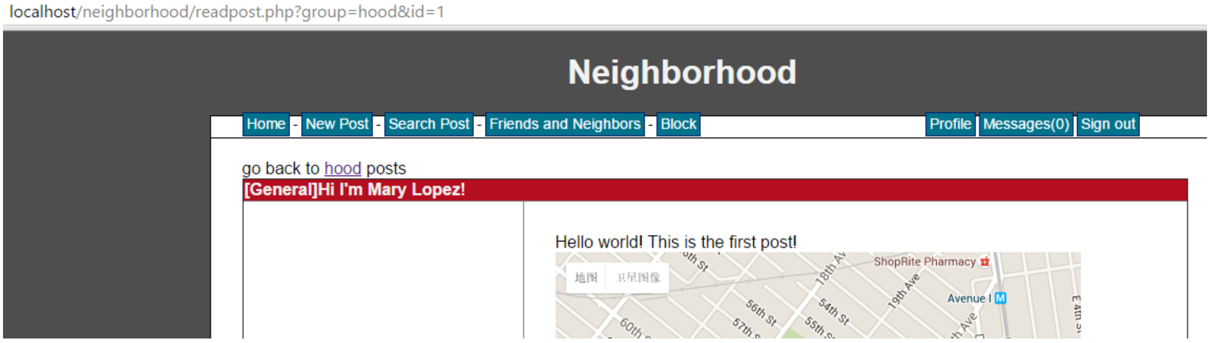
Search post

Search fields: all keywords: Hi I'm Search

Posts	From
[hood][General]Hi I'm Mary Lopez!	Mary Lopez(lm123@gmail.com) created at 2015-10-23 18:46:34
[hood][General]Hi I'm Mary Lopez's neighbor!	John Smith(sj321@gmail.com) created at 2015-10-24 18:46:34

3. check the user's privilege when accessing a message or a post

When user click the tile of a post, a postid is passed to the readpost page. And then the system will retrieve the post content and replies. Before that we only check whether the session user id is set but not whether this user has the privilege to access the content.



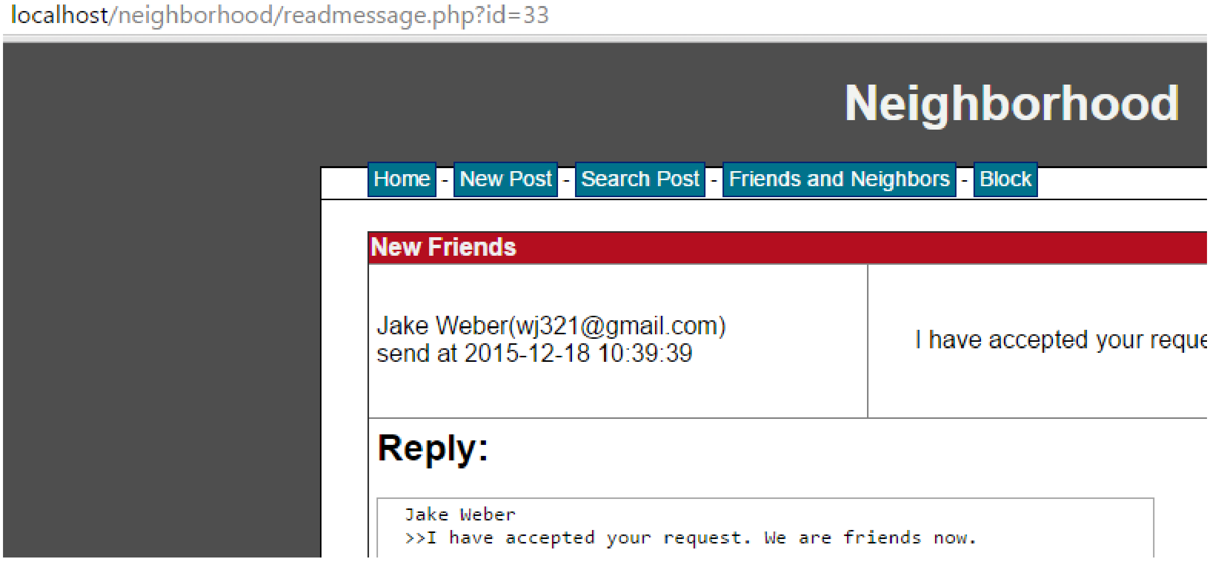
localhost/neighborhood/readpost.php?group=hood&id=1

go back to hood posts

[General]Hi I'm Mary Lopez!

Hello world! This is the first post!

The same with messages.



localhost/neighborhood/readmessage.php?id=33

New Friends

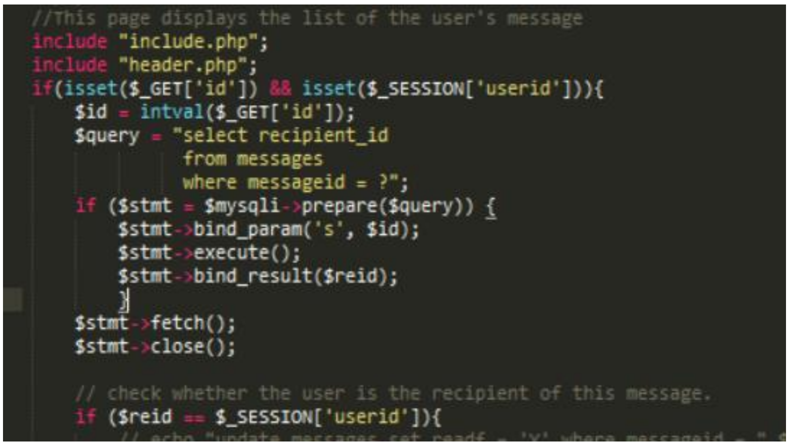
Jake Weber(wj321@gmail.com) send at 2015-12-18 10:39:39	I have accepted your reque

Reply:

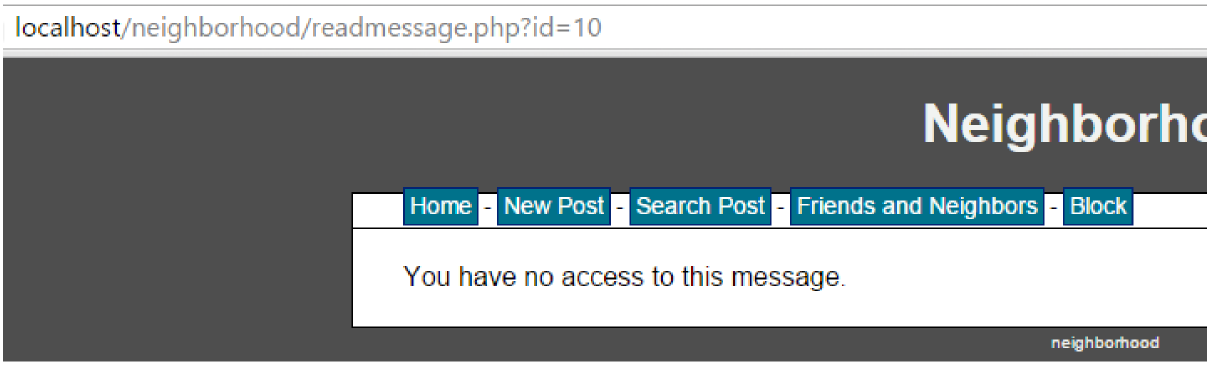
```
Jake Weber
>>I have accepted your request. We are friends now.
```

So a signed in user can access those posts or messages that are not supposed to be visible to him (those don’t show up in his post list) by simply modifying the url.

After adding the check, we block the posts and messages from the users who are not the recipient of the content.



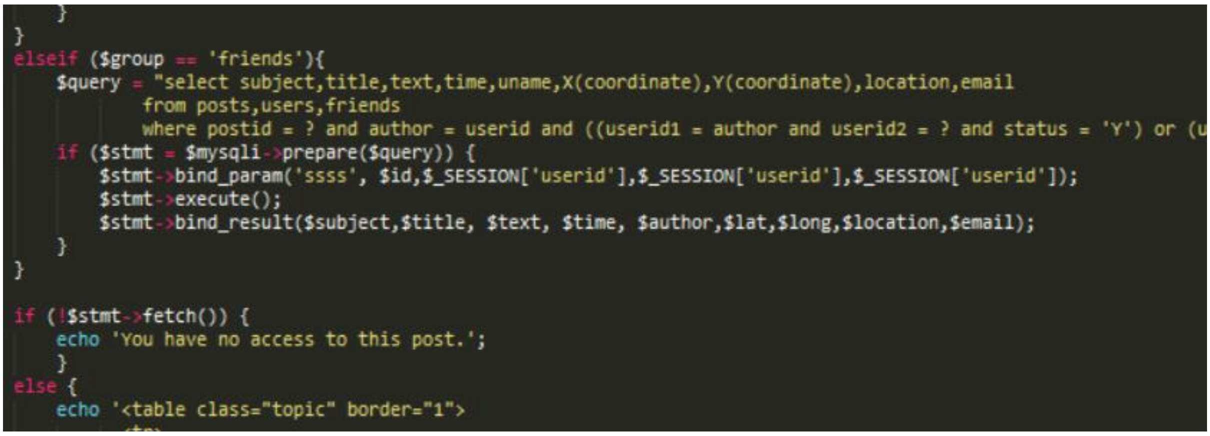
```
//This page displays the list of the user's message
include "include.php";
include "header.php";
if(isset($_GET['id']) && isset($_SESSION['userid'])){
    $id = intval($_GET['id']);
    $query = "select recipient_id
              from messages
              where messageid = ?";
    if ($stmt = $mysqli->prepare($query)) {
        $stmt->bind_param('s', $id);
        $stmt->execute();
        $stmt->bind_result($reid);
        }
    $stmt->fetch();
    $stmt->close();

    // check whether the user is the recipient of this message.
    if ($reid == $_SESSION['userid']){
```



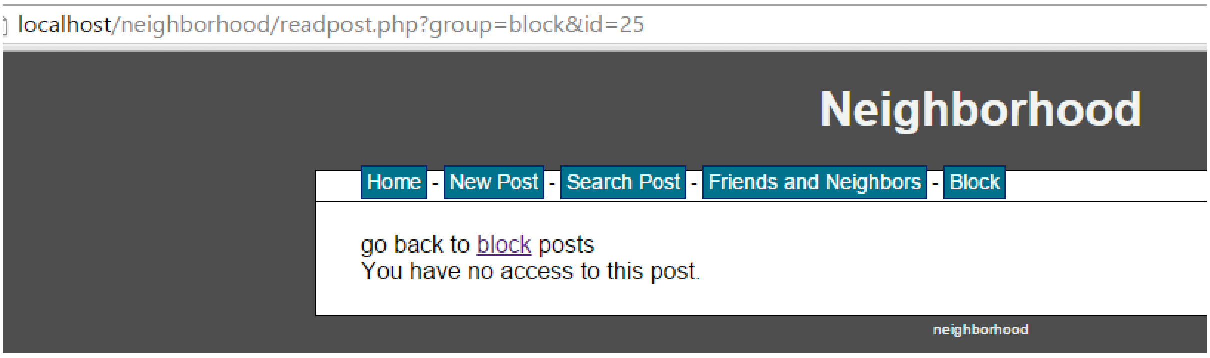
localhost/neighborhood/readmessage.php?id=10

Neighborho
Home - New Post - Search Post - Friends and Neighbors - Block

You have no access to this message.



```
        }
}
elseif ($group == 'friends'){
    $query = "select subject,title,text,time,uname,X(coordinate),Y(coordinate),location,email
            from posts,users,friends
            where postid = ? and author = userid and ((userid1 = author and userid2 = ? and status = 'Y') or (u
    if ($stmt = $mysqli->prepare($query)) {
        $stmt->bind_param('ssss', $id,$_SESSION['userid'],$_SESSION['userid'],$_SESSION['userid']);
        $stmt->execute();
        $stmt->bind_result($subject,$title, $text, $time, $author,$lat,$long,$location,$email);
    }
}

if (!$stmt->fetch()) {
    echo 'You have no access to this post.';
    }
else {
    echo '<table class="topic" border="1">
```



localhost/neighborhood/readpost.php?group=block&id=25

Neighborhood
Home - New Post - Search Post - Friends and Neighbors - Block

go back to block posts
You have no access to this post.