**Feature:** Manage Follows

**Name:** Michael Kwan

**Queries:**

Fine everyone the user is not following and is not themselves.

query = 'SELECT DISTINCT username FROM person WHERE username != %s and username NOT IN \  
 (SELECT username\_followed FROM follow WHERE username\_follower = %s)'  
cursor.execute(query, (follower, follower))

The user will follow the person in the follow table but the followstatus is set to 0.

ins = 'INSERT INTO follow VALUES(%s, %s, 0)'  
cursor.execute(ins, (username\_followed, username\_follower))

Determine if there is an entry in the follow table where someone is trying to follow the user.

query = 'SELECT DISTINCT username\_follower FROM follow WHERE \  
 username\_followed = %s and followStatus = 0'  
cursor.execute(query, username)

Accept the follow by setting the followstatus to 1 in that relationship.

upd = 'UPDATE follow SET followStatus = 1 WHERE username\_followed = %s AND username\_follower = %s'

**Locations:**

app.py (Lines 317-372)

select\_Person\_toFollow.html

selectRequestToAccept.html

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Feature:** Manage Tags

**Name:** Nahom Molla

**Queries:**

Accept the tag by changing the tagstatus to 1 in that relationship.

upd = 'UPDATE tagged SET tagStatus = 1 WHERE username = %s AND photoID = %s'

cursor.execute(upd, (username, photoID))

Determine if there is an entry in the follow table where someone is trying to follow the user and get all the photos the user is tagged in.

query = 'SELECT photoPoster, photoID FROM tagged \  
 NATURAL JOIN photo WHERE username = %s AND tagStatus = 0'

cursor.execute(query, (user))

Determine if the user has already been tagged in the photo.

query = 'SELECT \* FROM tagged WHERE username = %s AND photoID = %s'  
cursor.execute(query, (follower, photoID))

Tag the user with tagstatus of 1.

query = 'INSERT INTO tagged VALUES (%s, %s, 0)'  
cursor.execute(query, (follower, photoID))

Determine if the user is follows the poster of the photo or is in a friendgroup the photo is shared with.

query = 'CREATE OR REPLACE VIEW v AS \  
 ((SELECT photoID FROM Follow AS f JOIN \  
 Photo AS p on f.username\_followed = p.photoPoster \  
 WHERE f.username\_follower = %s AND f.followStatus = 1) \  
 UNION (SELECT photoID \  
 FROM (SharedWith AS sw JOIN BelongTo AS bt ON \  
 sw.groupOwner = bt.owner\_username AND \  
 sw.groupName = bt.groupName) NATURAL JOIN photo \  
 WHERE member\_username = %s))'  
cursor.execute(query, (follower, follower))  
conn.commit()  
  
query = 'SELECT photoID FROM v WHERE photoID = %s'  
cursor.execute(query, photoID)  
data = cursor.fetchone()  
query = 'DROP VIEW v'  
cursor.execute(query)

Tag the user with tagstatus of 0.

query = 'INSERT INTO tagged VALUES (%s, %s, 0)'  
cursor.execute(query, (follower, photoID))

**Locations:**

app.py (Lines 175-254)

tagPerson.html

**A screenshot of a computer

Description automatically generated**

**Feature:** Adding friendgroups

**Name:** Michael Kwan

**Queries:**

Determines if a friendgroup with the same groupOwner and groupName already exists.

check = 'SELECT \* FROM friendgroup WHERE groupOwner = %s and groupName = %s'

cursor.execute(check, (groupOwner, groupName))

Inserts the new friendgroup with the groupOwner and groupName into the friendgroup table

Then the groupOwner is added into the that friendgroup as a member in the belongto table

query = 'INSERT INTO friendgroup VALUES(%s, %s, %s)'  
cursor.execute(query, (groupOwner, groupName, description))  
query2 = 'INSERT INTO belongto VALUES(%s, %s, %s)'  
cursor.execute(query2, (groupOwner, groupOwner, groupName))

**Locations:**

app.py (Lines 374-399)

createFriendGroup.html

A screenshot of a computer

Description automatically generated

**Feature:** Add friend

**Name:** Michael Kwan

**Queries:**

Check if the friendgroup you want to add a friend to exists in the friendgroup table

check = 'SELECT \* FROM friendgroup WHERE groupOwner = %s and groupName = %s'  
cursor.execute(check, (owner\_username, groupName))

Check if the friend username exists in person table

check2 = 'SELECT username FROM person WHERE username = %s'  
cursor.execute(check2, (member\_username))

Check if the friend username is already a member of the friendgroup in the belongto table

check3 = 'SELECT \* FROM belongto WHERE owner\_username = %s and groupName = %s and member\_username = %s'  
cursor.execute(check3, (owner\_username, groupName, member\_username))

Add the friend into the friendgroup in the belongto table

query = 'INSERT INTO belongto VALUES(%s, %s, %s)'  
cursor.execute(query, (member\_username, owner\_username, groupName))

**Locations:**

app.py (Lines 401-439)

addToFriendGroup.html

**A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated**

**Feature:** Search by poster

**Name:** Nahom Molla

**Queries:**

Determine if the user is the photoPoster and if so, get all the info about that photo.

query = 'SELECT photoPoster, photoID, caption, postingDate FROM Photo WHERE photoPoster=%s ORDER BY postingDate DESC'  
cursor.execute(query, (user))

Determine if the user is following the poster.

query = 'SELECT \* \  
 FROM follow \  
 WHERE followstatus = 1 AND username\_followed = %s AND username\_follower = %s'  
cursor.execute(query, (poster, user))

Get all the photos that the photoposter posted and has allFollowers = 1.

query = 'SELECT photoPoster, photoID, caption, postingDate FROM Photo WHERE photoPoster=%s AND \  
 allFollowers=1 ORDER BY postingDate DESC'  
  
cursor.execute(query, (poster))

Determine if the user is in the friendgroup the photos are shared with.

query = 'SELECT \* \  
 FROM belongto AS B JOIN sharedwith AS S ON B.groupName = S.groupName AND B.owner\_username = S.groupowner\  
 WHERE member\_username = %s AND B.owner\_username = %s'  
cursor = conn.cursor()  
cursor.execute(query, (user, poster))

Get all the photos that the photoposter posted and has allFollowers = 0.

query = 'SELECT photoPoster, photoID, caption, postingDate FROM Photo WHERE photoPoster=%s \  
 AND allFollowers = 0 ORDER BY postingDate DESC'  
cursor.execute(query, (poster))

Get all the photos that the photoposter posted.

query = 'SELECT photoPoster, photoID, caption, postingDate FROM Photo WHERE photoPoster=%s \  
 ORDER BY postingDate DESC'  
cursor.execute(query, (poster))

**Locations:**

app.py (Lines 256-314)

searchByPoster.html

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated