

Tasnim Nehal (tn1078) Project Summary

Link: <https://github.com/TasNehal/Finstagram>

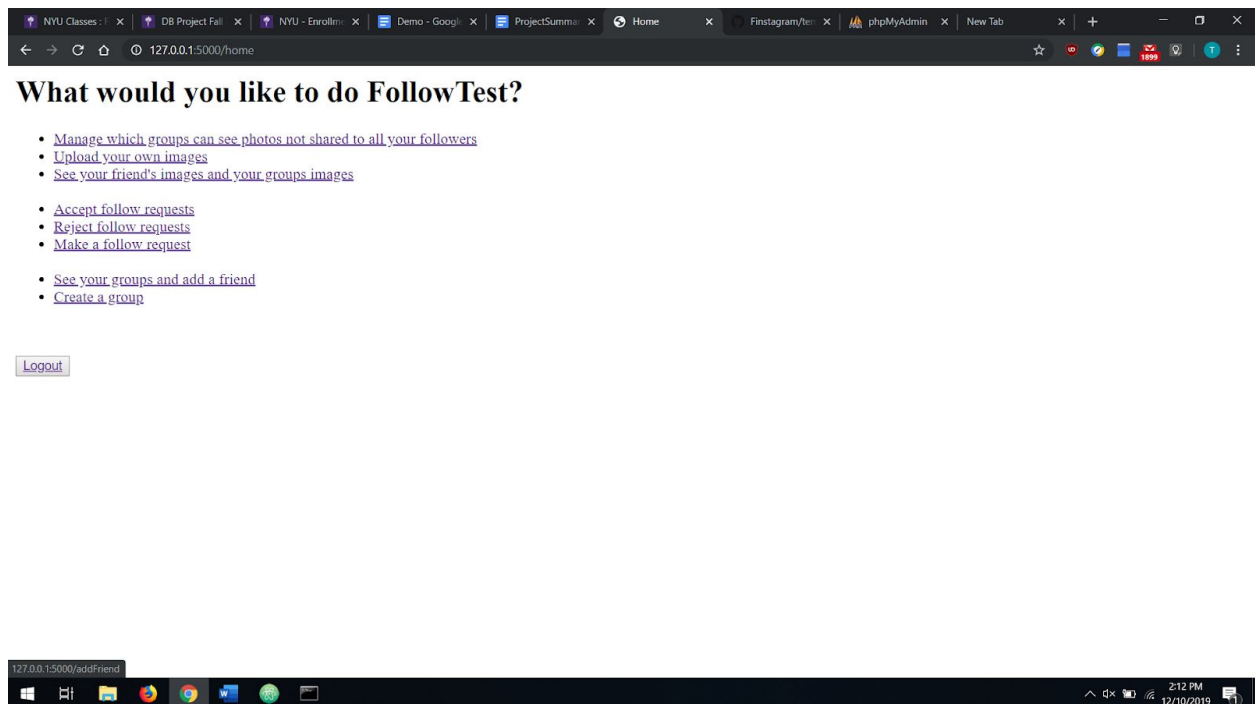
Date of last commit to repository: 12/10/19

Number of Team members: Solo project

Names and netIDs of Team members (one per line): tn1078

(skip to Feature 4-6 for the extra features, Features 1-3 focus on the core features)

Default home page:



Feature 1 + Feature 2:

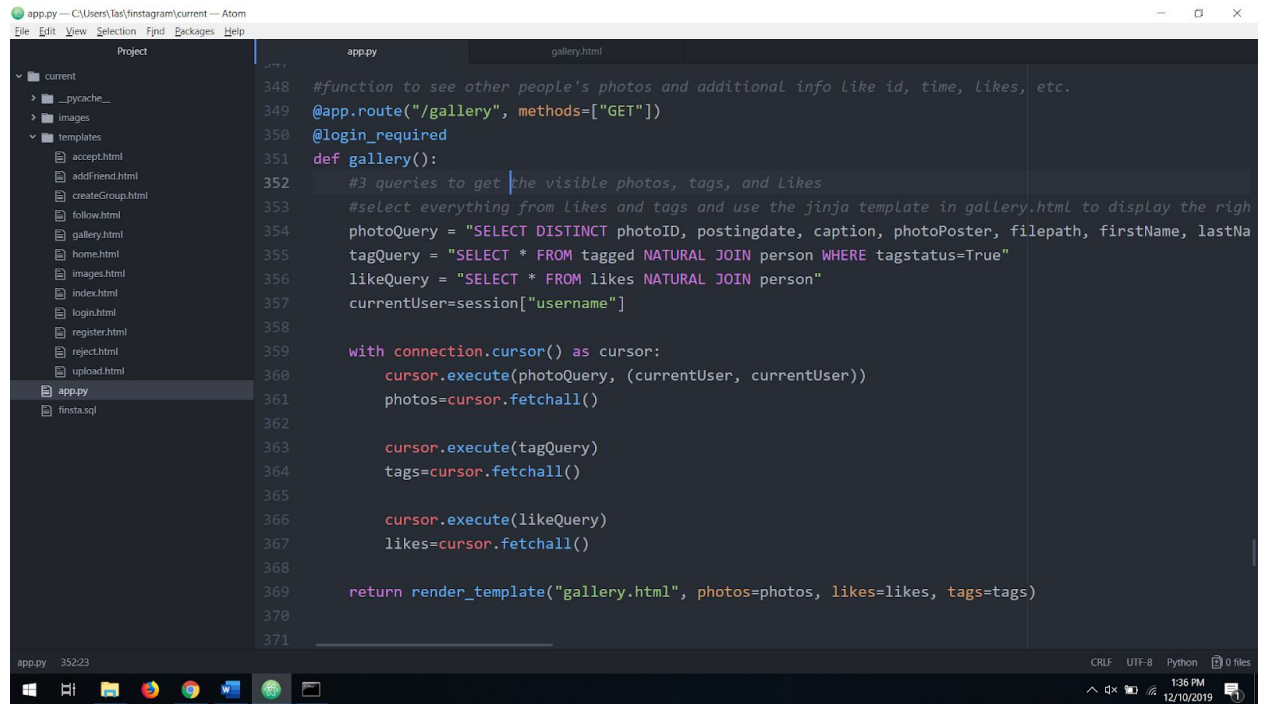
1. View visible photos + View further photo info (implemented this in one page)
2. Had three queries for this, one to pick out all the relevant photo information for the current user (%s), a query to get all the tags and the relevant data gets picked out through if conditions in the jinja template of templates/gallery.html and a third query to get all the likes where the likes for an associated photo get picked apart by if conditions in the jinja template of the same html file.

```
photoQuery = "SELECT DISTINCT photoID, postingdate, caption, photoPoster, filepath, firstName,  
  
lastName FROM photo INNER JOIN person ON person.username = photo.photoPoster  
WHERE photoPoster IN (SELECT username_followed FROM follow WHERE followstatus  
= 1 AND username_follower=%s) OR photoID IN ( SELECT photoID FROM sharedwith  
NATURAL JOIN belongto WHERE member_username=%s) ORDER BY postingdate  
DESC"
```

```
tagQuery = "SELECT * FROM tagged NATURAL JOIN person WHERE tagstatus=True"
```

```
likeQuery = "SELECT * FROM likes NATURAL JOIN person"
```

3. The html file for this page is found under templates/gallery.html and the relevant code is found under lines 348-369 of app.py:

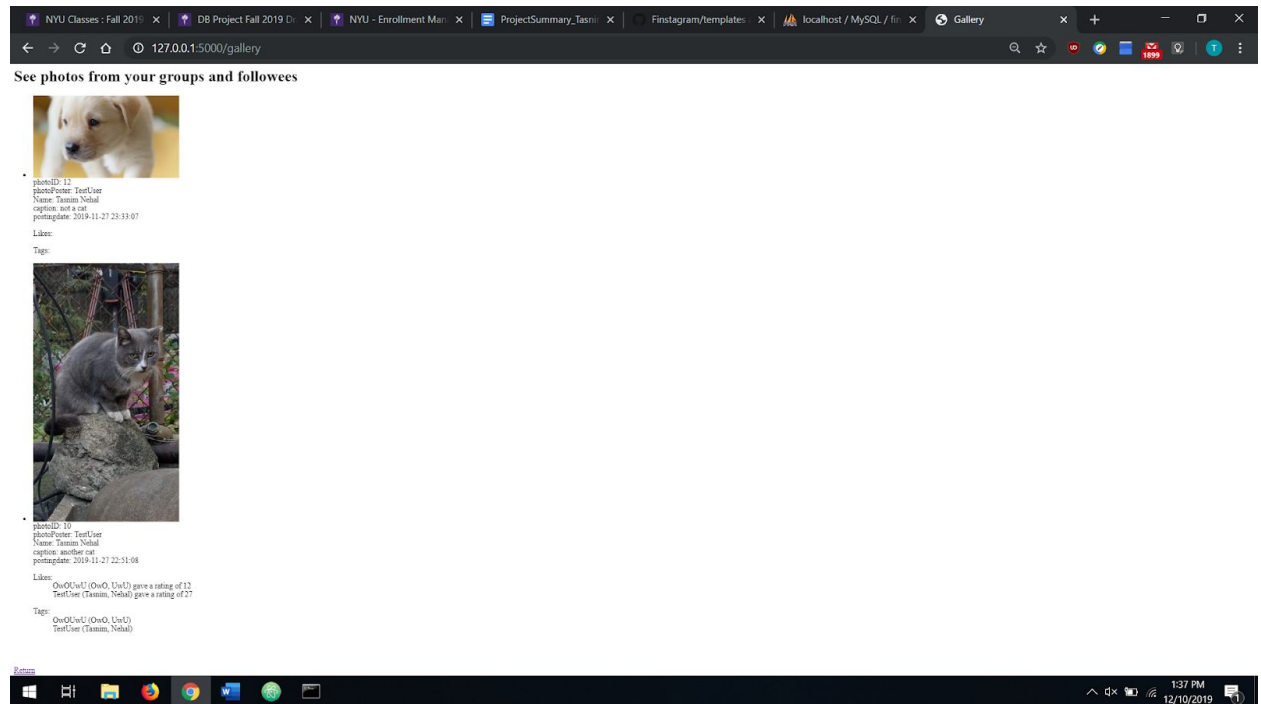


```
app.py — C:\Users\las\instagram\current — Atom
File Edit View Selection Fird Packages Help

Project
current
  _pycache_
  images
  templates
    accept.html
    addfriend.html
    createGroup.html
    follow.html
    gallery.html
    home.html
    images.html
    index.html
    login.html
    register.html
    reject.html
    upload.html
  app.py
  finsta.sql

app.py
348 #function to see other people's photos and additional info like id, time, likes, etc.
349 @app.route("/gallery", methods=["GET"])
350 @login_required
351 def gallery():
352     #3 queries to get the visible photos, tags, and Likes
353     #select everything from Likes and tags and use the jinja template in gallery.html to display the right
354     photoQuery = "SELECT DISTINCT photoID, postingdate, caption, photoPoster, filepath, firstName, lastNa
355     tagQuery = "SELECT * FROM tagged NATURAL JOIN person WHERE tagstatus=True"
356     likeQuery = "SELECT * FROM likes NATURAL JOIN person"
357     currentUser=session["username"]
358
359     with connection.cursor() as cursor:
360         cursor.execute(photoQuery, (currentUser, currentUser))
361         photos=cursor.fetchall()
362
363         cursor.execute(tagQuery)
364         tags=cursor.fetchall()
365
366         cursor.execute(likeQuery)
367         likes=cursor.fetchall()
368
369     return render_template("gallery.html", photos=photos, likes=likes, tags=tags)
370
371
```

4. This is the gallery screen showing photos for user FollowTest with a photo with no likes and tags and a photo with 2 likes and 2 tags. This can be found by going through the link “See your friend’s images and your groups images” on the home page after logging in.



This is the relevant tables (Note my photoId's 1-8 are missing in the screenshots because I deleted those rows while testing beforehand. Deleting the database and starting over with the exact same data will fix this if that is an issue)

localhost/phpmyadmin/sql.php?server=1&db=finstagram&table=photo&pos=0

SELECT * FROM `photo`

Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

| | photoID | postingdate | filepath | allFollowers | caption | photoPoster |
|---|---------|---------------------|--------------|--------------|-------------|-------------|
| <input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete | 9 | 2019-11-27 21:15:05 | whitecat.jpg | 1 | cat | OwOUwU |
| <input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete | 10 | 2019-11-27 22:51:08 | graycat.jpg | 0 | another cat | TestUser |
| <input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete | 11 | 2019-11-27 23:27:20 | kitten.png | 1 | Small cat | OwOUwU |
| <input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete | 12 | 2019-11-27 23:33:07 | dog.jpg | 0 | not a cat | TestUser |

☐ Check all With selected: ☐ Edit ☐ Copy ☐ Delete ☐ Export

Number of rows: 25 Filter rows: Search this table Sort by key: None

Query results operations

☐ Print ☐ Copy to clipboard ☐ Export ☐ Display chart ☐ Create view

localhost/phpmyadmin/sql.php?server=1&db=finstagram&table=likes&pos=0

Server: MySQL:3306 » Database: finstagram » Table: likes

Showing rows 0 - 1 (2 total, Query took 0.0005 seconds.)

SELECT * FROM `likes`

Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

| | username | photoID | liketime | rating |
|---|----------|---------|---------------------|--------|
| <input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete | OwOUwU | 10 | 2019-12-09 00:00:00 | 12 |
| <input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete | TestUser | 10 | 2019-12-09 00:00:00 | 27 |

☐ Check all With selected: ☐ Edit ☐ Copy ☐ Delete ☐ Export

Number of rows: 25 Filter rows: Search this table Sort by key: None

Query results operations

☐ Print ☐ Copy to clipboard ☐ Export ☐ Display chart ☐ Create view

phpMyAdmin

Current server: MySQL

Recent Favorites

- New
- finstagram
 - New
 - belongto
 - follow
 - friendgroup
 - likes
 - person
 - photo
 - sharedwith
 - tagged
- information_schema
- moviesaw
- mysql
- part2
- performance_schema
- sys

Server: MySQL:3306 » Database: finstagram » Table: tagged

Browse Structure SQL Search Insert Export Import Privileges

Showing rows 0 - 1 (2 total, Query took 0.0005 seconds.)

```
SELECT * FROM `tagged`
```

Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

+ Options

| | username | photoID | tagstatus |
|---|----------|---------|-----------|
| <input type="checkbox"/> Edit Copy Delete | OwOUwU | 10 | 1 |
| <input type="checkbox"/> Edit Copy Delete | TestUser | 10 | 1 |

Check all | With selected: Edit Copy Delete Export

Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

Query results operations

Print Copy to clipboard Export Display chart Create view

1:41 PM 12/10/2019

Feature 3

1. Post a photo: Allows user to upload an image in “/upload” and choose whether or not to send to allFollowers. If allFollowers is chosen, the SharedWith table is updated with all the groups the user is in. If not, the user can choose what photo to share to what group in “/images”
2. Had a photo query to insert data into the photo table after the current user inserts data, a search query to find all the groups a user belongs to if that user chose to share with all followers, another query to find the max photoID (IE the last photo that was submitted which was submitted by this user), and one final query to insert into the sharedwith table for each group the user belongs to if all followers is chosen (this last query is repeated in a for loop for every such group)

```
photoQuery = "INSERT INTO photo (postingdate, filePath, allFollowers, caption, photoPoster)  
VALUES (%s, %s, %s, %s, %s)"
```

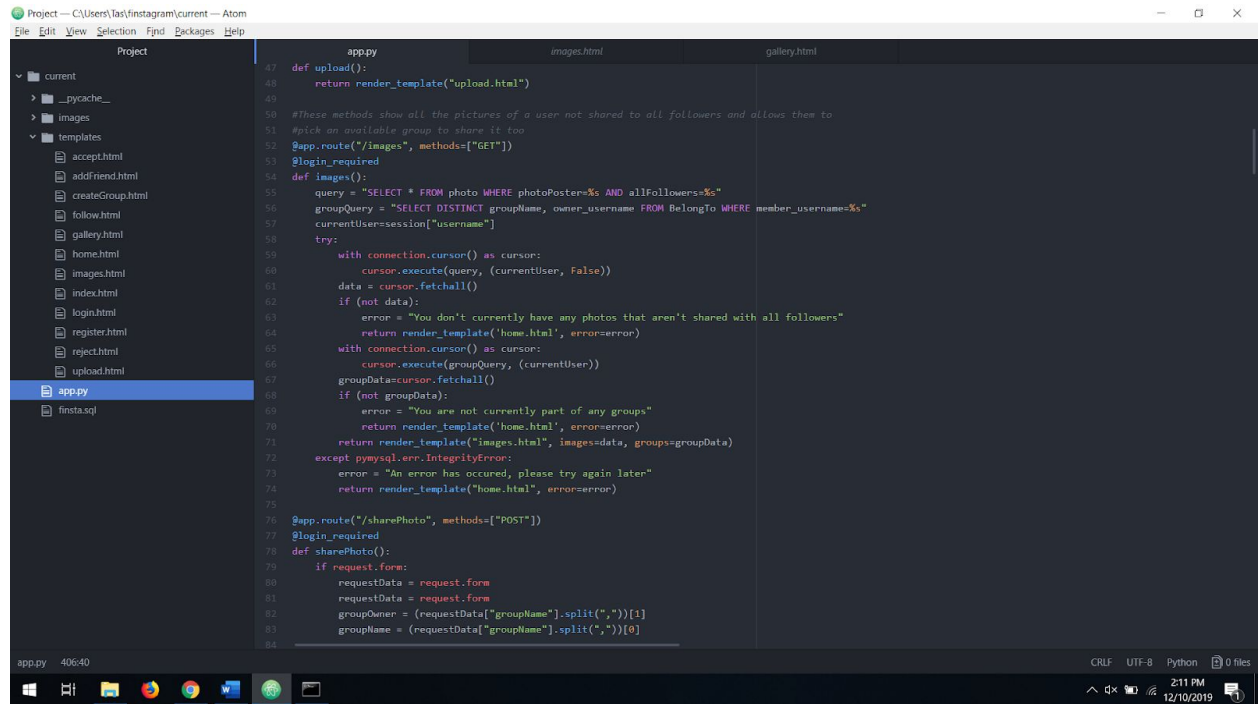
```
searchQuery = "SELECT DISTINCT groupName, owner_username FROM BelongTo WHERE  
member_username=%s"
```

```
idQuery = "SELECT MAX(photoID) as id FROM photo"
```

```
sharedQuery = "INSERT INTO SharedWith (groupOwner, groupName, photoID) VALUES (%s,  
%s, %s)"
```

3. The user can upload images and choose allfollowers or not under templates/upload.html and if a user has photos that are not shared with allfollowers, that user can highlight the groups that the photo should be shared with under templates/images.html but this page can only be accessed if the user is part of a group and has existing photos where allFollowers=False. The code for uploading is found under lines 380-418 of app.py and the code for sharing photos with groups is found under lines 50-98

and I used two functions for rendering the forms and another for processing the forms and inserting values into the table..



```
Project — C:\Users\lor\instagram\current — Atom
File Edit View Selection Find Packages Help

Project
├── current
│   ├── __pycache__
│   ├── images
│   └── templates
│       ├── accept.html
│       ├── addfriend.html
│       ├── createGroup.html
│       ├── follow.html
│       ├── gallery.html
│       ├── home.html
│       ├── images.html
│       ├── index.html
│       ├── login.html
│       ├── register.html
│       ├── reject.html
│       └── upload.html
└── app.py
    └── insta.sql

app.py
47 def upload():
48     return render_template("upload.html")
49
50 #These methods show all the pictures of a user not shared to all followers and allows them to
51 #pick an available group to share it too
52 @app.route("/images", methods=["GET"])
53 @login_required
54 def images():
55     query = "SELECT * FROM photo WHERE photoPoster=%s AND allFollowers=%s"
56     groupQuery = "SELECT DISTINCT groupName, owner_username FROM BelongsTo WHERE member_username=%s"
57     currentUserSession["username"]
58     try:
59         with connection.cursor() as cursor:
60             cursor.execute(query, (currentUser, False))
61             data = cursor.fetchall()
62             if (not data):
63                 error = "You don't currently have any photos that aren't shared with all followers"
64                 return render_template("home.html", error=error)
65             with connection.cursor() as cursor:
66                 cursor.execute(groupQuery, (currentUser,))
67             groupData=cursor.fetchall()
68             if (not groupData):
69                 error = "You are not currently part of any groups"
70                 return render_template("home.html", error=error)
71             return render_template("images.html", images=data, groups=groupData)
72     except pymysql.err.IntegrityError:
73         error = "An error has occurred, please try again later"
74         return render_template("home.html", error=error)
75
76 @app.route("/sharePhoto", methods=["POST"])
77 @login_required
78 def sharePhoto():
79     if request.form:
80         requestData = request.form
81         requestData = request.form
82         groupOwner = (requestData["groupName"].split(",")[1])
83         groupName = (requestData["groupName"].split(",")[0])
84
```



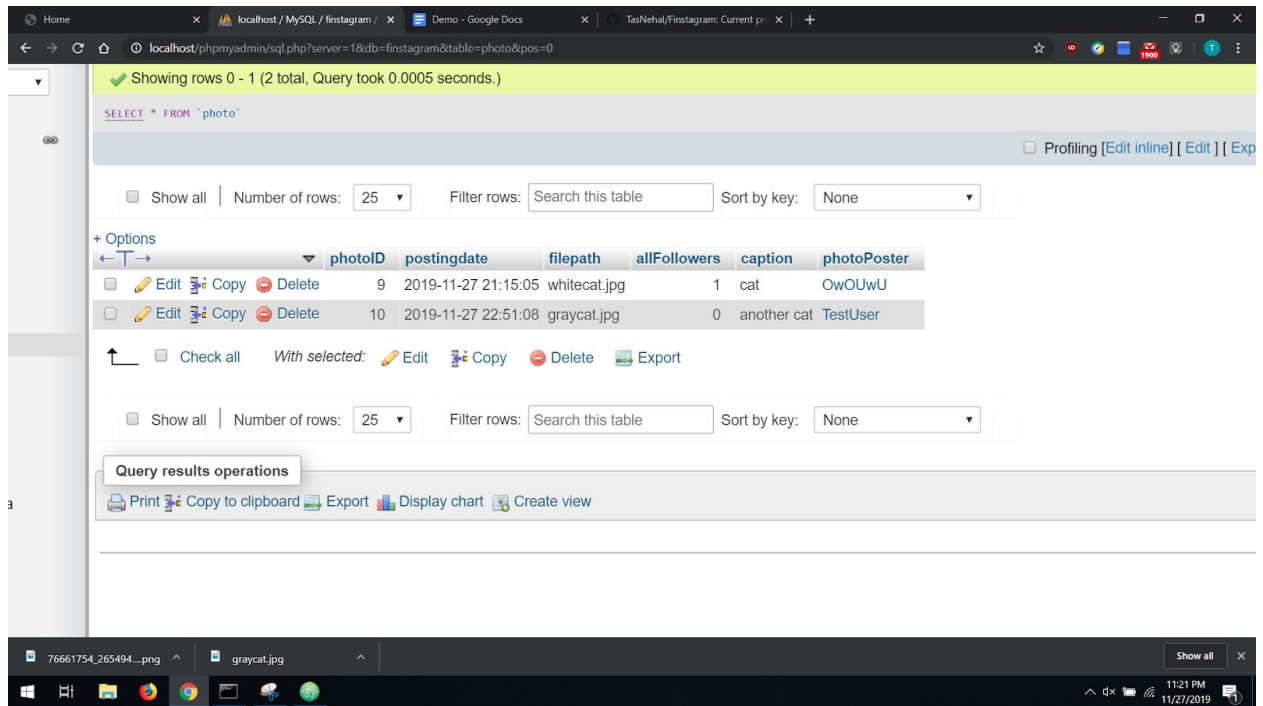
```
Project — C:\Users\lari\instagram\current — Atom
File Edit View Selection Find Packages Help

Project
  current
    __pycache__
    images
    templates
      accept.html
      addfriend.html
      createGroup.html
      follow.html
      gallery.html
      home.html
      images.html
      index.html
      login.html
      register.html
      reject.html
      upload.html
  app.py
  insta.sql

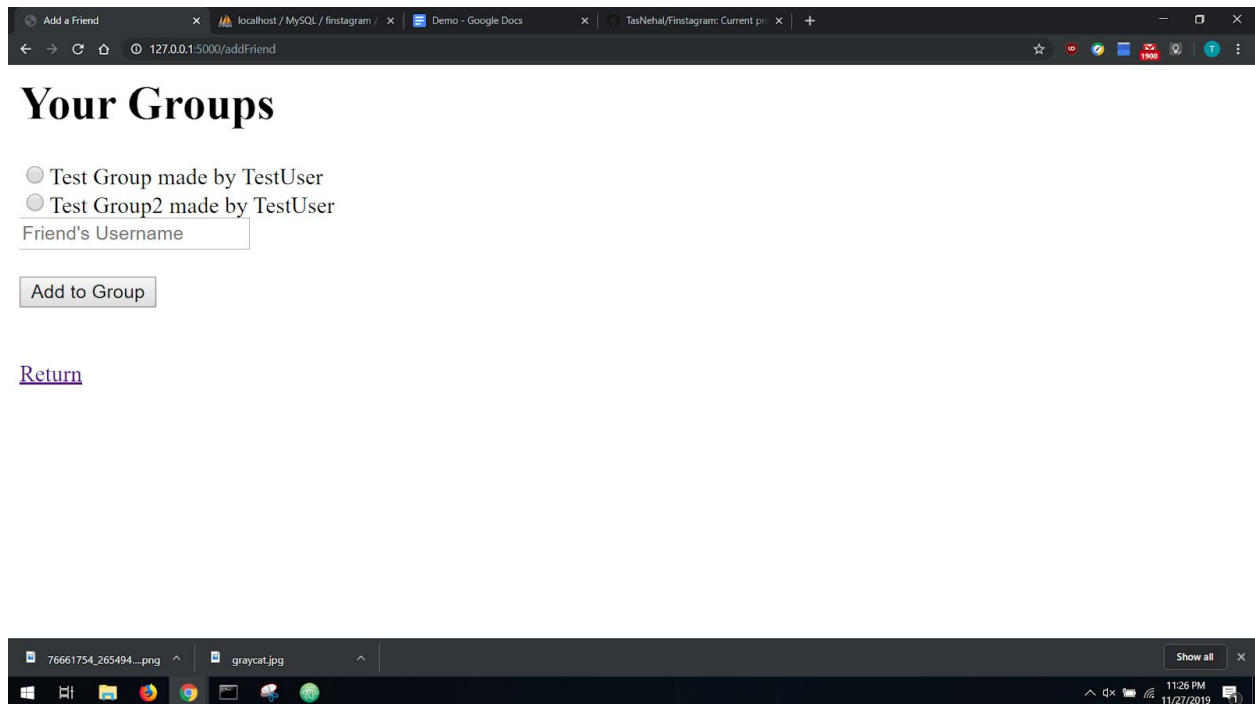
app.py
383 @app.route("/uploadImage", methods=["POST"])
384 @login_required
385 def upload_image():
386     if request.files:
387         requestData = request.form
388         image_file = request.files.get("imageToUpload", "")
389         image_name = image_file.filename
390         filepath = os.path.join(IMAGES_DIR, image_name)
391         image_file.save(filepath)
392         caption=requestData["caption"]
393         photoPoster=session["username"]
394         if (requestData["status"] == "True"):
395             allfollowers=True
396         else:
397             allfollowers=False
398         photoQuery = "INSERT INTO photo (postingdate, filePath, allfollowers, caption, photoPoster) VALUES (%s, %s, %s, %s, %s)"
399         with connection.cursor() as cursor:
400             cursor.execute(photoQuery, (time.strftime('%Y-%m-%d %H:%M:%S'), image_name, allfollowers, caption, photoPoster))
401             if (allfollowers):
402                 searchQuery = "SELECT DISTINCT groupName, owner_username FROM BelongTo WHERE member_username=%s"
403                 cursor.execute(searchQuery, (photoPoster))
404                 groups=cursor.fetchall()
405                 idQuery = "SELECT MAX(photoID) as id FROM photo"
406                 cursor.execute(idQuery)
407                 photoID = cursor.fetchone()[0]
408                 sharedQuery = "INSERT INTO SharedWith (groupOwner, groupName, photoID) VALUES (%s, %s, %s)"
409                 for group in groups:
410                     groupName=group["groupName"]
411                     groupOwner=group["owner_username"]
412                     cursor.execute(sharedQuery, (groupOwner, groupName, photoID))
413
414             message = "Image has been successfully uploaded."
415             return render_template("upload.html", message=message)
416         else:
417             message = "Failed to upload image."
418             return render_template("upload.html", message=message)
419
```

4. /upload is found by following the link called “Upload your own images” and /images is found under “Manage which groups can see photos not shared to all your followers”. Here is the demo:

Photos at the start of demo:



Current user OwOUwU is part of groups Test Group and Test Group2



User uploads photo of a kitten

Image Upload

localhost / MySQL / firstagram / Demo - Google Docs TasNehal/Firstagram: Current pr

127.0.0.1:5000/upload

Upload an Image!

Choose File kitten.png

Upload to all your followers?
☒ Yes ☐ No

Small cat

Upload

[Return](#)



Photo table is updated and shared with table for two groups is updated accordingly

Showing rows 0 - 2 (3 total, Query took 0.0004 seconds.)

SELECT * FROM `photo`

Profiling [Edit inline]

Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

+ Options

| | photoID | postingdate | filepath | allFollowers | caption | photoPoster |
|---|---------|---------------------|--------------|--------------|-------------|-------------|
| <input type="checkbox"/> Edit Copy Delete | 9 | 2019-11-27 21:15:05 | whitecat.jpg | 1 | cat | OwOUwU |
| <input type="checkbox"/> Edit Copy Delete | 10 | 2019-11-27 22:51:08 | graycat.jpg | 0 | another cat | TestUser |
| <input type="checkbox"/> Edit Copy Delete | 11 | 2019-11-27 23:27:20 | kitten.png | 1 | Small cat | OwOUwU |

Check all | With selected: Edit Copy Delete Export

Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

Query results operations

Print Copy to clipboard Export Display chart Create view

Image Upload localhost / MySQL / instagram / Demo - Google Docs TasNehal/Instagram: Current p... dog - Google Search

localhost/phpmyadmin/sql.php?server=1&db=instagram&table=sharedwith&pos=0

☐ Show all Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

| | | | | groupOwner | groupName | photoID |
|--------------------------|------|------|--------|------------|-------------|---------|
| <input type="checkbox"/> | Edit | Copy | Delete | TestUser | Test Group | 9 |
| <input type="checkbox"/> | Edit | Copy | Delete | TestUser | Test Group2 | 9 |
| <input type="checkbox"/> | Edit | Copy | Delete | TestUser | Test Group2 | 10 |
| <input type="checkbox"/> | Edit | Copy | Delete | TestUser | Test Group | 11 |
| <input type="checkbox"/> | Edit | Copy | Delete | TestUser | Test Group2 | 11 |

↑ ☐ Check all With selected: Edit Copy Delete Export

☐ Show all Number of rows: 25 Filter rows: Search this table Sort by key: None

Query results operations

dog.jpg 76661754_265494...png graycat.jpg Show all

11:31 PM
11/27/2019

Feature 4:

1. Manage follows

2. These are the queries for following someone. The first query checks if there is already an existing follow request between two users before allowing a user to create another follow request, the second query actually inserts a follow request into the follows table once it passes the check.

query = "SELECT * FROM Follow WHERE username_followed=%s AND username_follower=%s"

query = "INSERT INTO Follow (username_followed, username_follower, followstatus) VALUES (%s, %s, %s)"

This query displays all pending follow requests for the current user(follow status is false) and displays them.

query = "SELECT * FROM Follow WHERE username_followed=%s AND followstatus=False"

After a user chooses a request to accept, this query updates the associated row:

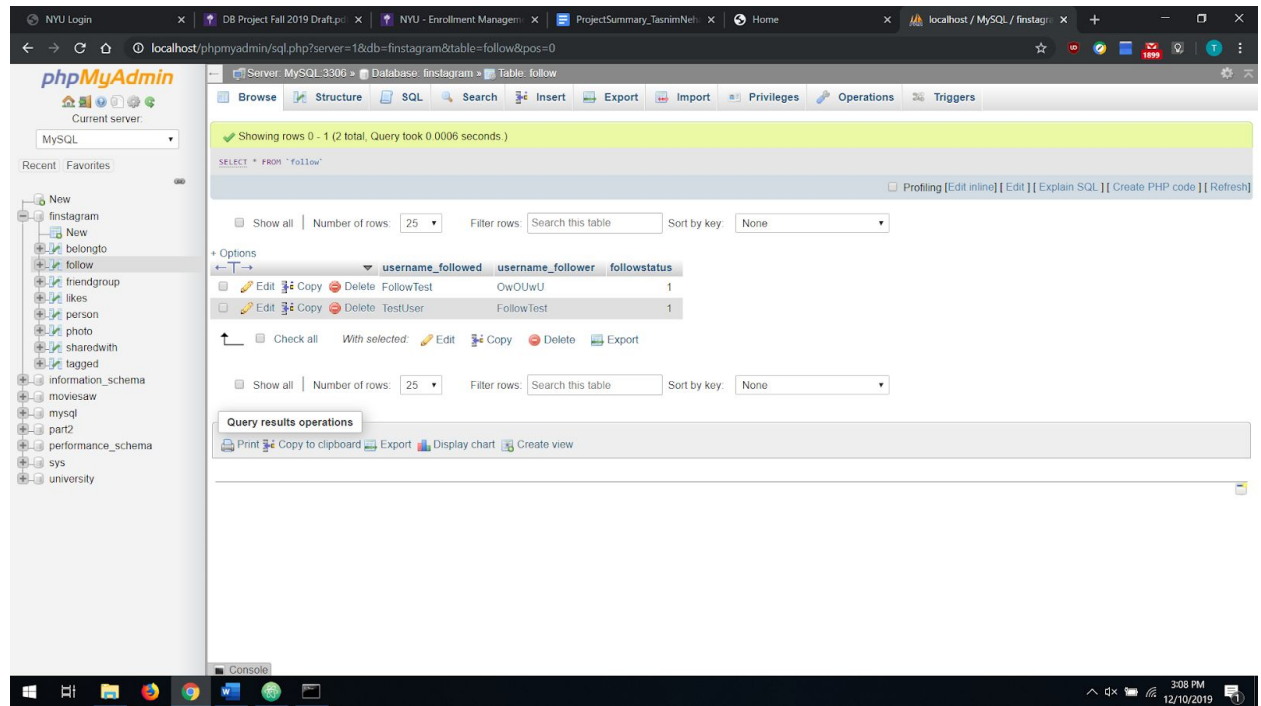
query = "UPDATE Follow SET followstatus=%s WHERE username_followed=%s AND
username_follower=%s"

If a user chooses to reject a request, the associated row is deleted with this query:

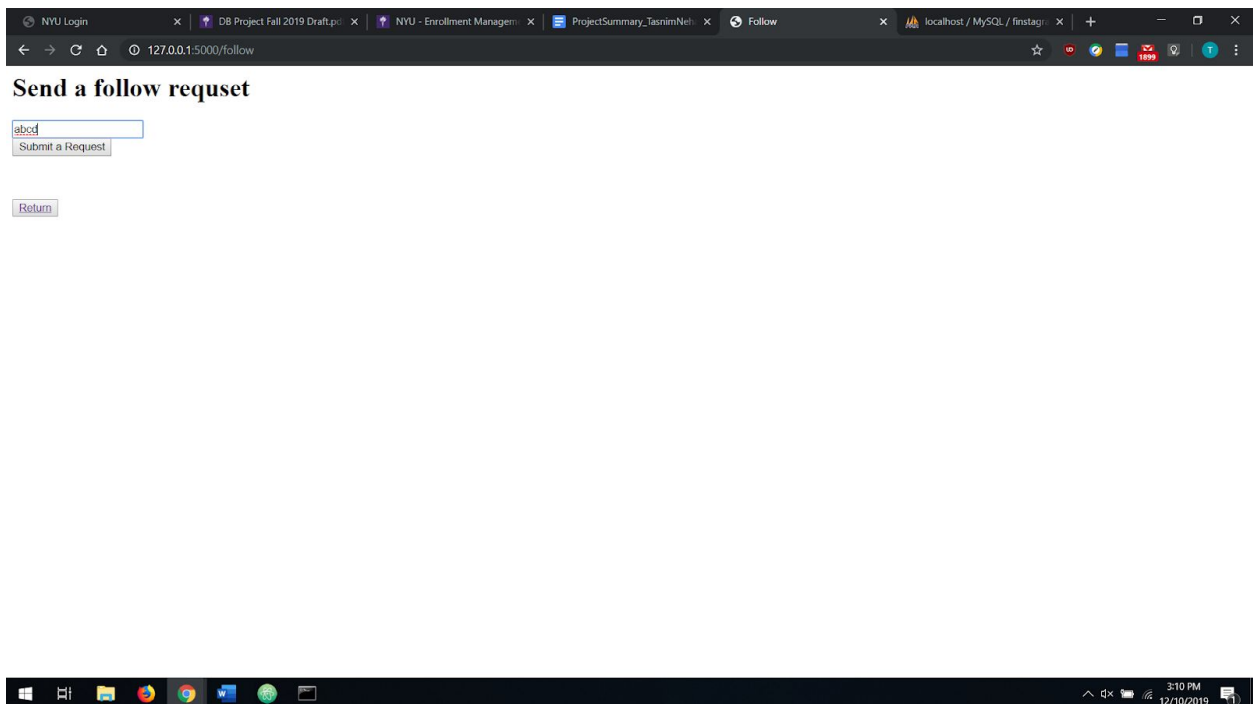
query = "DELETE FROM Follow WHERE username_follower=%s AND username_followed=%s"

3. Manage follows was incorporated over 3 different pages. The three links in home page are “Make a follow request” (template/follow), “Reject a follow request” (template/reject), “Accept a follow request” (/accept). The last two pages requires the user to have a pending follow request first before they can be accessed. Each page used two different functions each (6 total), one to generate the forms and one to process the forms and insert or remove data into the tables) The 6 functions occupy lines 240-343. Only one follow can be accepted or rejected at a time.

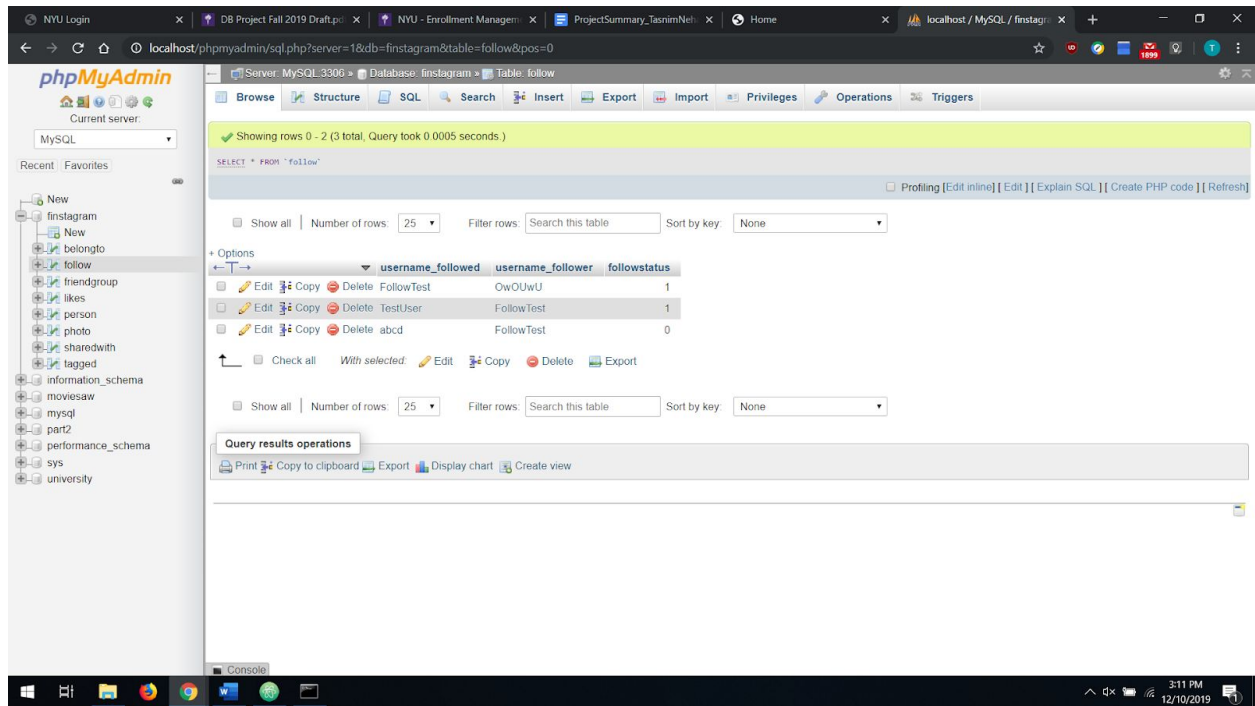
4. This is the current follow table:



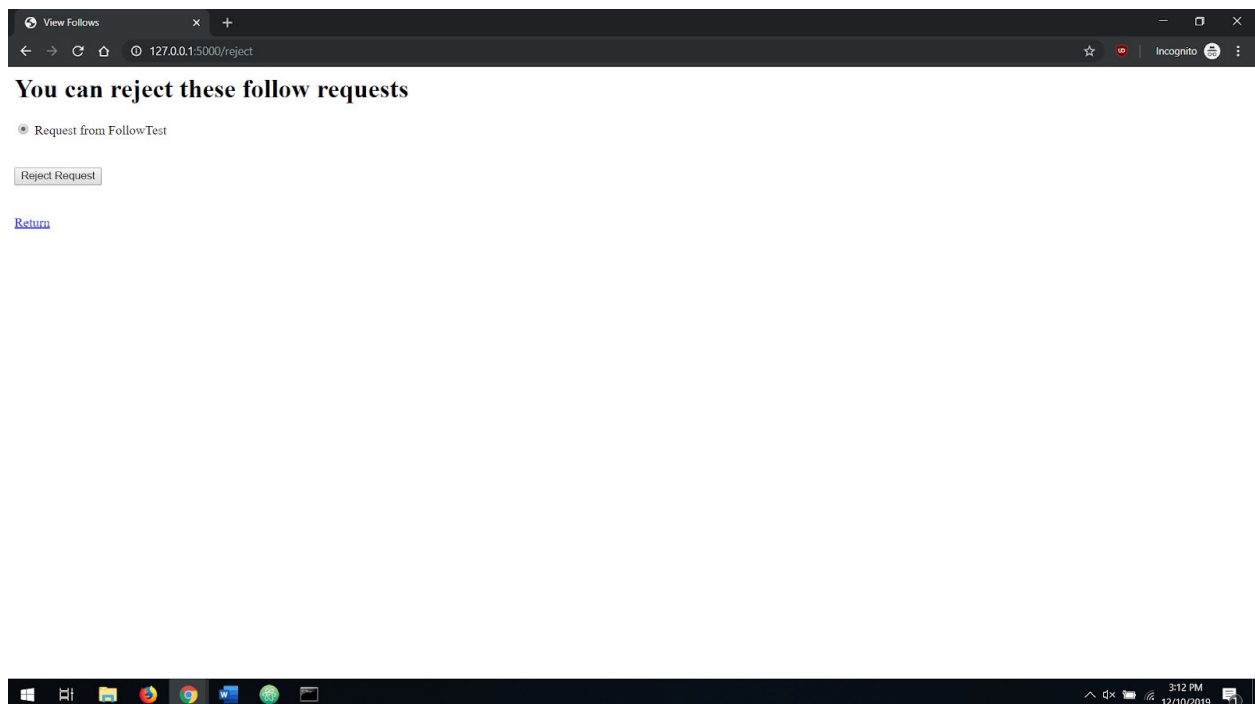
User FollowTest has to send a follow request to an existing user, in this case user abcd, or the request will not go through. Only the user on the receiving end can accept the request.

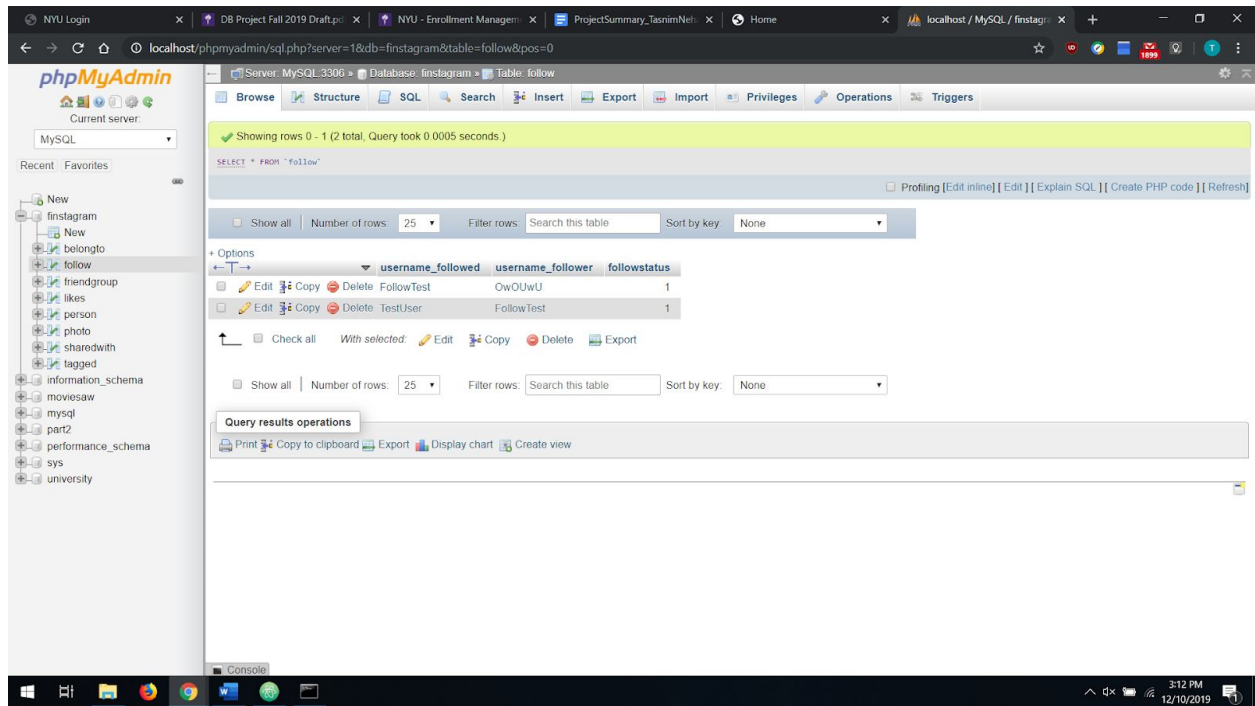


The request is sent

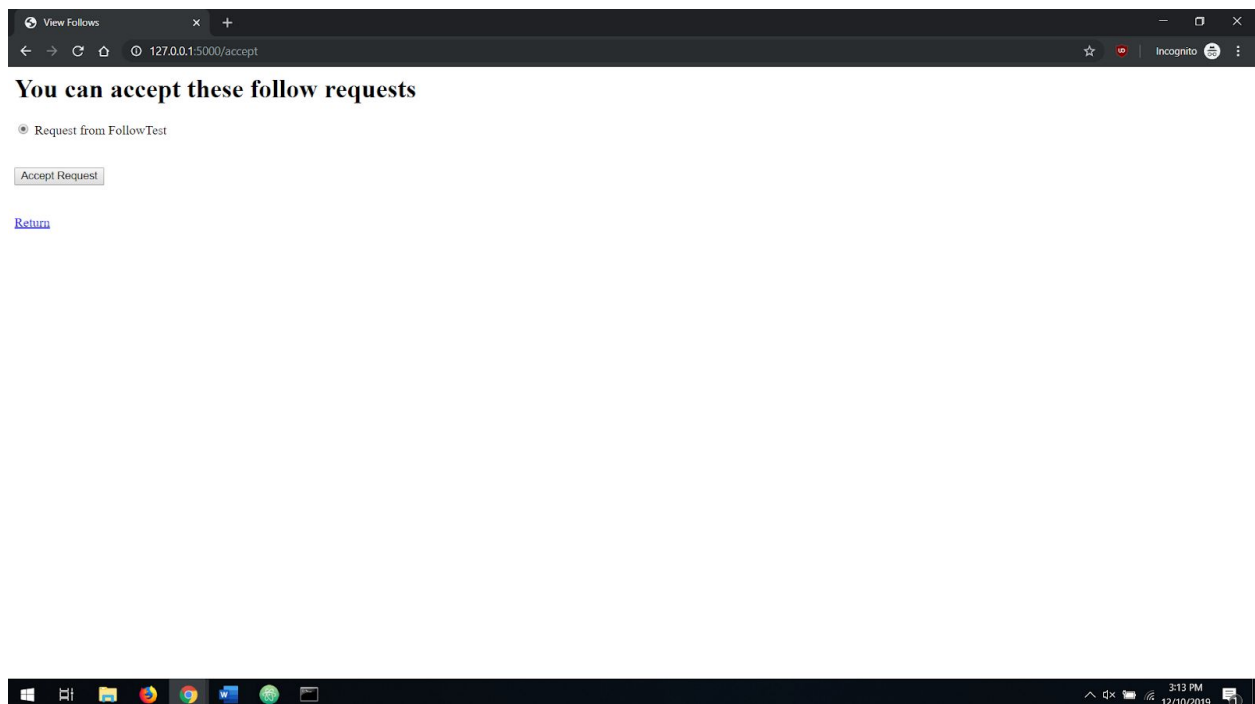


User abcd can now reject the request removing it from the table.





After resending the request, abcd accepts it this time and the table is updated.



NYU Login | DB Project Fall 2019 Draft.p... | NYU - Enrollment Manager... | ProjectSummary_TasnimNe... | localhost / MySQL / finstag... | Home

localhost/phpmyadmin/sql.php?server=1&db=finstagram&table=follow&pos=0

Server: MySQL 3306 | Database: finstagram | Table: follow

Showing rows 0 - 2 (3 total, Query took 0.0005 seconds)

SELECT * FROM `follow`

Options: Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

| | username_followed | username_follower | followstatus |
|--------------------------|-------------------|-------------------|--------------|
| <input type="checkbox"/> | FollowTest | OwOUwU | 1 |
| <input type="checkbox"/> | TestUser | FollowTest | 1 |
| <input type="checkbox"/> | abcd | FollowTest | 1 |

Query results operations: Print | Copy to clipboard | Export | Display chart | Create view

localhost/phpmyadmin/sql.php?server=1&db=finstagram&table=follow&pos=0#

3:14 PM 12/10/2019

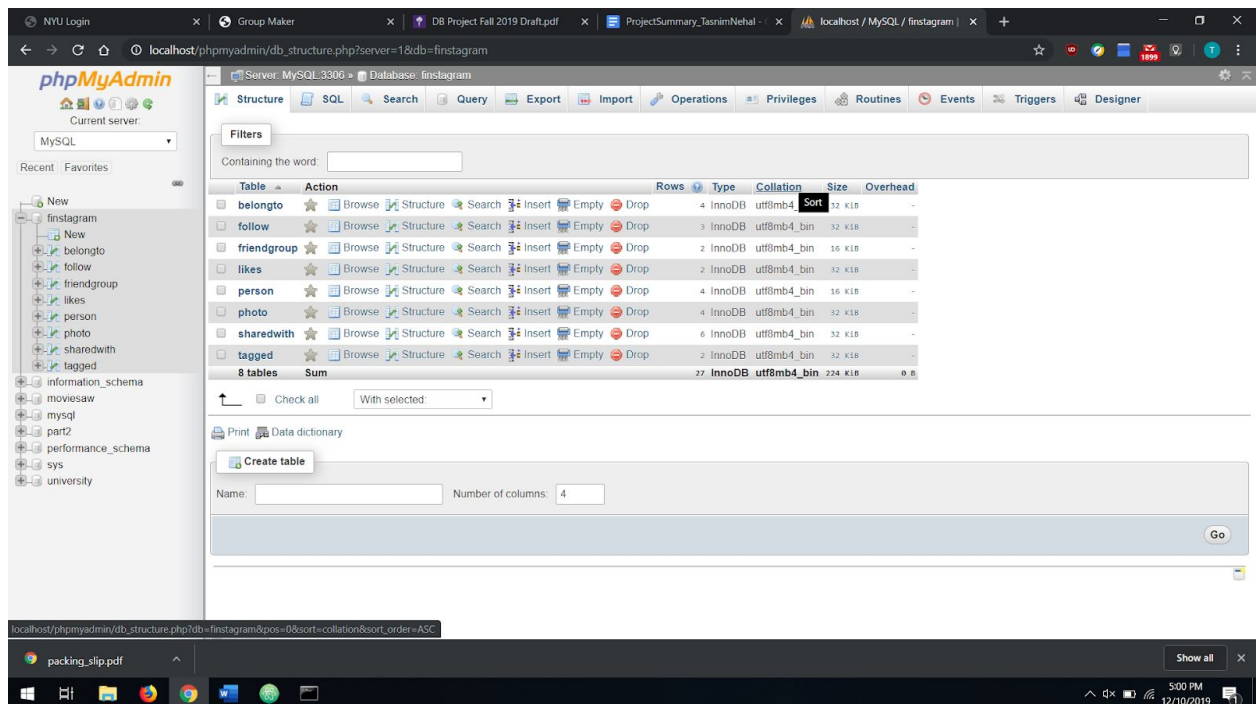
Feature 5:

1. Add friend group
2. The first query creates the group itself while the second query adds the creator of the group as a member of that group that was just made

query = "INSERT INTO Friendgroup (groupOwner, groupName, description) VALUES (%s, %s, %s)"

query2 = "INSERT INTO BelongTo (member_username, owner_username, groupName) VALUES (%s, %s, %s)"

3. This function relies on the page template/createGroup.html and can be accessed from the link "Create a group" on the home page. The two methods that generate the forms and processes the forms for inserting into the table respectively can be found on lines 163-192 of app.py
4. This is the current groups list



User called FollowTest creates a group called Demo Group and that group and its description is inserted

NYU Login x Group Maker x DB Project Fall 2019 Draft.pdf x ProjectSummary_TasnimNehal - x localhost / MySQL / finstagram | x +

127.0.0.1:5000/createGroup

Create a new group

Demo Group

testing123

Create

Return



NYU Login x Home x DB Project Fall 2019 Draft.pdf x ProjectSummary_TasnimNehal - x localhost / MySQL / finstagram | x +

localhost/phpmyadmin/sql.php?server=1&db=finstagram&table=friendgroup&pos=0

phpMyAdmin

Current server: MySQL

Recent Favorites

- New
- instagram
- New
- belongto
- follow
- friendgroup
- likes
- person
- photo
- sharedwith
- tagged
- information_schema
- movesaw
- mysql
- part2
- performance_schema
- sys
- university

Server: MySQL 3306 Database: finstagram Table: friendgroup

Browse Structure SQL Search Insert Export Import Privileges Operations Triggers

Showing rows 0 - 2 (3 total, Query took 0.0007 seconds)

SELECT * FROM "friendgroup"

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

Options

| | groupOwner | groupName | description |
|---|------------|-------------|---------------------|
| <input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete | FollowTest | Demo Group | testing123 |
| <input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete | TestUser | Test Group | make group function |
| <input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete | TestUser | Test Group2 | asdasd |

Check all | With selected: Edit Copy Delete Export

Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

Query results operations

Print Copy to clipboard Export Display chart Create view

Console

packing_slip.pdf

Show all

5:02 PM
12/10/2019

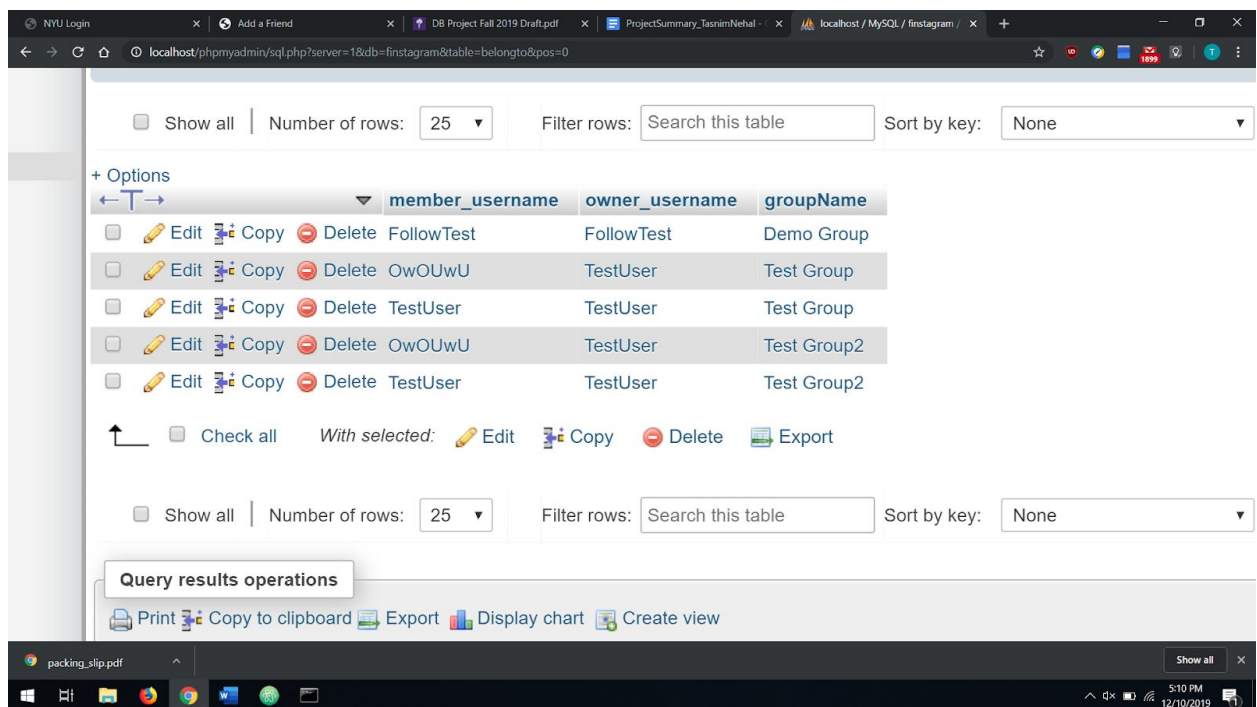
Feature 6:

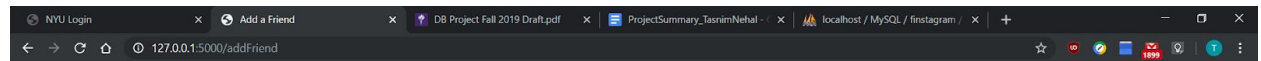
1. Add friend
2. The first query displays the current groups the user is a member of and therefore the groups a user is allowed to add a member to. The second query inserts the chosen user into the group that was selected if that user wasn't already there.

query = "SELECT DISTINCT groupName, owner_username FROM BelongTo WHERE member_username=%s"

query = "INSERT INTO BelongTo (member_username, owner_username, groupName) VALUES (%s, %s, %s)"

3. This function relies on the page template/addFriend.html and can be accessed from the link "See your groups and add a friend" on the home page. The two methods that generate the forms and processes the forms for inserting into the table respectively can be found on lines 194-236 of app.py
4. The addFriend page can only be accessed if the user is part of a member of a group first. As shown, FollowTest is the only member of Demo Group. The user can select one group at a time and type out the name of the user who is going to be added. Then the table is updated.





Your Groups

● Demo Group made by FollowTest

OwOUwU

Add to Group

[Return](#)



Showing rows 0 - 5 (6 total, Query took 0.0009 seconds)

```
SELECT * FROM `belongto`
```

Options: member_username, owner_username, groupName

| | member_username | owner_username | groupName |
|--------------------------|-----------------|----------------|-------------|
| <input type="checkbox"/> | FollowTest | FollowTest | Demo Group |
| <input type="checkbox"/> | OwOUwU | FollowTest | Demo Group |
| <input type="checkbox"/> | OwOUwU | TestUser | Test Group |
| <input type="checkbox"/> | TestUser | TestUser | Test Group |
| <input type="checkbox"/> | OwOUwU | TestUser | Test Group2 |
| <input type="checkbox"/> | TestUser | TestUser | Test Group2 |

Query results operations: Print, Copy to clipboard, Export, Display chart, Create view

