Link to GitHub repository:

https://github.com/NabiWu/finstagram

Date of last commit to repository:

12/9/2019

Number of Team members:

3

Names and netIDs of Team members (one per line):

Bohan Wu bw1417

Kevin Chang kc3192

Trang Duong tpd259

Feature 1:

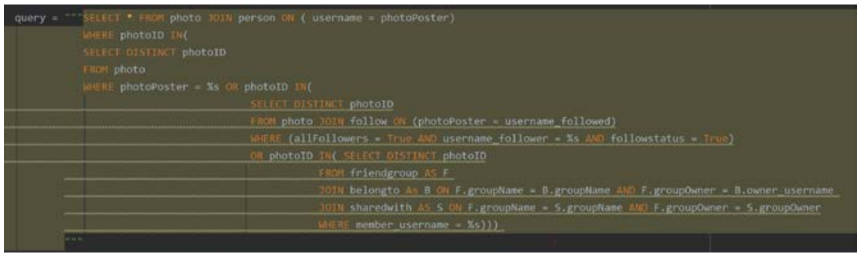
View visible photos

Team Member in charged:

Bohan, Kevin, Trang

Queries:

This query helps us to find the visible photo by a given username.



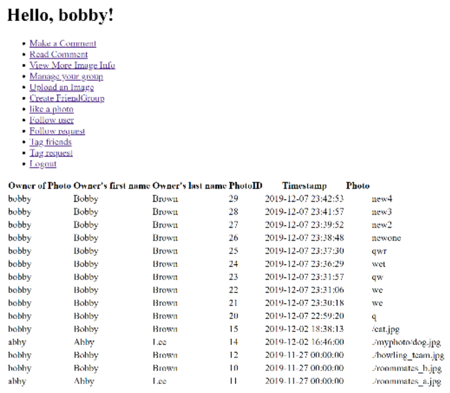
```
query = """SELECT * FROM photo JOIN person ON ( username = photoPoster)
        WHERE photoID IN(
        SELECT DISTINCT photoID
        FROM photo
        WHERE photoPoster = %s OR photoID IN(
                                    SELECT DISTINCT photoID
                                    FROM photo JOIN follow ON (photoPoster = username_followed)
                                    WHERE (allFollowers = True AND username_follower = %s AND followstatus = True)
                                    OR photoID IN( SELECT DISTINCT photoID
                                            FROM friendgroup AS F
                                            JOIN belongto As B ON F.groupName = B.groupName AND F.groupOwner = B.owner_username
                                            JOIN sharedwith AS S ON F.groupName = S.groupName AND F.groupOwner = S.groupOwner
                                            WHERE member_username = %s)))
    """
```

Related Code and files:

Line 40 – 78 in app.py

Finstagram/templatesPart4/home.html

Feature explaination:



Hello, bobby!

- Make a Comment
- Read Comment
- View More Image Info
- Manage your group
- Upload an Image
- Create FriendGroup
- like a photo
- Follow user
- Follow request
- Tag friends
- Tag request
- Logout

Owner of Photo	Owner's first name	Owner's last name	PhotoID	Timestamp	Photo
bobby	Bobby	Brown	29	2019-12-07 23:42:53	new4
bobby	Bobby	Brown	28	2019-12-07 23:41:57	new3
bobby	Bobby	Brown	27	2019-12-07 23:39:52	new2
bobby	Bobby	Brown	26	2019-12-07 23:38:48	newone
bobby	Bobby	Brown	25	2019-12-07 23:37:30	qwr
bobby	Bobby	Brown	24	2019-12-07 23:36:29	wet
bobby	Bobby	Brown	23	2019-12-07 23:31:57	qw
bobby	Bobby	Brown	22	2019-12-07 23:31:06	we
bobby	Bobby	Brown	21	2019-12-07 23:30:18	we
bobby	Bobby	Brown	20	2019-12-07 22:59:20	q
bobby	Bobby	Brown	15	2019-12-02 18:38:13	/cat.jpg
abby	Abby	Lee	14	2019-12-02 16:46:00	./myphoto/dog.jpg
bobby	Bobby	Brown	12	2019-11-27 00:00:00	./bowling_team.jpg
bobby	Bobby	Brown	10	2019-11-27 00:00:00	./roommates_b.jpg
abby	Abby	Lee	11	2019-11-27 00:00:00	./roommates_a.jpg

This is the homepage, while user log in into finstagram. They will view all the photo being visible to them in reverse chronological order. Finstagram also shows the photoID and the photoPoster of each photo.

Feature 2:

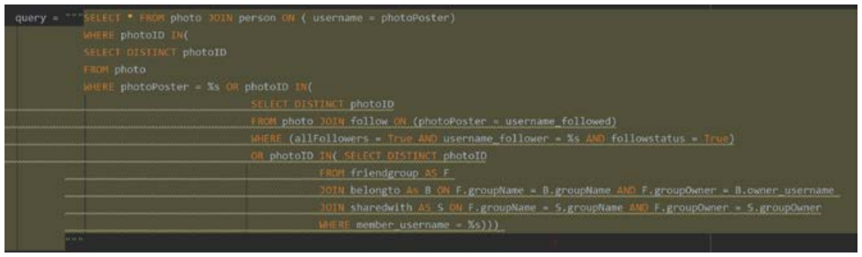
View further photo info

Team Member in charged:

Bohan, Kevin, Trang

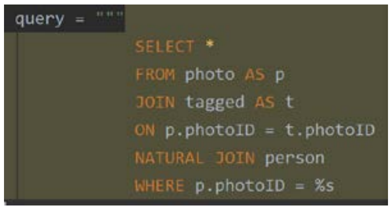
Queries:

This query helps us to find the visible photo by a given username.



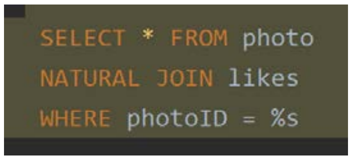
```
query = """SELECT * FROM photo JOIN person ON ( username = photoPoster)
          WHERE photoID IN(
          SELECT DISTINCT photoID
          FROM photo
          WHERE photoPoster = %s OR photoID IN(
                                  SELECT DISTINCT photoID
                                  FROM photo JOIN follow ON (photoPoster = username_followed)
                                  WHERE (allFollowers = True AND username_follower = %s AND followstatus = True)
                                  OR photoID IN( SELECT DISTINCT photoID
                                          FROM friendgroup AS F
                                          JOIN belongto As B ON F.groupName = B.groupName AND F.groupOwner = B.owner_username
                                          JOIN sharedwith AS S ON F.groupName = S.groupName AND F.groupOwner = S.groupOwner
                                          WHERE member_username = %s)))
        """
```

This query helps us to find the tag information



```
query = """
            SELECT *
            FROM photo AS p
            JOIN tagged AS t
            ON p.photoID = t.photoID
            NATURAL JOIN person
            WHERE p.photoID = %s
```

This query helps us to find the like and rating information:



```
SELECT * FROM photo
NATURAL JOIN likes
WHERE photoID = %s
```

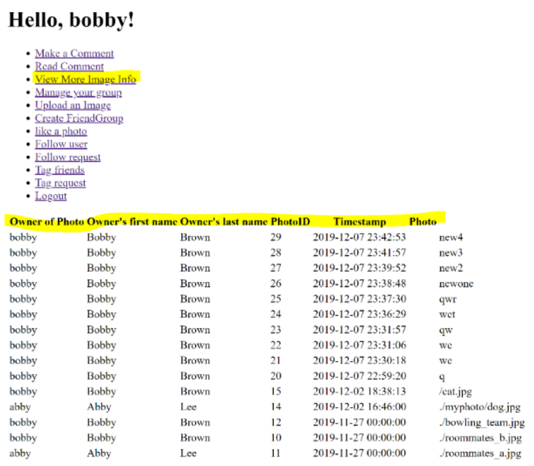
Related Code and Files:

Line 58 – 61, 452 – 538 in app.py

Finstagram/templatesPart4:

- ImgLikeInfo.html
- ImgLikeTable.html
- ImgTagInfo.html
- ImgTagTable.html
- additionalPicInfo.html

Feature explaination:



Hello, bobby!

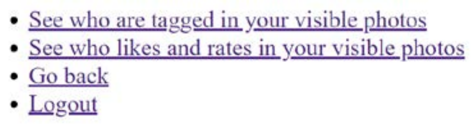
- Make a Comment
- Read Comment
- View More Image Info
- Manage your group
- Upload an Image
- Create FriendGroup
- like a photo
- Follow user
- Follow request
- Tag friends
- Tag request
- Logout

Owner of Photo	Owner's first name	Owner's last name	PhotoID	Timestamp	Photo
bobby	Bobby	Brown	29	2019-12-07 23:42:53	new4
bobby	Bobby	Brown	28	2019-12-07 23:41:57	new3
bobby	Bobby	Brown	27	2019-12-07 23:39:52	new2
bobby	Bobby	Brown	26	2019-12-07 23:38:48	newone
bobby	Bobby	Brown	25	2019-12-07 23:37:30	qwr
bobby	Bobby	Brown	24	2019-12-07 23:36:29	wet
bobby	Bobby	Brown	23	2019-12-07 23:31:57	qw
bobby	Bobby	Brown	22	2019-12-07 23:31:06	we
bobby	Bobby	Brown	21	2019-12-07 23:30:18	we
bobby	Bobby	Brown	20	2019-12-07 22:59:20	q
bobby	Bobby	Brown	15	2019-12-02 18:38:13	/cat.jpg
abby	Abby	Lee	14	2019-12-02 16:46:00	./myphoto/dog.jpg
bobby	Bobby	Brown	12	2019-11-27 00:00:00	./bowling_team.jpg
bobby	Bobby	Brown	10	2019-11-27 00:00:00	./roommates_b.jpg
abby	Abby	Lee	11	2019-11-27 00:00:00	./roommates_a.jpg

The above screenshot shows that finstagram provide:

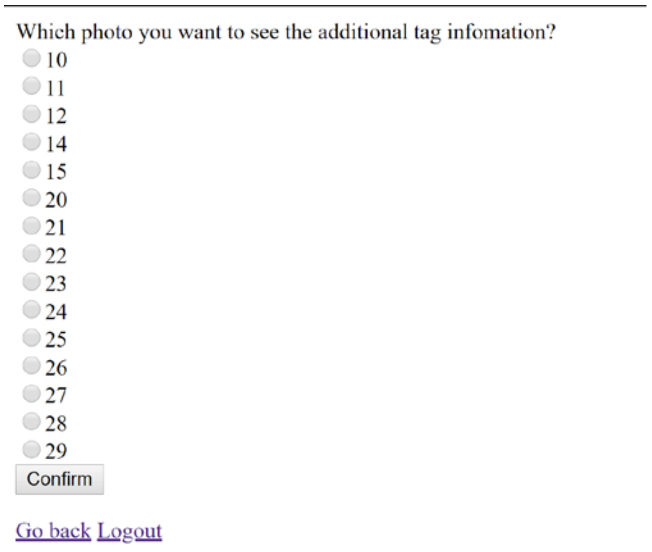
- The firstName and lastName of the photoPoster
- The timestamp,
- The filePath to display it

By clicking the View More Image info, user will jump to here:



- See who are tagged in your visible photos
- See who likes and rates in your visible photos
- Go back
- Logout

By clicking the first link, user will see:

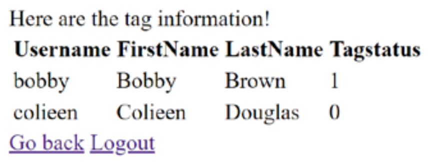


Which photo you want to see the additional tag infomation?

- ○ 10
- ○ 11
- ○ 12
- ○ 14
- ○ 15
- ○ 20
- ○ 21
- ○ 22
- ○ 23
- ○ 24
- ○ 25
- ○ 26
- ○ 27
- ○ 28
- ○ 29

Go back Logout

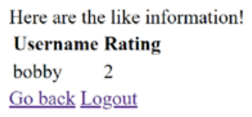
After choosing one of the photo ID, user will see who are tagged in this photo and the tagstatus:


Here are the tag information!

Username	FirstName	LastName	Tagstatus
bobby	Bobby	Brown	1
colieen	Colieen	Douglas	0

Go back Logout

By clicking the second link and choose one of the photo ID, use will see who have liked this photo and the rating information:


Here are the like information!

Username	Rating
bobby	2

Go back Logout

Feature3:

Post a photo

Team member in charged:

Bohan, Kevin, Trang

Queries:

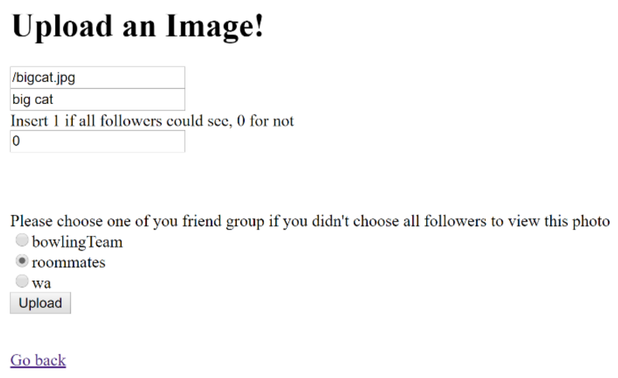
```
'INSERT INTO photo(postingdate, photoPoster, filePath, caption, allFollowers)
VALUES(%s, %s, %s, %s, %s)'

'INSERT INTO SharedWith(groupOwner, groupName, photoID) VALUES(%s, %s, %s)'
```

Related code and files:

Line 81 – 129 in app.py

upload.html

Feature explanation:



Upload an Image!

/bigcat.jpg

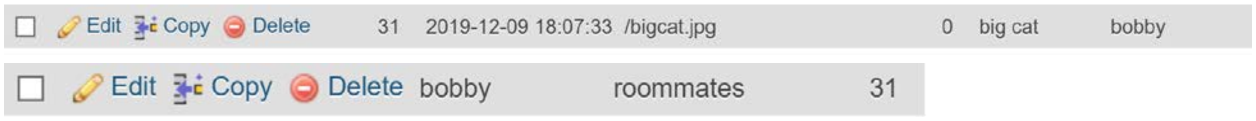
big cat

Insert 1 if all followers could see, 0 for not

0

Please choose one of you friend group if you didn't choose all followers to view this photo

- bowlingTeam
- [✓ roommates]
- wa



Edit Copy Delete	31	2019-12-09 18:07:33	/bigcat.jpg	0	big cat	bobby

Edit Copy Delete	bobby	roommates	31

User could upload the photo by providing the filePath, caption. User could also choose whether allow all their user to view this photo. If not, they can choose one of the friend group to make sure only people belong to that group can view this photo.

Feature4:

Manage Follows

Team member in charged:

Bohan

Queries:

```
"INSERT INTO Follow (username_followed, username_follower, followstatus) VALUES
(%s, %s, 0)"
SELECT DISTINCT username_follower FROM Follow WHERE username_followed = %s AND
followstatus = 0
UPDATE Follow SET followstatus = 1 WHERE username_follower = %s AND username_followed
= %s
DELETE FROM Follow WHERE username_follower = %s AND username_followed = %s
```

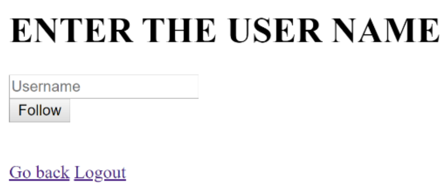
Related code and files:

Line 219-285 in app.py

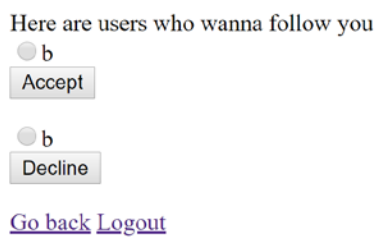
followRequest.html

followUser.html

Feature explanation:



User can follow another user by typing in their usernames. If that username doesn't exist that will be the error message shows – "Invalid username"


Here are users who wanna follow you

User also can accept the follow request or decline the request. After accept the request, the follow statues will change to 1. After decline the request, database will delete the relative row in the follow table.

Feature5:

Manage tags

Team Member in charged:

Trang Duong

Queries:

- Same query used in feature 1 to find the visible photos to certain user.

-
  ```
  SELECT username
  FROM person
  WHERE username NOT IN (SELECT username
                         FROM tagged
                         WHERE photoID = %s)
  ```
  (This query will help us to find who can be tagged in certain photo and make sure a person can be tagged only once per photo)
- INSERT INTO Tagged(username, photoID, tagstatus) VALUES (%s, %s , %s)

Related Code and Files:

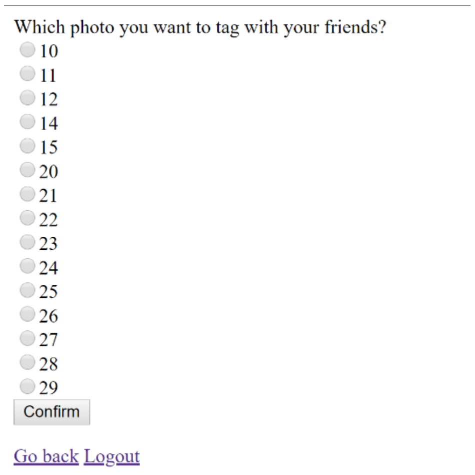
Line 288 - 402 in app.py

tagOnMyImg.html

showTagList.html

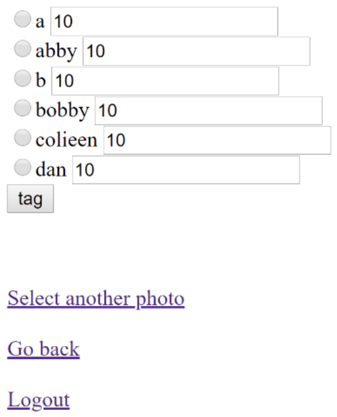
tagRequest.html

Feature explanation:


Which photo you want to tag with your friends?
10
11
12
14
15
20
21
22
23
24
25
26
27
28
29
Confirm

Go back Logout

Users could choose one of visible photos which they want to tag a person.

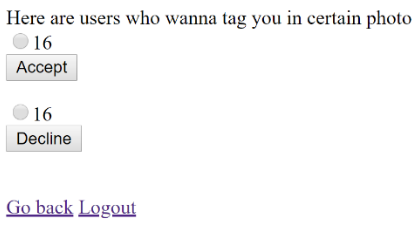

a 10
abby 10
b 10
bobby 10
colieen 10
dan 10
tag

Select another photo

Go back

Logout

After choosing the photoID, user could choose the person he wants to tag.


Here are users who wanna tag you in certain photo
16
Accept

16
Decline

Go back Logout

If users tag themselves to particular photo, that will automatically accept the tag request.

If not, user will receive a tag request and accept or decline it.

Feature7:

Add Comments

Team Member in Charged:

Kevin Chang

Queries:

- Query in feature 1 return all the visible photo
- INSERT INTO comment(commentStr, ts, photoID, username) VALUES (%s, %s, %s, %s)
- SELECT commentStr
  FROM comment
  WHERE photoID = %s

Related Code and Files:

Line 541-558 app.py

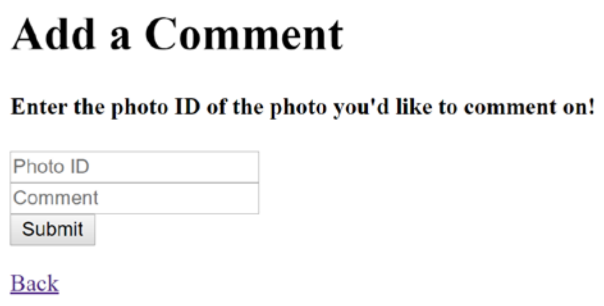
Line 586-631 app.py

addComment.html

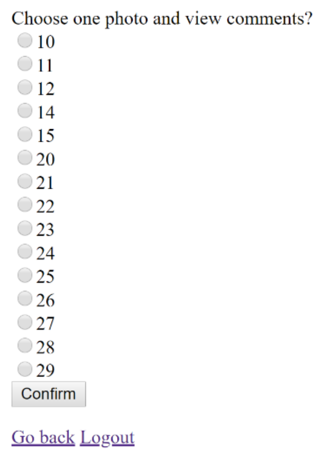
showCommentList.html

readComment.html

Feature Explanation:

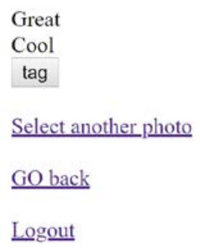


User can type the photo id and comment to certain visible photo.


Choose one photo and view comments?

○ 10
○ 11
○ 12
○ 14
○ 15
○ 20
○ 21
○ 22
○ 23
○ 24
○ 25
○ 26
○ 27
○ 28
○ 29

User could also choose one of a visible photo and view all comments of it.


Great
Cool

Feature8:

Like Photo

Team Member in Charged:

Bohan Wu

Queries:

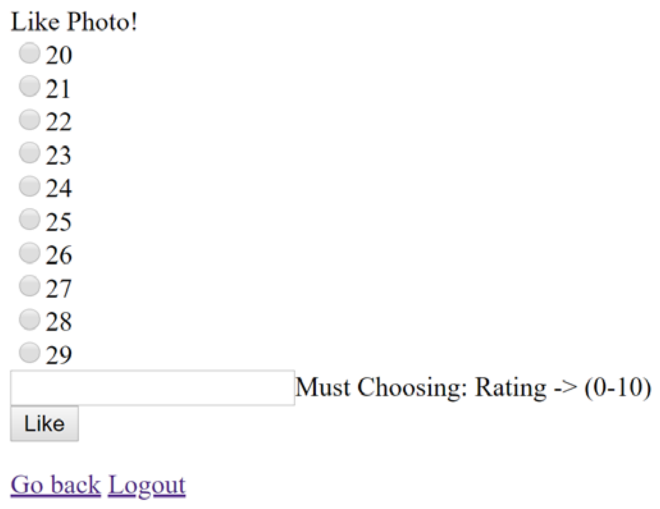
- Query in feature 1 to find all visible photoID, need to modified to make sure an user cannot like a photo more than once. Query shown at Line 410-425 app.py
- INSERT INTO Likes(username, photoID, liketime, rating) VALUES (%s, %s , %s, %s);

Related Code and Files:

Line 405-449 in app.py

likePhoto.html

Feature Explanation:



Like Photo!

- ○ 20
- ○ 21
- ○ 22
- ○ 23
- ○ 24
- ○ 25
- ○ 26
- ○ 27
- ○ 28
- ○ 29

Must Choosing: Rating -> (0-10)

User can choose one of the visible photos and provide a rating score of it. User cannot like one certain photo more than once.

Feature12:

Add friendGroup

Member in charged:

Kevin Chang

Queries:

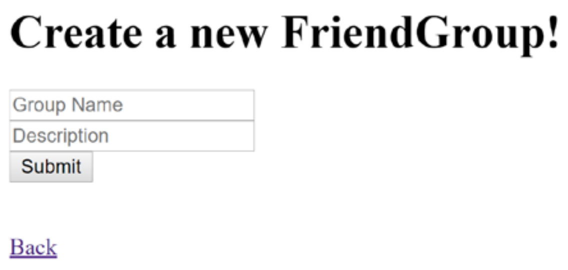
- INSERT INTO friendgroup(groupOwner, groupName, description) VALUES (%s, %s, %s)"

Related Code and Files:

Line 561-584 in app.py

friendGroup.html

Feature Explanation:



User could also generate friend groups by providing the group name and description. If the group already exist, it won't be added to the database.

Feature13:

Add Friends

Member in charged:

Bohan Wu

Queries:

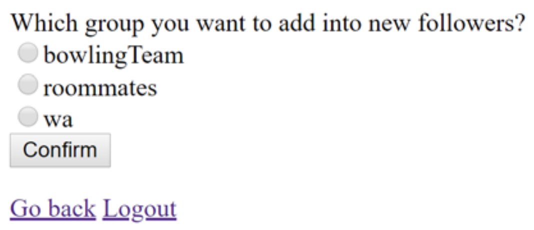
-
  ```
  SELECT username_follower
  FROM Follow
   WHERE username_followed = %s AND followstatus = 1 AND username_follower NOT IN (
                      SELECT member_username
                     FROM BelongTo     WHERE groupName = %s AND owner_username = %s)
  ```

-
  ```
  INSERT INTO BelongTo(member_username, owner_username, groupName) VALUES (%s, %s , %s);
  ```

-
  ```
  SELECT groupName
  FROM friendgroup
  WHERE groupOwner = %s
  ```

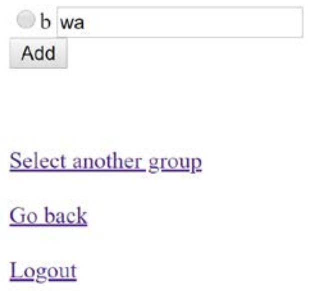
Related code and files:
Line 636-700 in app.py
manageGroup.html
showAddList.html

Feature explanation:


Which group you want to add into new followers?

- bowlingTeam
- roommates
- wa

Users start to choose which group they want to add new follower to.


b wa

Users could add new members into their friend groups by choosing one of their friend group and one of their followers. (Follow request must be accepted)