Tasnim Nehal (tn1078) Project Summary

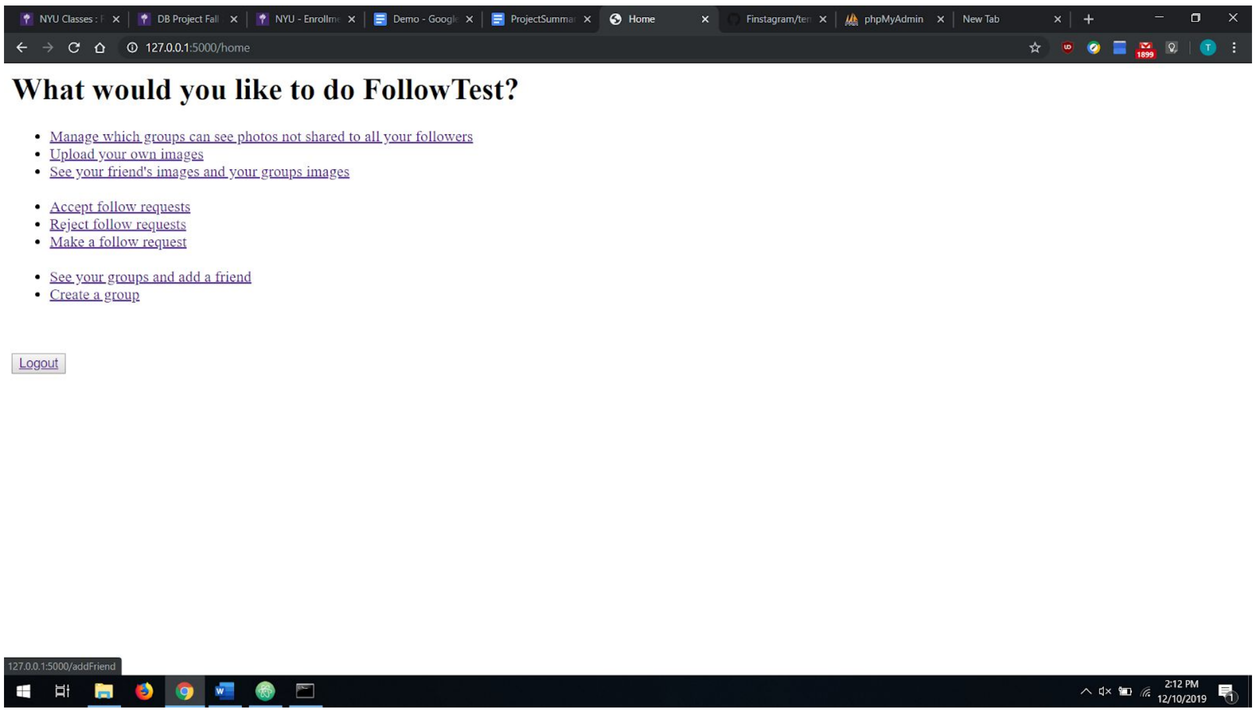
Date of last commit to repository: 12/10/19

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:



What would you like to do FollowTest?

- Manage which groups can see photos not shared to all your followers
- Upload your own images
- See your friend's images and your groups images

- Accept follow requests
- Reject follow requests
- Make a follow request

- See your groups and add a friend
- Create a group

Logout

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.

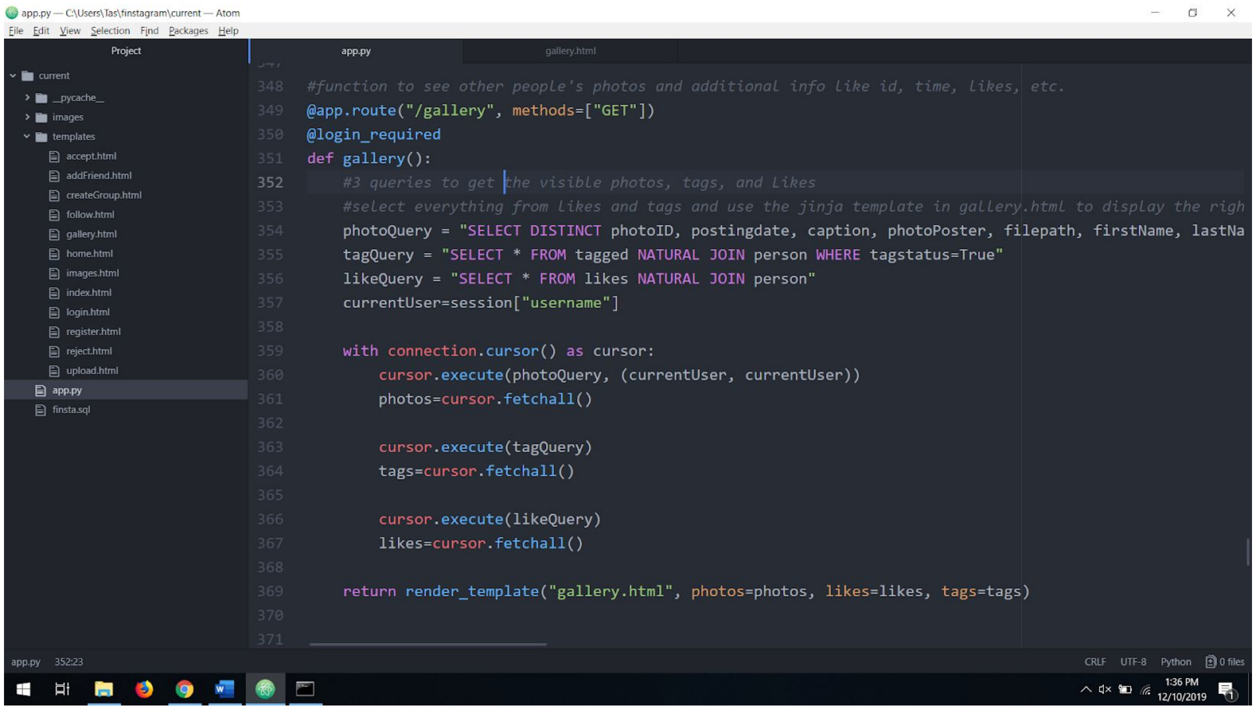
photoQuery = "SELECT DISTINCT photoID, postingdate, caption, photoPoster, filepath, firstName,

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"

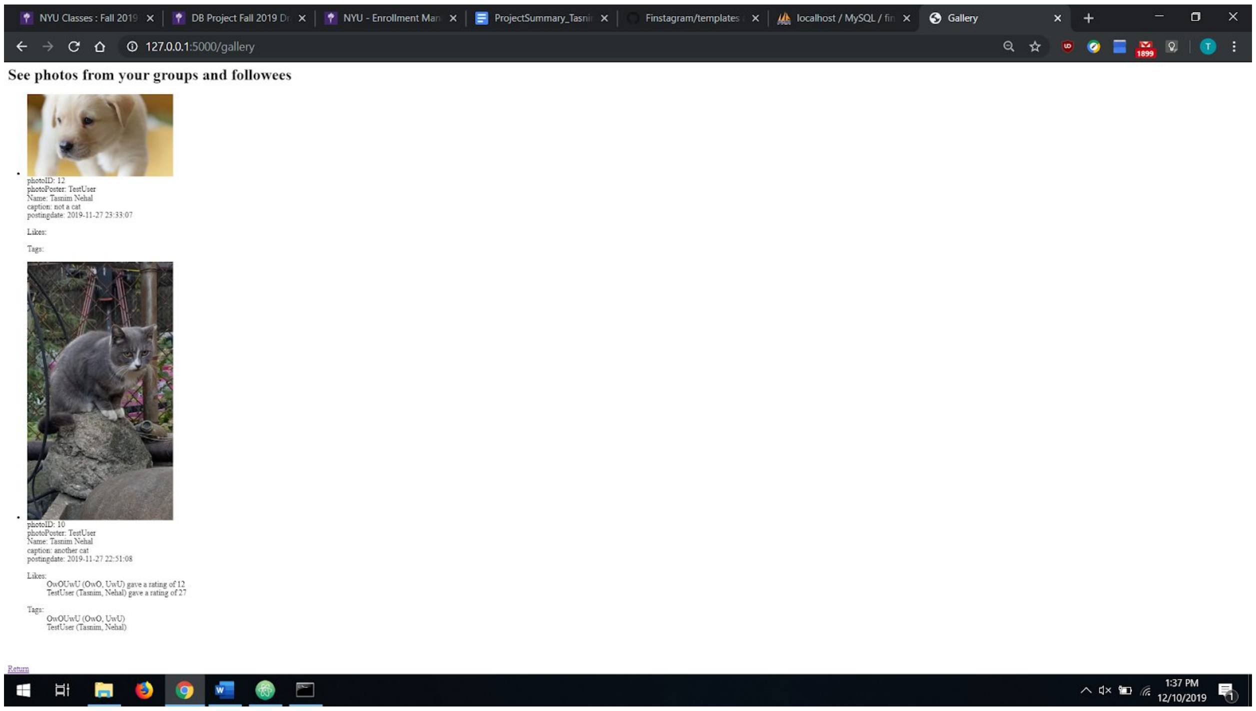
likeQuery = "SELECT * FROM likes NATURAL JOIN person"

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:



```python
#function to see other people's photos and additional info like id, time, likes, etc.
@app.route("/gallery", methods=["GET"])
@login_required
def gallery():
    #3 queries to get the visible photos, tags, and Likes
    #select everything from likes and tags and use the jinja template in gallery.html to display the righ
    photoQuery = "SELECT DISTINCT photoID, postingdate, caption, photoPoster, filepath, firstName, lastNa
    tagQuery = "SELECT * FROM tagged NATURAL JOIN person WHERE tagstatus=True"
    likeQuery = "SELECT * FROM likes NATURAL JOIN person"
    currentUser=session["username"]

    with connection.cursor() as cursor:
        cursor.execute(photoQuery, (currentUser, currentUser))
        photos=cursor.fetchall()

        cursor.execute(tagQuery)
        tags=cursor.fetchall()

        cursor.execute(likeQuery)
        likes=cursor.fetchall()

    return render_template("gallery.html", photos=photos, likes=likes, tags=tags)
```

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.



See photos from your groups and followees

- photoID: 12
  photoPoster: TestUser
  Name: Tasnim Nehal
  caption: not a cat
  postingdate: 2019-11-27 23:33:07

  Likes:

  Tags:

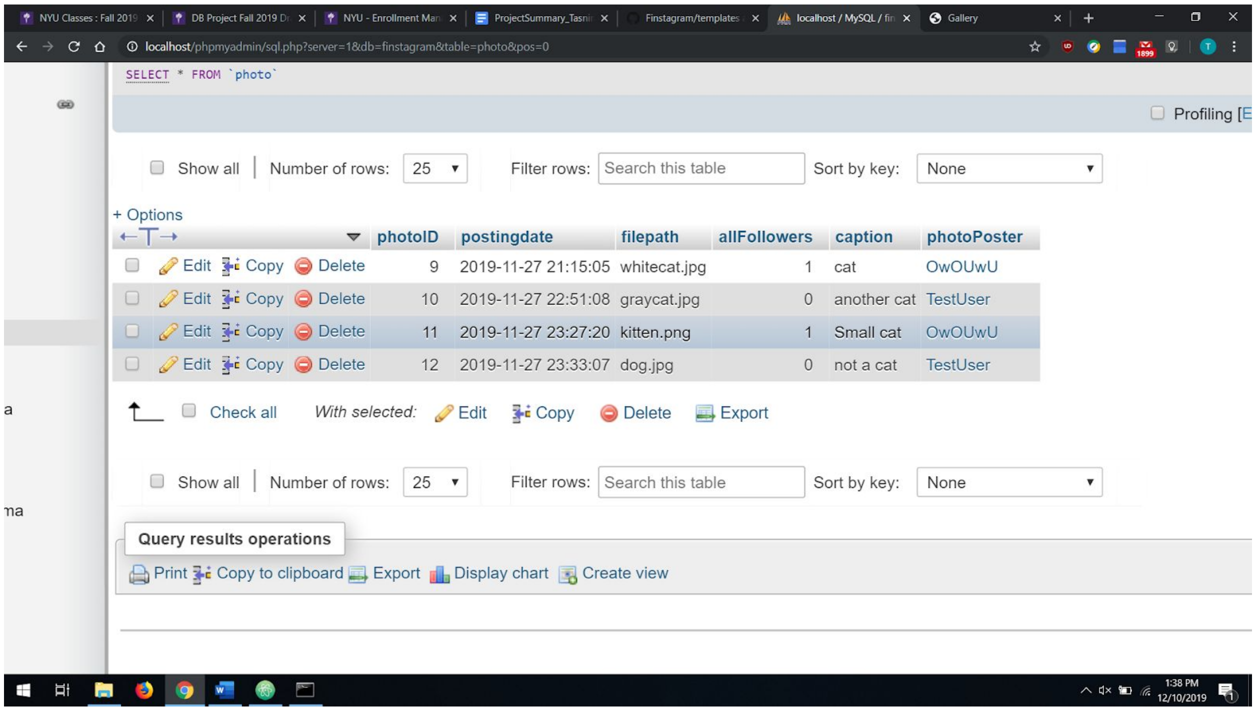
- photoID: 10
  photoPoster: TestUser
  Name: Tasnim Nehal
  caption: another cat
  postingdate: 2019-11-27 22:51:08

  Likes:
  OwOUwU (OwO, UwU) gave a rating of 12
  TestUser (Tasnim, Nehal) gave a rating of 27

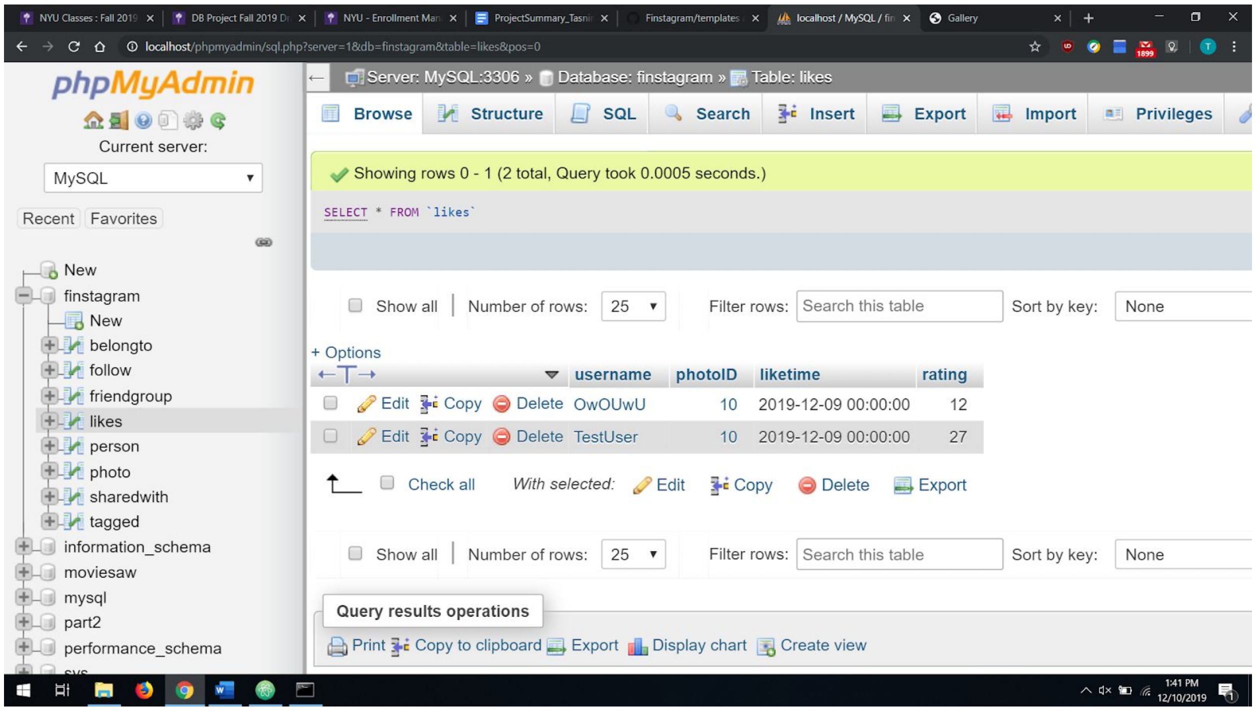
  Tags:
  OwOUwU (OwO, UwU)
  TestUser (Tasnim, Nehal)

Return

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)

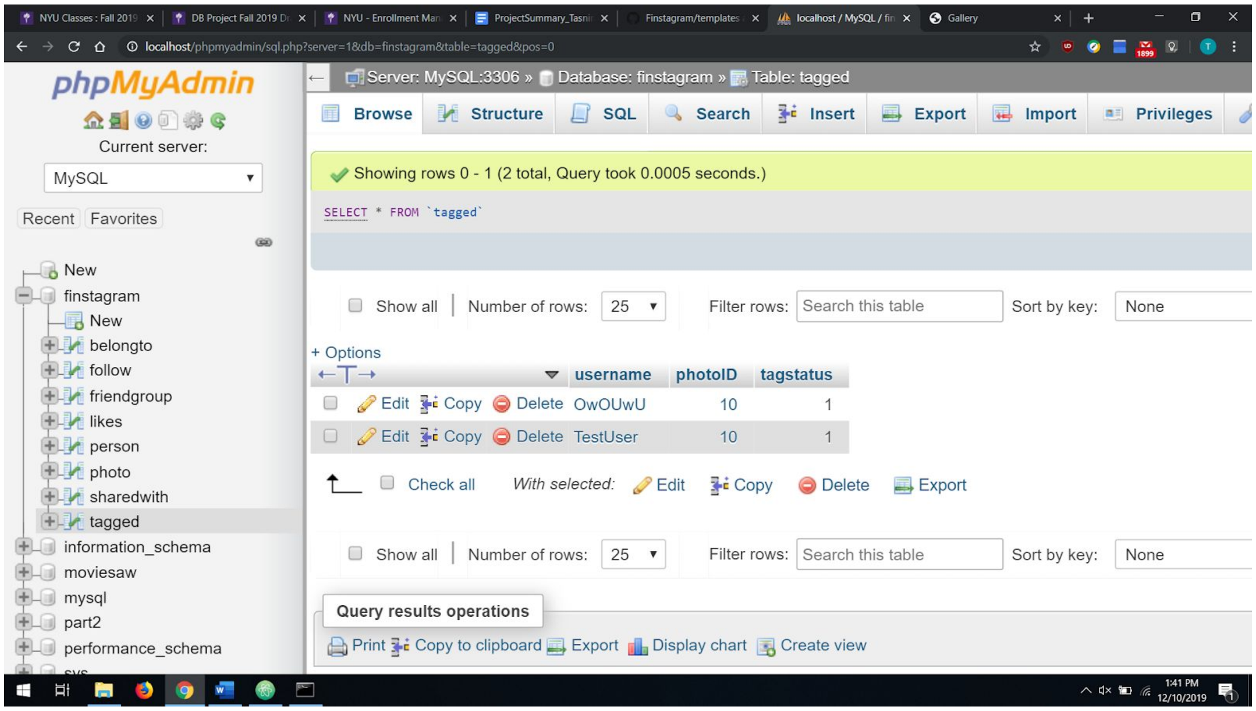

`SELECT * FROM `photo``

photoID	postingdate	filepath	allFollowers	caption	photoPoster
9	2019-11-27 21:15:05	whitecat.jpg	1	cat	OwOUwU
10	2019-11-27 22:51:08	graycat.jpg	0	another cat	TestUser
11	2019-11-27 23:27:20	kitten.png	1	Small cat	OwOUwU
12	2019-11-27 23:33:07	dog.jpg	0	not a cat	TestUser


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

`SELECT * FROM `likes``

username	photoID	liketime	rating
OwOUwU	10	2019-12-09 00:00:00	12
TestUser	10	2019-12-09 00:00:00	27



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

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

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

username	photoID	tagstatus
OwOUwU	10	1
TestUser	10	1

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)"

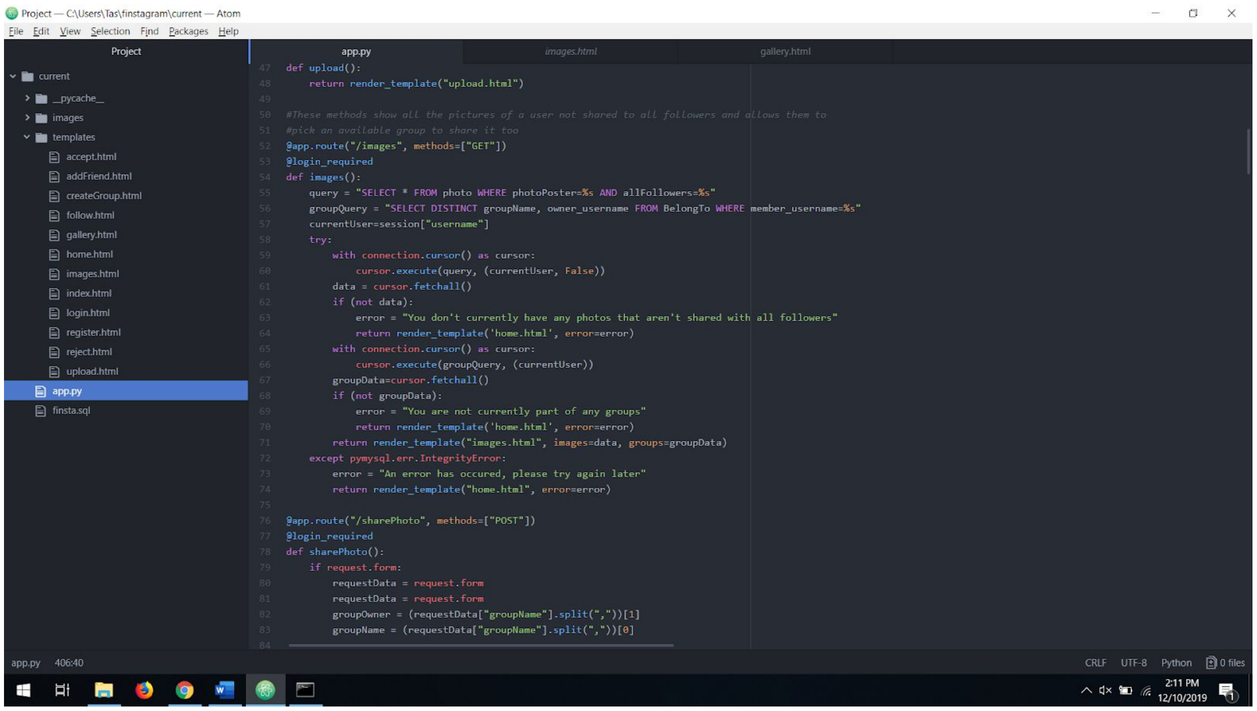
searchQuery = "SELECT DISTINCT groupName, owner_username FROM BelongTo WHERE member_username=%s"

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

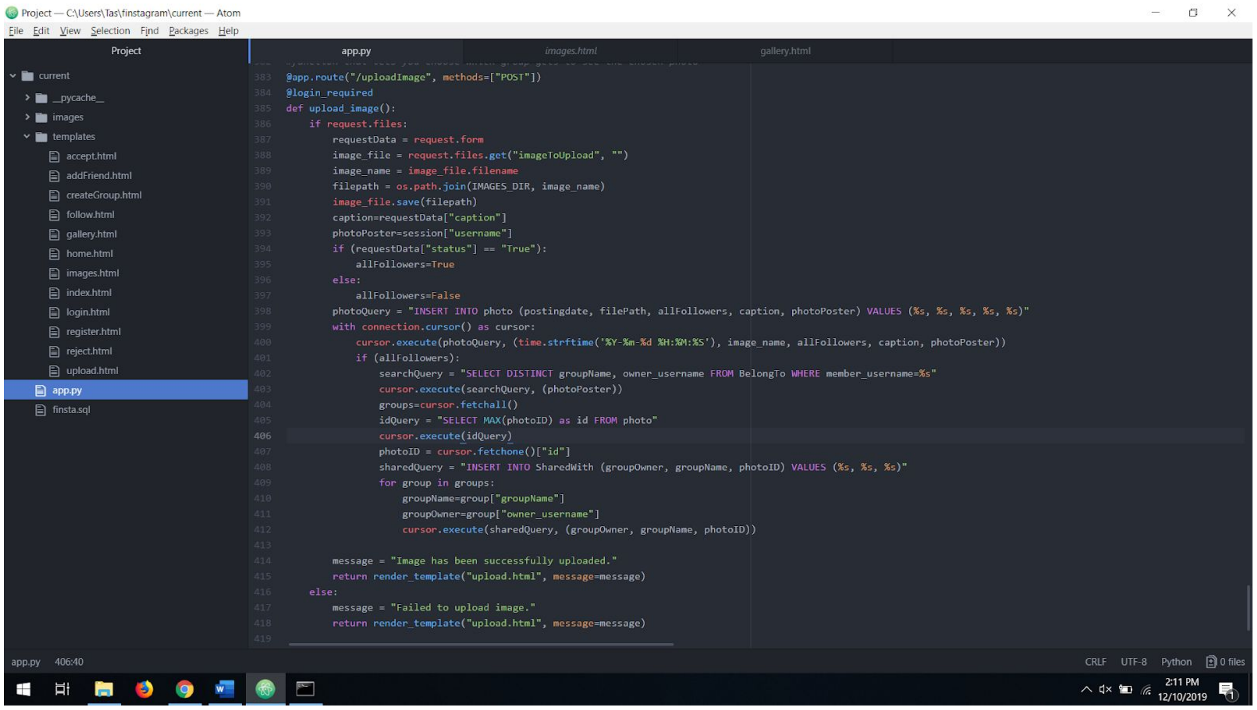
sharedQuery = "INSERT INTO SharedWith (groupOwner, groupName, photoID) VALUES (%s, %s, %s)"

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..



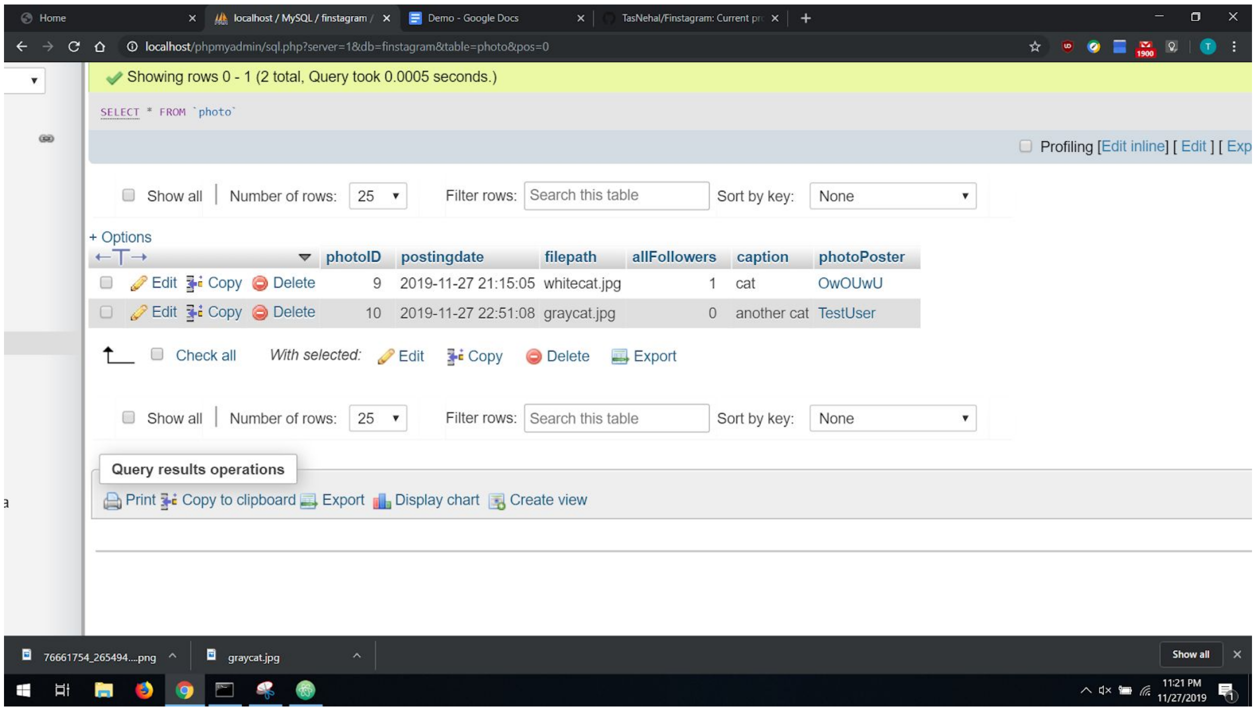
```python
def upload():
    return render_template("upload.html")

#These methods show all the pictures of a user not shared to all followers and allows them to
#pick an available group to share it too
@app.route("/images", methods=["GET"])
@login_required
def images():
    query = "SELECT * FROM photo WHERE photoPoster=%s AND allFollowers=%s"
    groupQuery = "SELECT DISTINCT groupName, owner_username FROM BelongTo WHERE member_username=%s"
    currentUser=session["username"]
    try:
        with connection.cursor() as cursor:
            cursor.execute(query, (currentUser, False))
        data = cursor.fetchall()
        if (not data):
            error = "You don't currently have any photos that aren't shared with all followers"
            return render_template('home.html', error=error)
        with connection.cursor() as cursor:
            cursor.execute(groupQuery, (currentUser))
        groupData=cursor.fetchall()
        if (not groupData):
            error = "You are not currently part of any groups"
            return render_template('home.html', error=error)
        return render_template("images.html", images=data, groups=groupData)
    except pymysql.err.IntegrityError:
        error = "An error has occured, please try again later"
        return render_template("home.html", error=error)

@app.route("/sharePhoto", methods=["POST"])
@login_required
def sharePhoto():
    if request.form:
        requestData = request.form
        requestData = request.form
        groupOwner = (requestData["groupName"].split(","))[1]
        groupName = (requestData["groupName"].split(","))[0]
```



```python
@app.route("/uploadImage", methods=["POST"])
@login_required
def upload_image():
    if request.files:
        requestData = request.form
        image_file = request.files.get("imageToUpload", "")
        image_name = image_file.filename
        filepath = os.path.join(IMAGES_DIR, image_name)
        image_file.save(filepath)
        caption=requestData["caption"]
        photoPoster=session["username"]
        if (requestData["status"] == "True"):
            allFollowers=True
        else:
            allFollowers=False
        photoQuery = "INSERT INTO photo (postingdate, filePath, allFollowers, caption, photoPoster) VALUES (%s, %s, %s, %s, %s)"
        with connection.cursor() as cursor:
            cursor.execute(photoQuery, (time.strftime('%Y-%m-%d %H:%M:%S'), image_name, allFollowers, caption, photoPoster))
            if (allFollowers):
                searchQuery = "SELECT DISTINCT groupName, owner_username FROM BelongTo WHERE member_username=%s"
                cursor.execute(searchQuery, (photoPoster))
                groups=cursor.fetchall()
                idQuery = "SELECT MAX(photoID) as id FROM photo"
                cursor.execute(idQuery)
                photoID = cursor.fetchone()["id"]
                sharedQuery = "INSERT INTO SharedWith (groupOwner, groupName, photoID) VALUES (%s, %s, %s)"
                for group in groups:
                    groupName=group["groupName"]
                    groupOwner=group["owner_username"]
                    cursor.execute(sharedQuery, (groupOwner, groupName, photoID))

        message = "Image has been successfully uploaded."
        return render_template("upload.html", message=message)
    else:
        message = "Failed to upload image."
        return render_template("upload.html", message=message)
```

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:

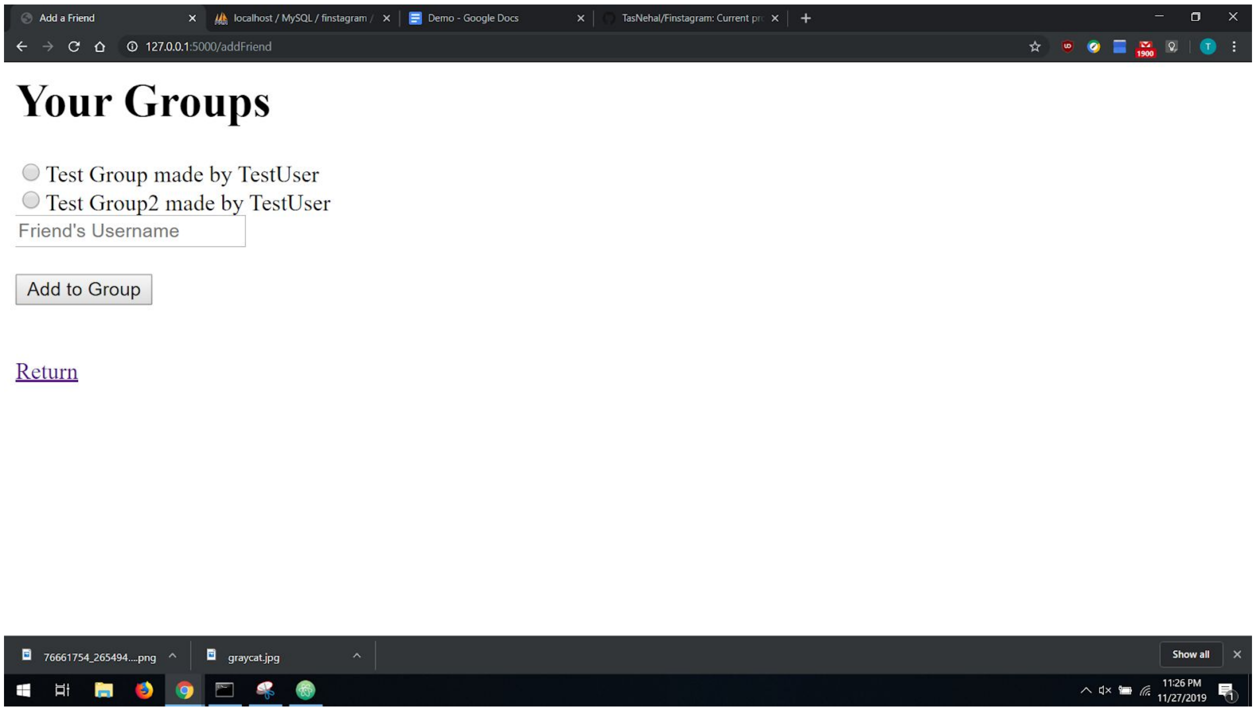


Showing rows 0 - 1 (2 total, Query took 0.0005 seconds.)

```
SELECT * FROM `photo`
```

photoID	postingdate	filepath	allFollowers	caption	photoPoster
9	2019-11-27 21:15:05	whitecat.jpg	1	cat	OwOUwU
10	2019-11-27 22:51:08	graycat.jpg	0	another cat	TestUser

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



Your Groups

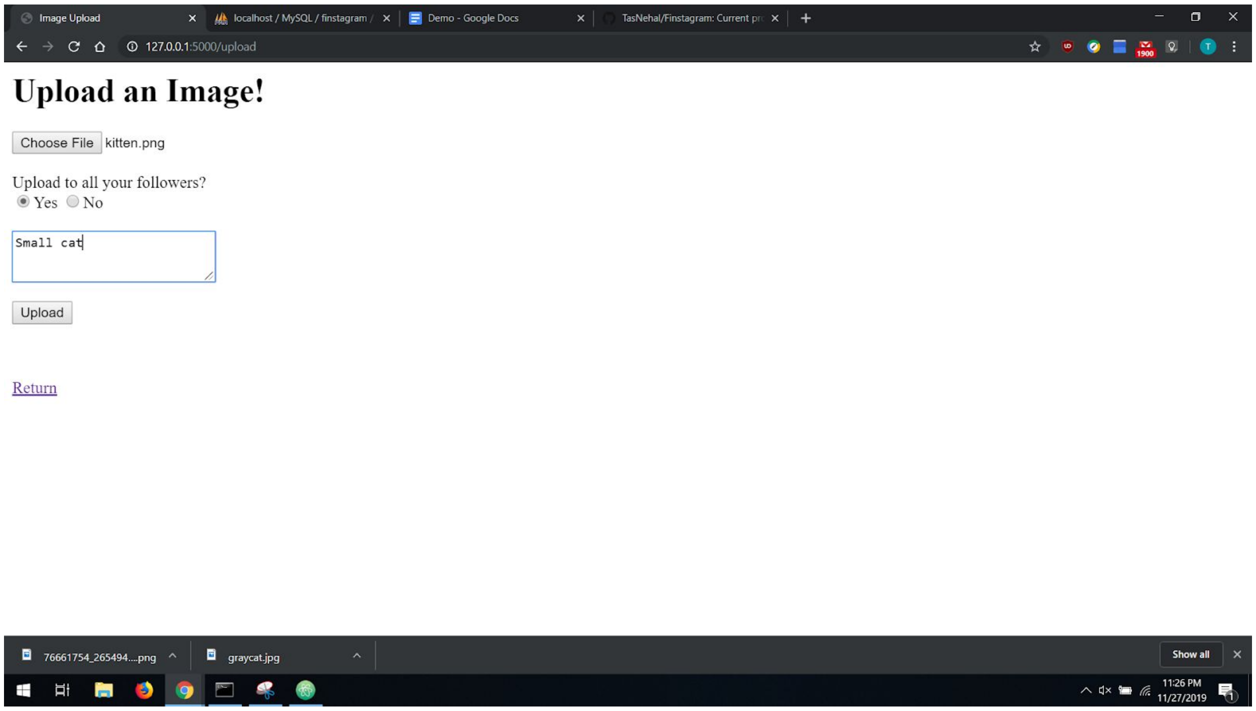
- Test Group made by TestUser
- Test Group2 made by TestUser

Friend's Username

Add to Group

Return

User uploads photo of a kitten



127.0.0.1:5000/upload

Upload an Image!

Choose File kitten.png

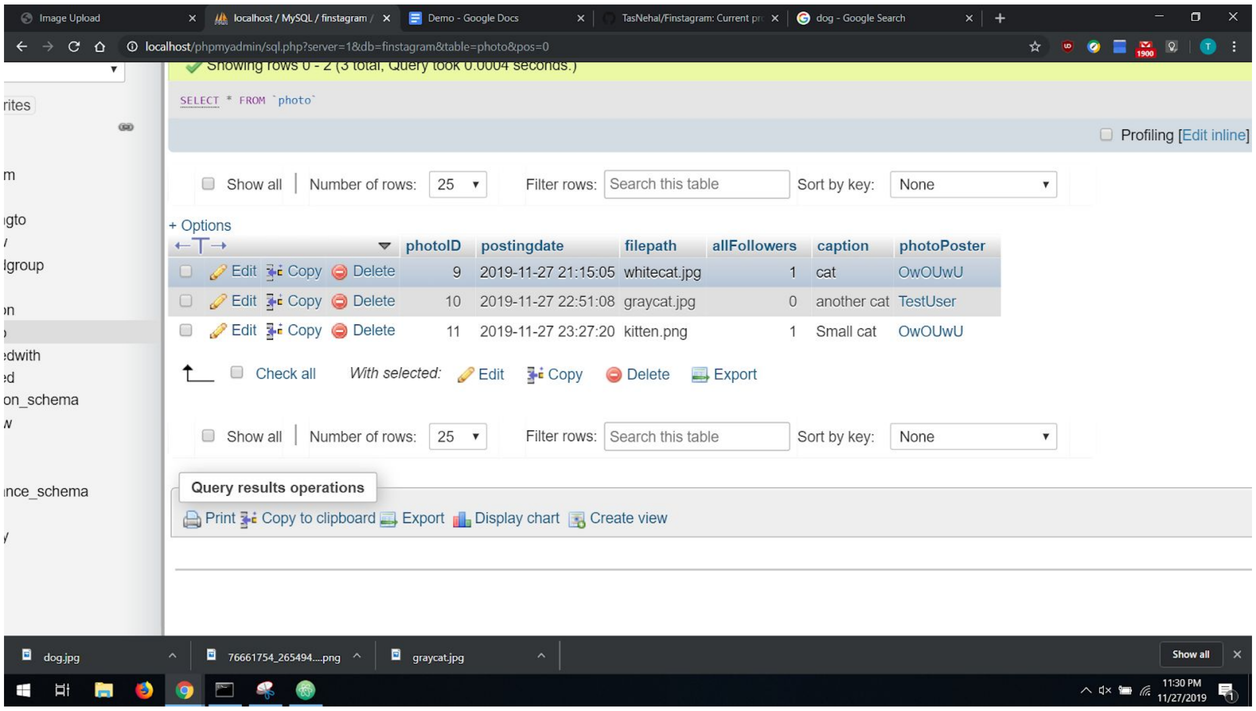
Upload to all your followers?
[✓ Yes] No

Small cat

Upload

Return

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



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

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

```
SELECT * FROM `photo`
```

photoID	postingdate	filepath	allFollowers	caption	photoPoster
9	2019-11-27 21:15:05	whitecat.jpg	1	cat	OwOUwU
10	2019-11-27 22:51:08	graycat.jpg	0	another cat	TestUser
11	2019-11-27 23:27:20	kitten.png	1	Small cat	OwOUwU

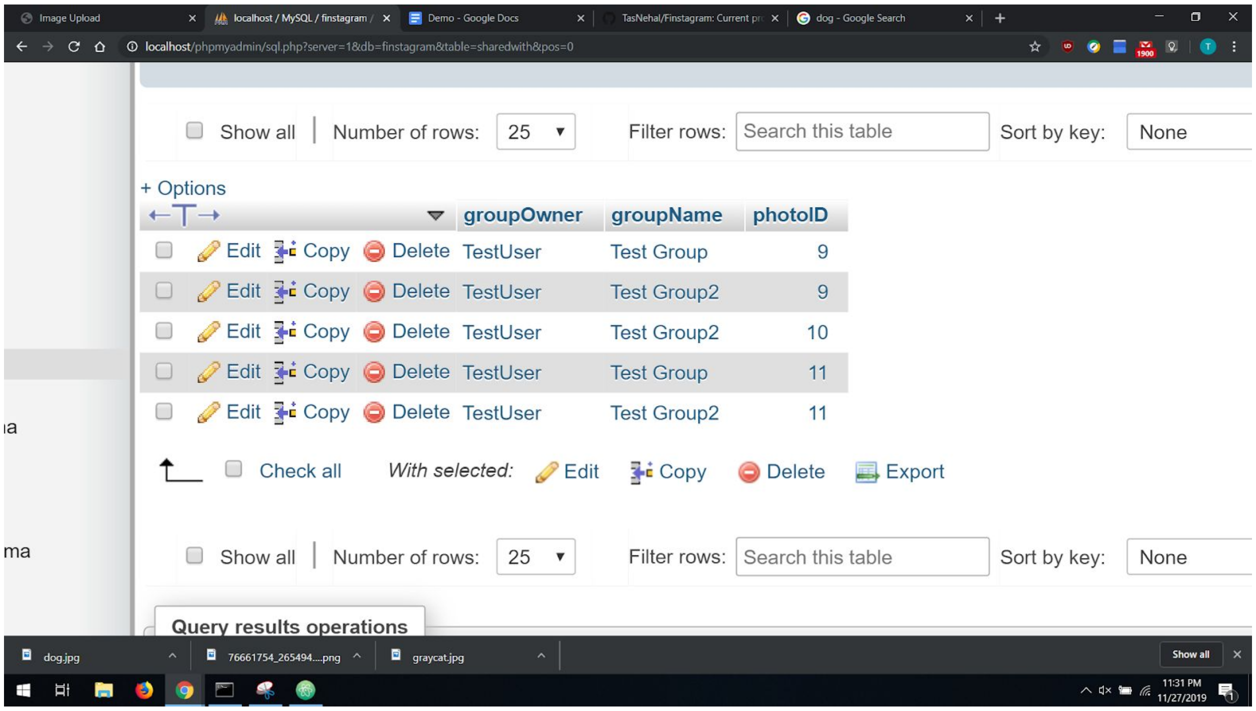


Image Upload
localhost / MySQL / finstagram /
Demo - Google Docs
TasNehal/Finstagram: Current pr
dog - Google Search
localhost/phpmyadmin/sql.php?server=1&db=finstagram&table=sharedwith&pos=0
Show all
Number of rows: 25
Filter rows: Search this table
Sort by key: None
+ Options

	groupOwner	groupName	photoID
Edit Copy Delete	TestUser	Test Group	9
Edit Copy Delete	TestUser	Test Group2	9
Edit Copy Delete	TestUser	Test Group2	10
Edit Copy Delete	TestUser	Test Group	11
Edit Copy Delete	TestUser	Test Group2	11

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

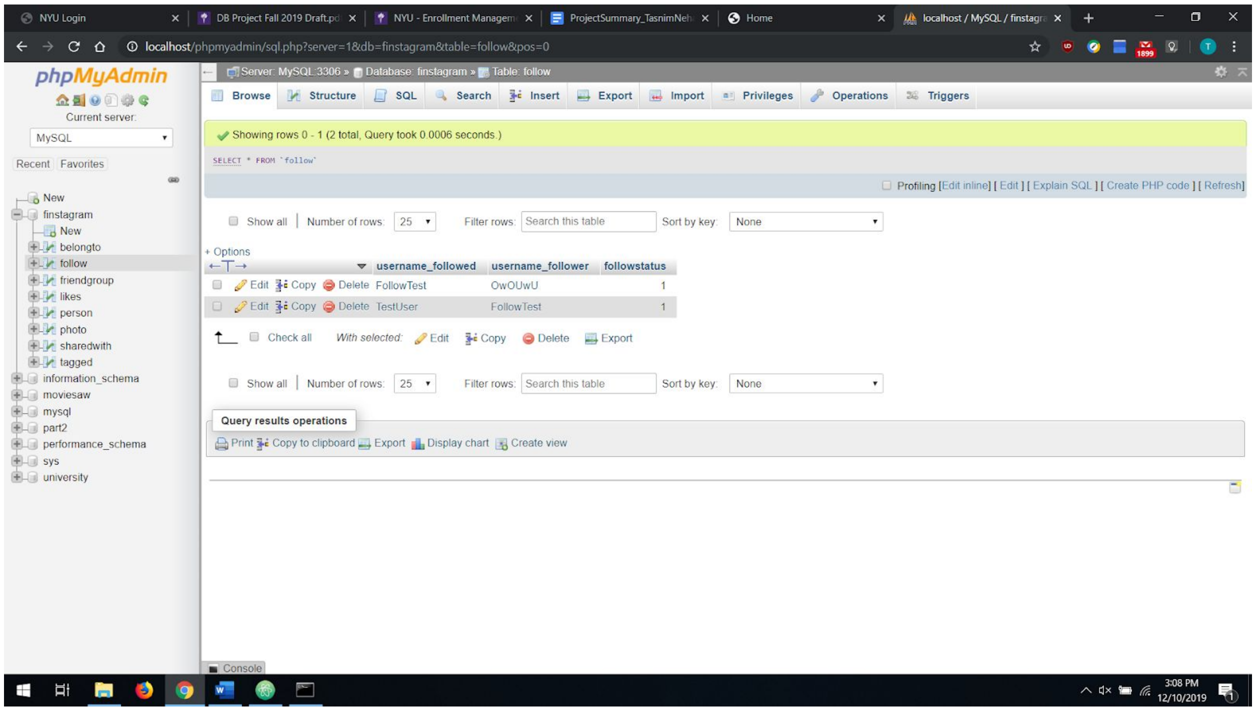
username_follower=%s"

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:

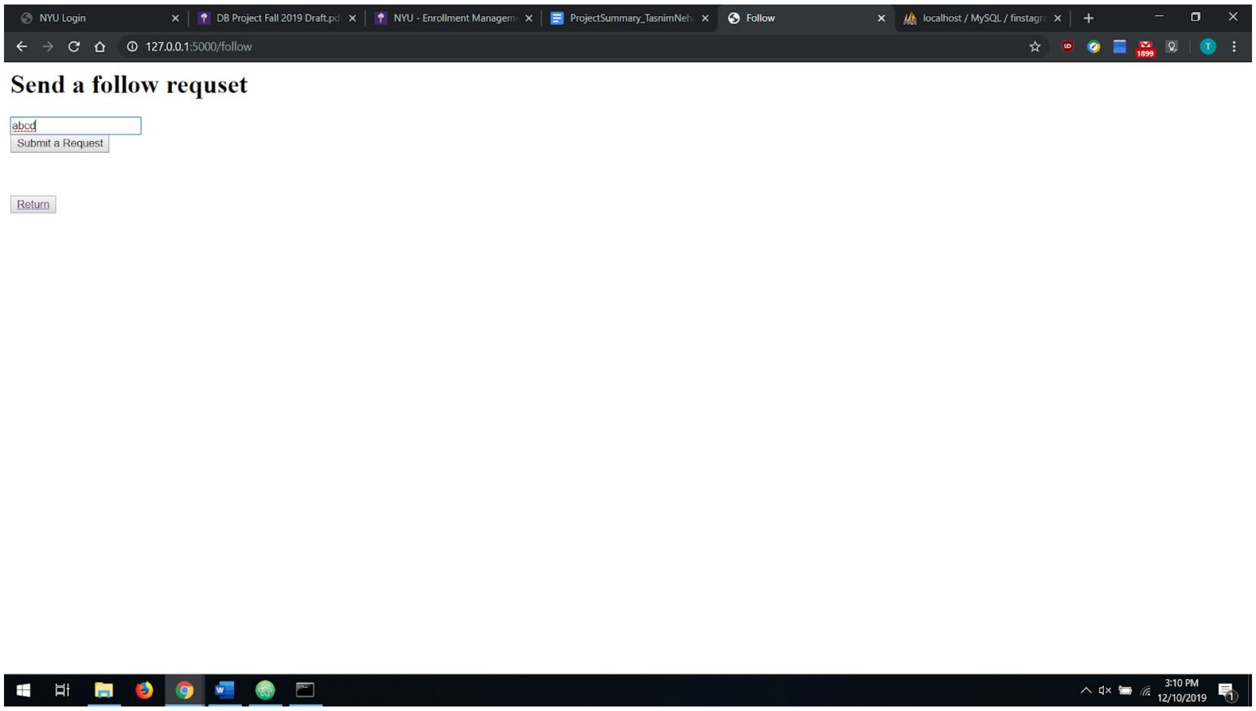


Showing rows 0 - 1 (2 total, Query took 0.0006 seconds.)

```
SELECT * FROM `follow`
```

username_followed	username_follower	followstatus
FollowTest	OwOUwU	1
TestUser	FollowTest	1

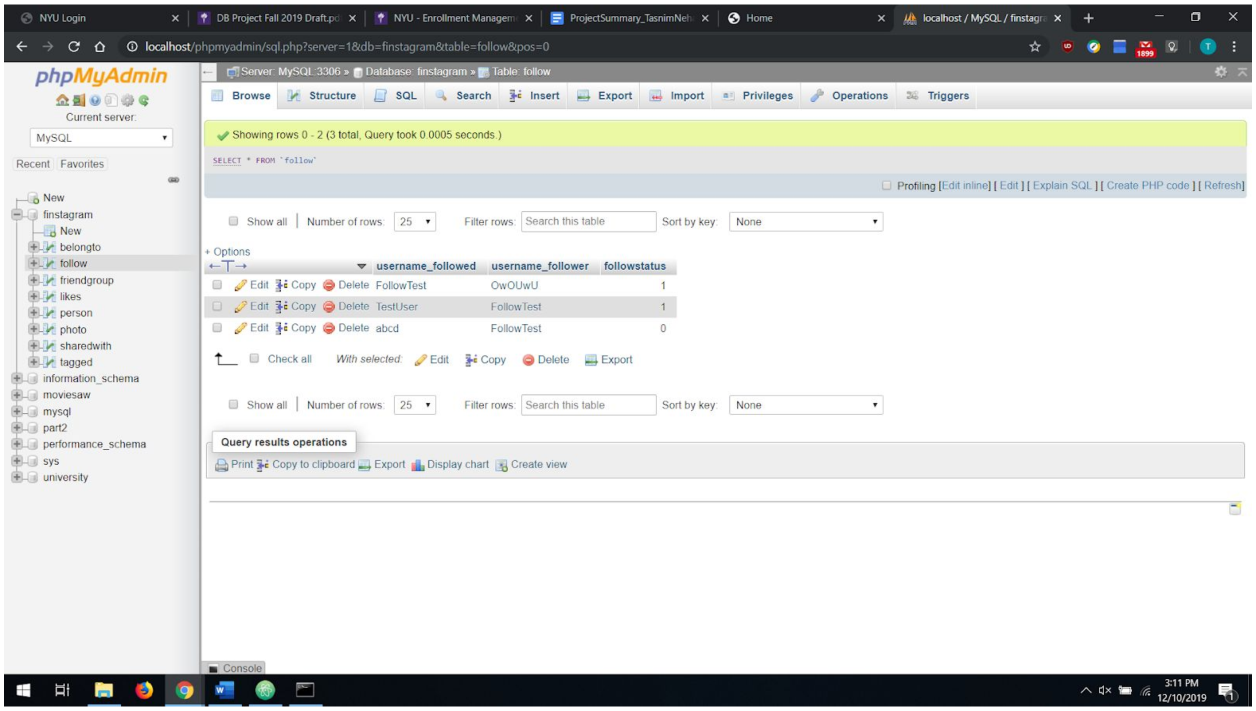
User FollowTest has to send a follow request to an existing user, in this case user abcd, or the request will not go through. Only the user on the receiving end can accept the request.



Send a follow requset

abcd

The request is sent

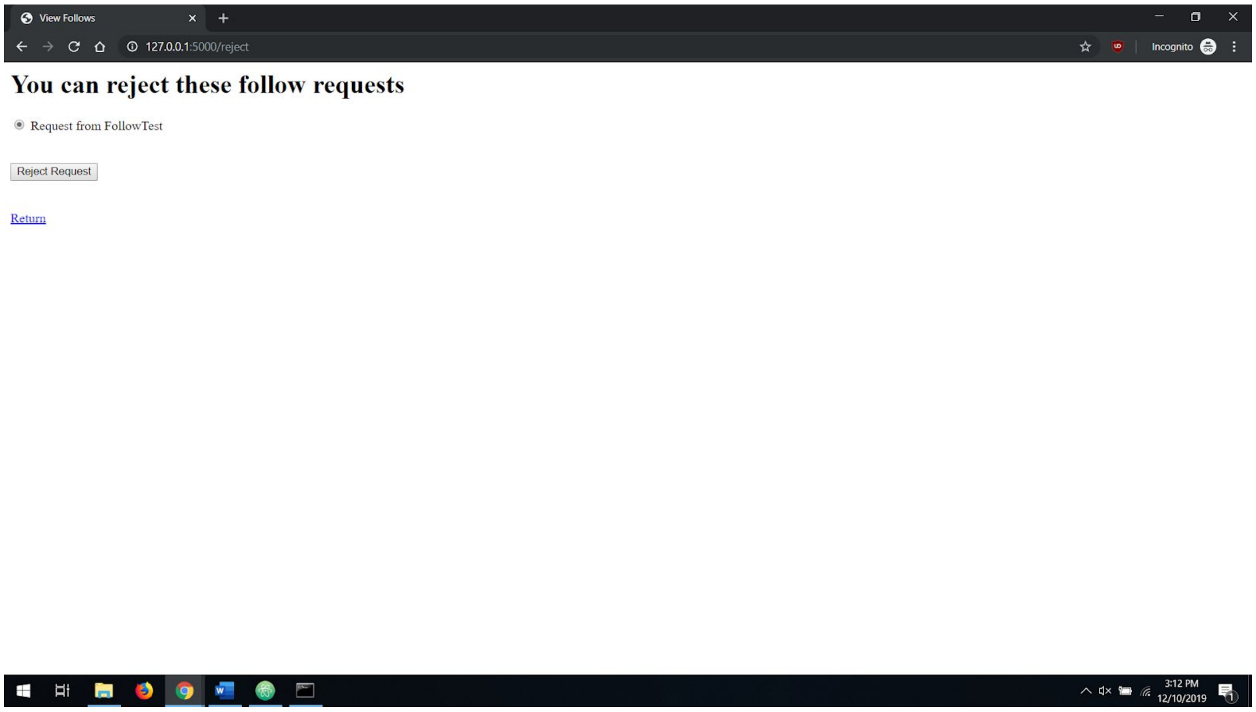

localhost/phpmyadmin/sql.php?server=1&db=finstagram&table=follow&pos=0

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

```
SELECT * FROM `follow`
```

username_followed	username_follower	followstatus
FollowTest	OwOUwU	1
TestUser	FollowTest	1
abcd	FollowTest	0

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

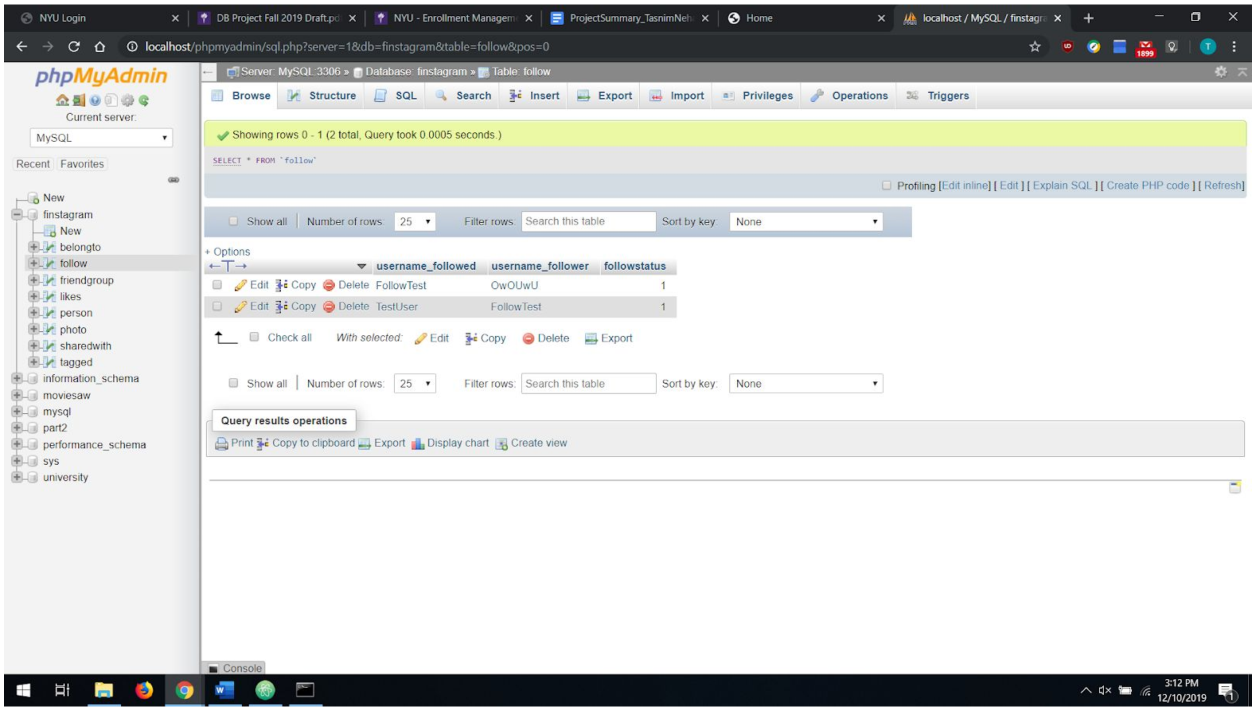

127.0.0.1:5000/reject

You can reject these follow requests

(●) Request from FollowTest

Reject Request

Return



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

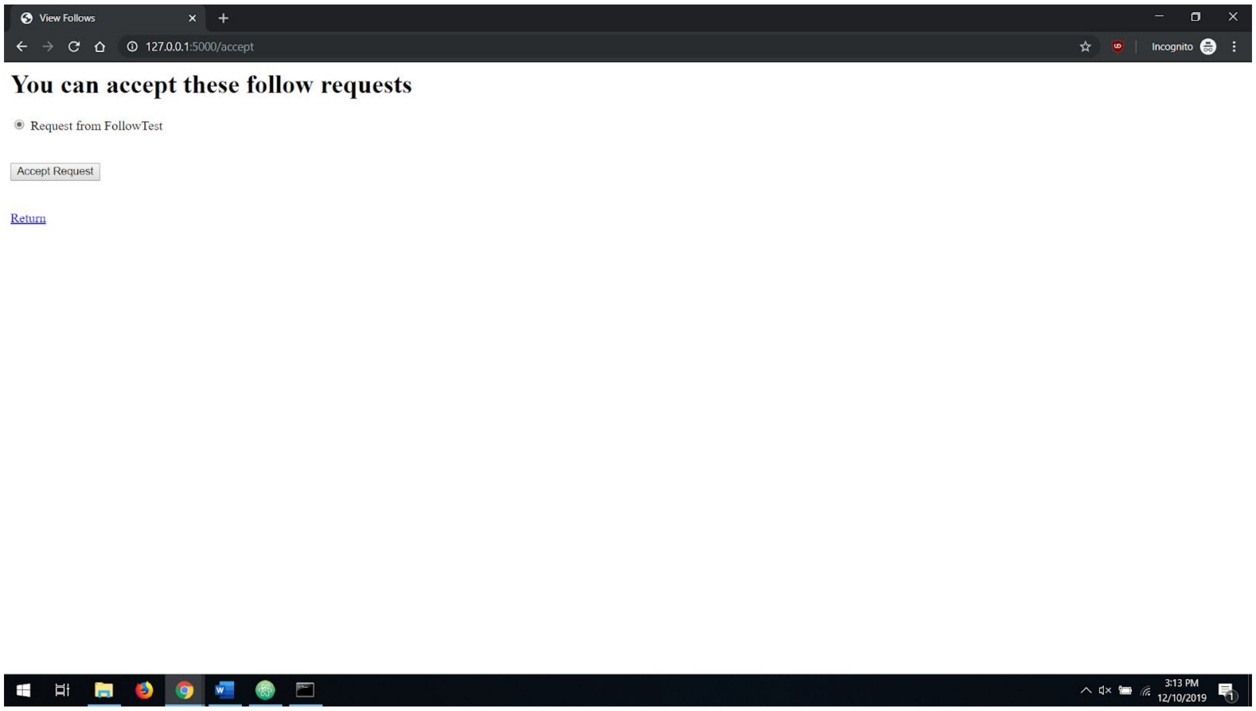
Server: MySQL:3306 » Database: finstagram » Table: follow

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

```
SELECT * FROM `follow`
```

username_followed	username_follower	followstatus
FollowTest	OwOUwU	1
TestUser	FollowTest	1

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



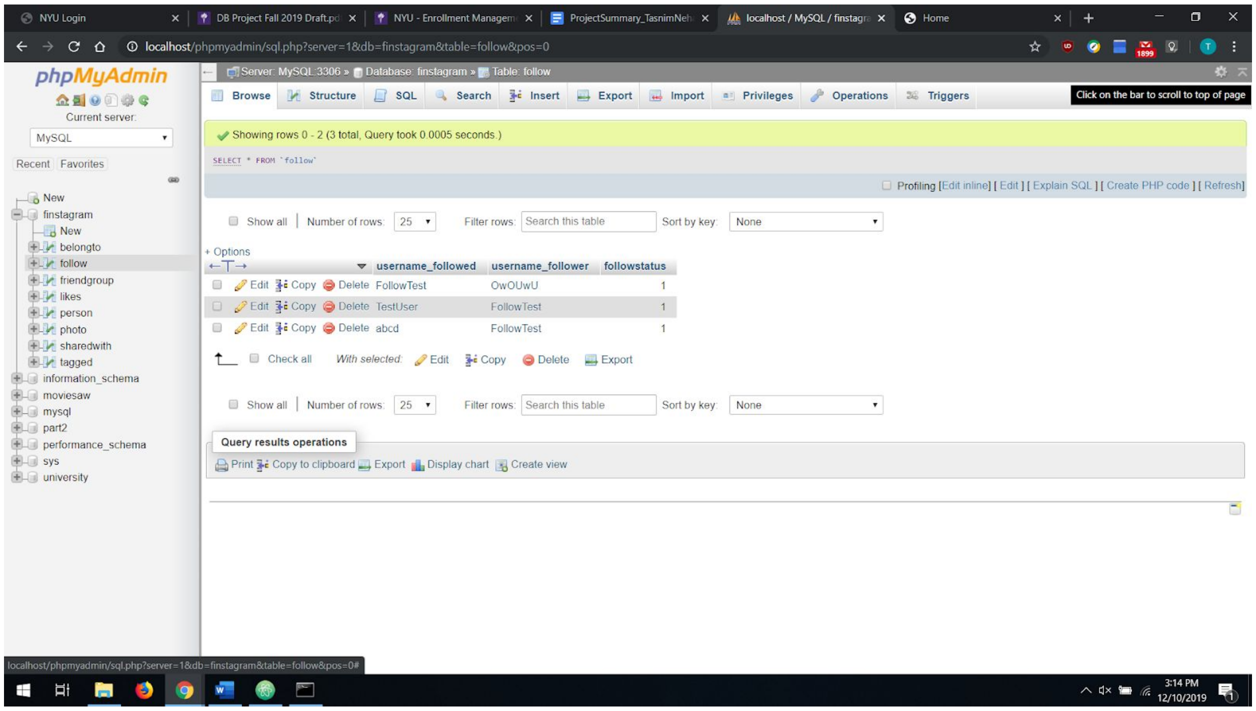
127.0.0.1:5000/accept

You can accept these follow requests

Request from FollowTest

Accept Request

Return



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

```
SELECT * FROM `follow`
```

username_followed	username_follower	followstatus
FollowTest	OwOUwU	1
TestUser	FollowTest	1
abcd	FollowTest	1

Feature 5:

1. Add friend group

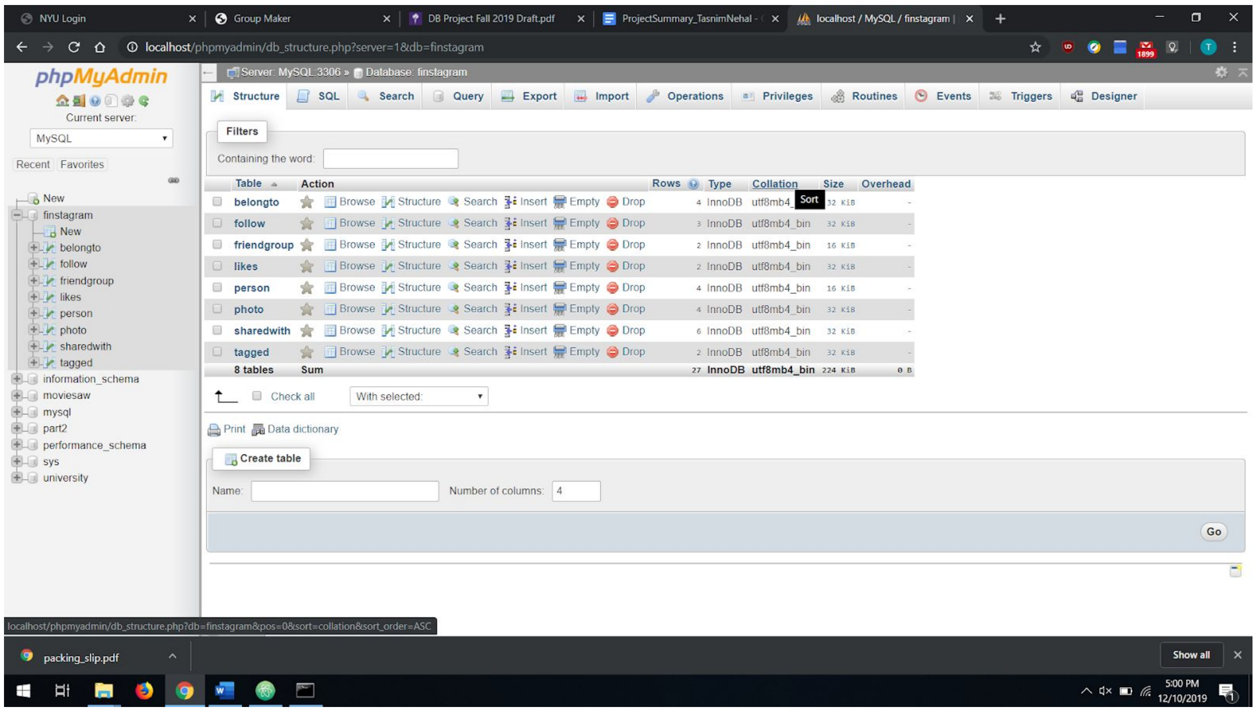
2. The first query creates the group itself while the second query adds the creator of the group as a member of that group that was just made

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

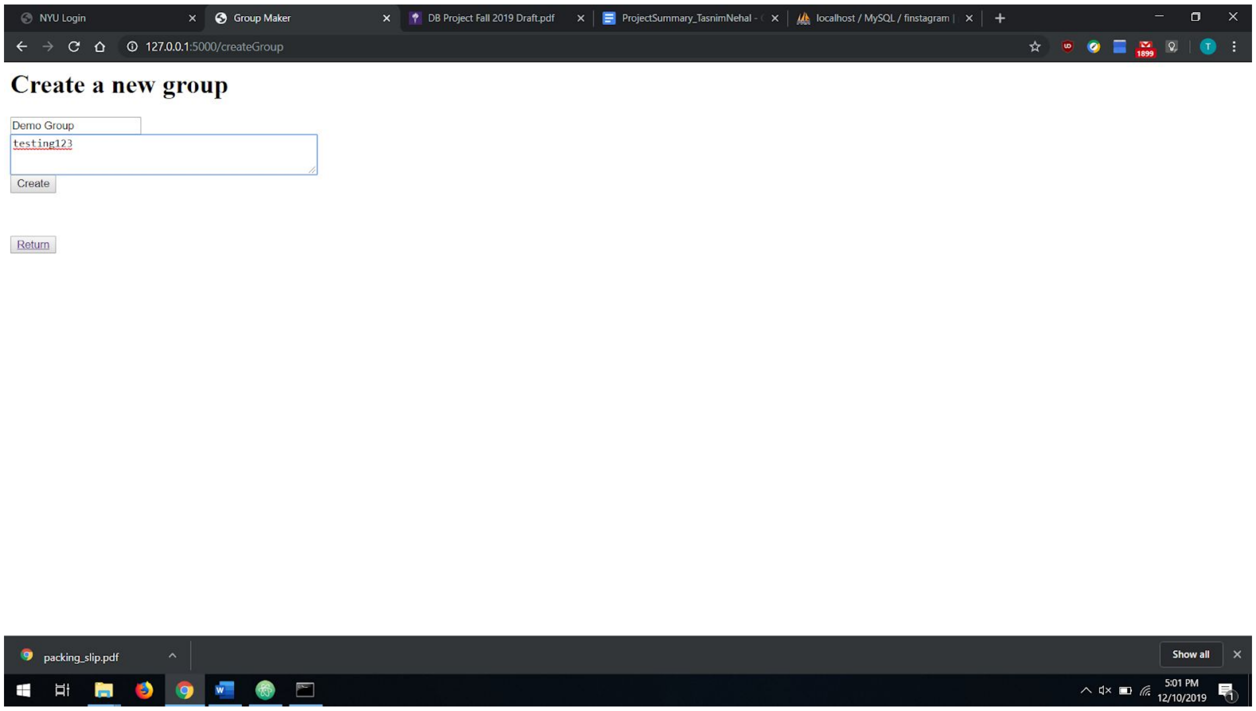


Server: MySQL:3306 » Database: finstagram

Table	Action	Rows	Type	Collation	Size	Overhead
belongto	Browse Structure Search Insert Empty Drop	4	InnoDB	utf8mb4_	32 KiB	-
follow	Browse Structure Search Insert Empty Drop	3	InnoDB	utf8mb4_bin	32 KiB	-
friendgroup	Browse Structure Search Insert Empty Drop	2	InnoDB	utf8mb4_bin	16 KiB	-
likes	Browse Structure Search Insert Empty Drop	2	InnoDB	utf8mb4_bin	32 KiB	-
person	Browse Structure Search Insert Empty Drop	4	InnoDB	utf8mb4_bin	16 KiB	-
photo	Browse Structure Search Insert Empty Drop	4	InnoDB	utf8mb4_bin	32 KiB	-
sharedwith	Browse Structure Search Insert Empty Drop	6	InnoDB	utf8mb4_bin	32 KiB	-
tagged	Browse Structure Search Insert Empty Drop	2	InnoDB	utf8mb4_bin	32 KiB	-
8 tables	Sum	27	InnoDB	utf8mb4_bin	224 KiB	0 B

localhost/phpmyadmin/db_structure.php?server=1&db=finstagram

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


NYU Login
Group Maker
DB Project Fall 2019 Draft.pdf
ProjectSummary_TasnimNehal
localhost / MySQL / finstagram
127.0.0.1:5000/createGroup
packing_slip.pdf
Show all
5:01 PM
12/10/2019

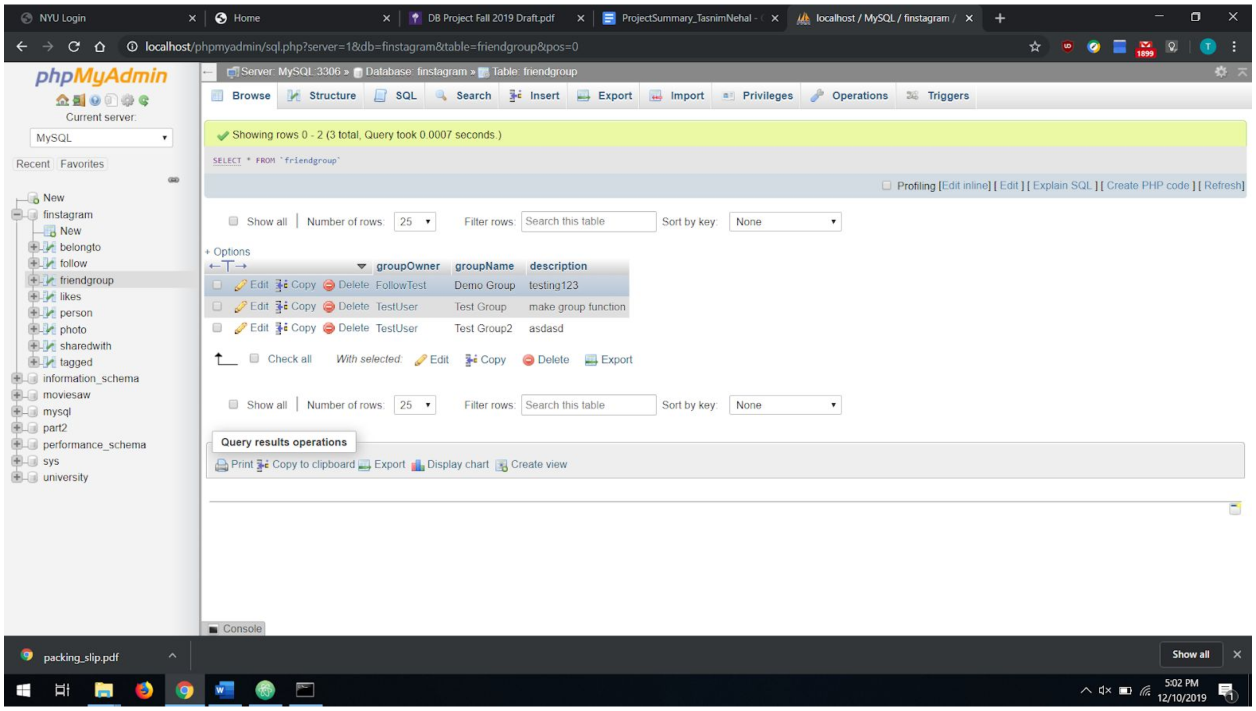
Create a new group

Demo Group

testing123

Create

Return



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

```
SELECT * FROM `friendgroup`
```

	groupOwner	groupName	description
Edit Copy Delete	FollowTest	Demo Group	testing123
Edit Copy Delete	TestUser	Test Group	make group function
Edit Copy Delete	TestUser	Test Group2	asdasd

Feature 6:

1. Add friend

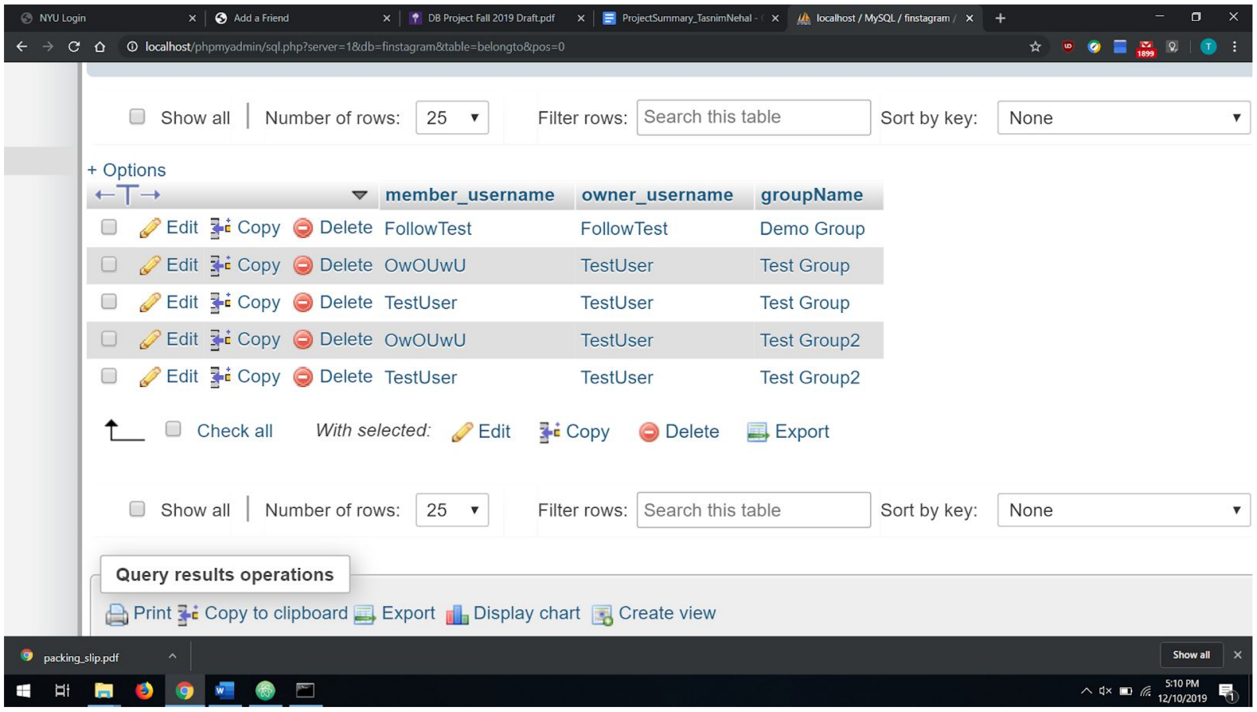
2. The first query displays the current groups the user is a member of and therefore the groups a user is allowed to add a member to. The second query inserts the chosen user into the group that was selected if that user wasn't already there.

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

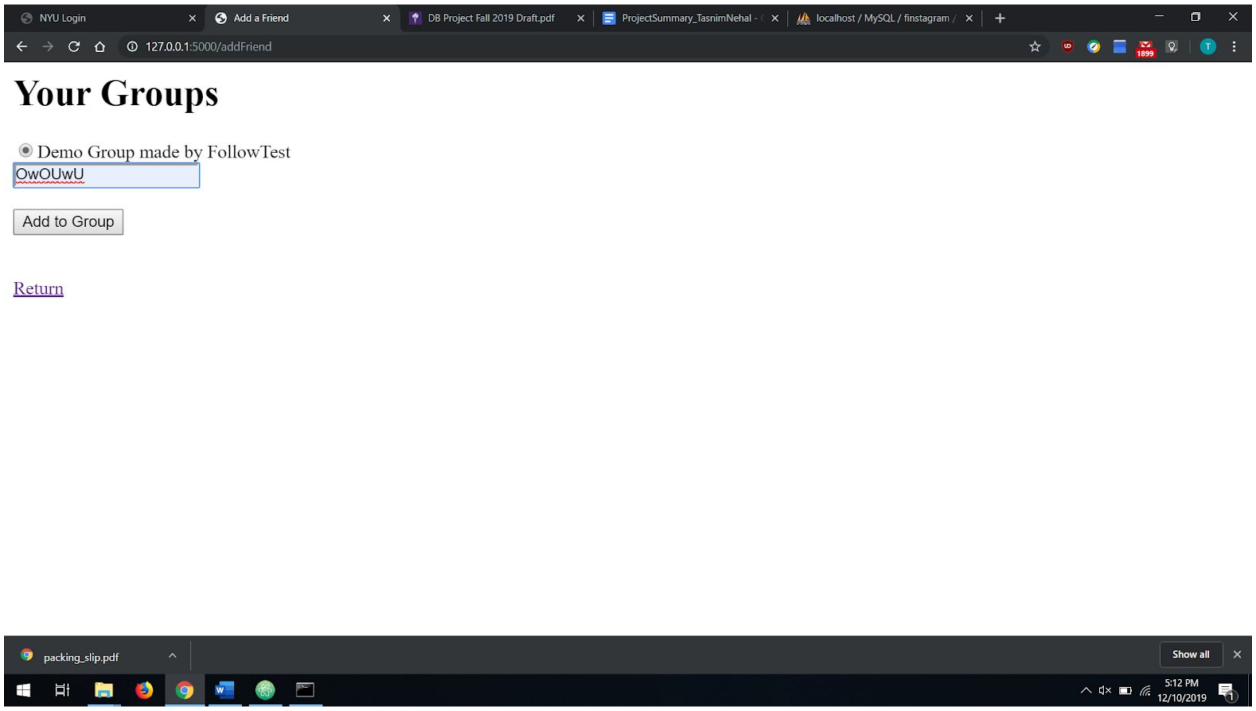
query = "INSERT INTO BelongTo (member_username, owner_username, groupName) VALUES (%s, %s, %s)"

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.


NYU Login
Add a Friend
DB Project Fall 2019 Draft.pdf
ProjectSummary_TasnimNehal
localhost / MySQL / finstagram
localhost/phpmyadmin/sql.php?server=1&db=finstagram&table=belongto&pos=0
Show all
Number of rows: 25
Filter rows: Search this table
Sort by key: None
+ Options
Check all
With selected: Edit Copy Delete Export
Query results operations
Print
Copy to clipboard
Export
Display chart
Create view
packing_slip.pdf
Show all
5:10 PM
12/10/2019

member_username	owner_username	groupName
FollowTest	FollowTest	Demo Group
OwOUwU	TestUser	Test Group
TestUser	TestUser	Test Group
OwOUwU	TestUser	Test Group2
TestUser	TestUser	Test Group2


NYU Login
Add a Friend
DB Project Fall 2019 Draft.pdf
ProjectSummary_TasnimNehal
localhost / MySQL / finstagram /
127.0.0.1:5000/addFriend
packing_slip.pdf
Show all
5:12 PM
12/10/2019

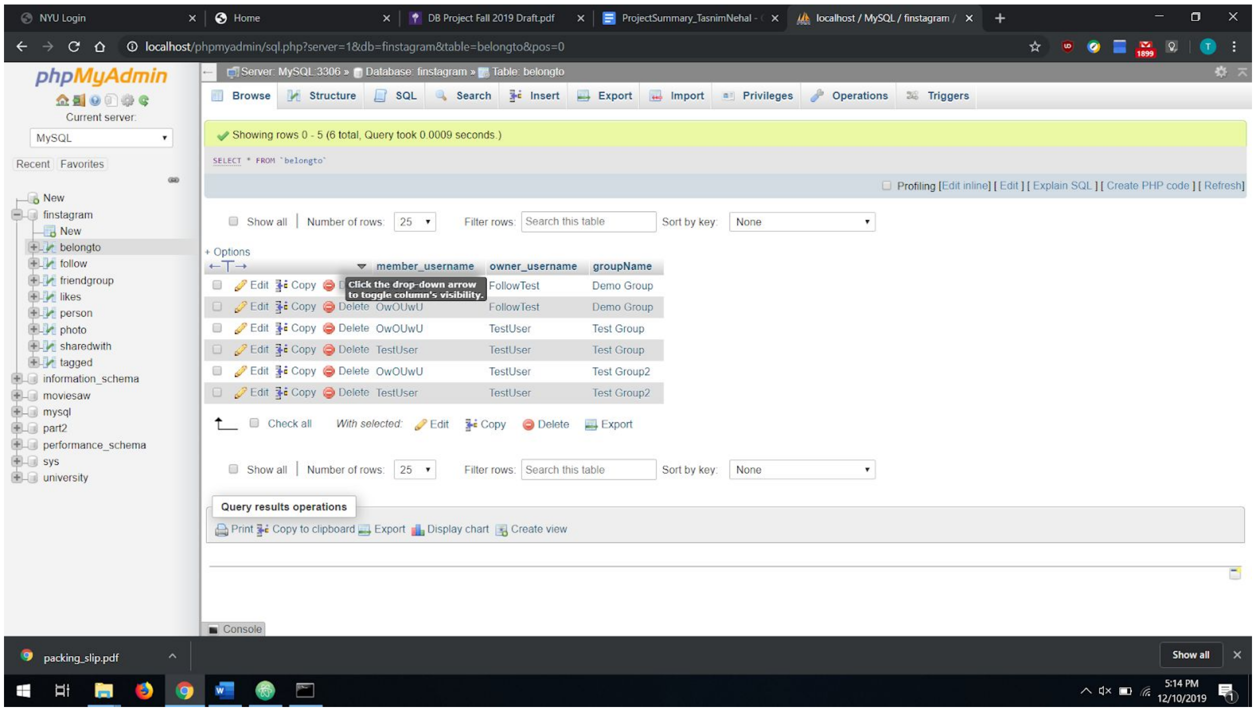
Your Groups

● Demo Group made by FollowTest

OwOUwU

Add to Group

Return



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

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

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

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

	member_username	owner_username	groupName
Edit Copy Delete	Click the drop-down arrow to toggle column's visibility.	FollowTest	Demo Group
Edit Copy Delete	OwOUwU	FollowTest	Demo Group
Edit Copy Delete	OwOUwU	TestUser	Test Group
Edit Copy Delete	TestUser	TestUser	Test Group
Edit Copy Delete	OwOUwU	TestUser	Test Group2
Edit Copy Delete	TestUser	TestUser	Test Group2

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