Snigdha Kalathur and Yash Bengali CS460 PA1-Part2 Final Report

Final Schema

```
CREATE DATABASE IF NOT EXISTS photoshare;
USE photoshare;


CREATE TABLE Users (
    user_id int4  AUTO_INCREMENT,
    email varchar(255) NOT NULL UNIQUE,
    password varchar(255) NOT NULL,
    fName varchar(255),
    lName varchar(255),
    DOB DATE,
    hometown varchar(255),
    gender varchar(255),
  CONSTRAINT users_pk PRIMARY KEY (user_id)
);

CREATE TABLE Albums(
        albumID                 INTEGER AUTO_INCREMENT,
        albumName       CHAR(20) NOT NULL,
        albumDate       DATE,
        albumOwnedBy    int4,
        PRIMARY KEY (albumID),
        FOREIGN KEY (albumOwnedBy)
REFERENCES Users (user_id)
);
CREATE TABLE Pictures
(
  picture_id int4  AUTO_INCREMENT,
  user_id int4,
  imgdata longblob ,
  caption VARCHAR(255),
  belongs INTEGER NOT NULL,
  INDEX upid_idx (user_id),
  CONSTRAINT pictures_pk PRIMARY KEY (picture_id),
  FOREIGN KEY (belongs) REFERENCES Albums (albumID) ON DELETE CASCADE
```

```
);

CREATE TABLE FriendsWith (
        userID1         INTEGER,
        userID2         INTEGER,
        PRIMARY KEY (userID1, userID2),
        FOREIGN KEY (userID1) REFERENCES Users (user_id),
        FOREIGN KEY (userID2) REFERENCES Users (user_id),
        CONSTRAINT selfFriend CHECK (userID1 != userID2)
);


CREATE TABLE Likes(
        userID          INTEGER,
        photoID         INTEGER,
        PRIMARY KEY (userID, photoID),
        FOREIGN KEY (userID)
REFERENCES Users (user_id),
        FOREIGN KEY (photoID)
 REFERENCES Pictures (picture_id)
);

CREATE TABLE Tags(
        tagDescription          CHAR(20),
        PRIMARY KEY (tagDescription)
);

CREATE TABLE taggedWith(
        photoID         INTEGER,
        tagDescription CHAR(20),
        PRIMARY KEY (photoID, tagDescription),
        FOREIGN KEY (photoID)
                REFERENCES Pictures(picture_id),
        FOREIGN KEY (tagDescription)
                REFERENCES Tags(tagDescription)
);

CREATE TABLE Comments(
        commentID               INTEGER AUTO_INCREMENT,
        commentText             CHAR(100) NOT NULL,
```

```
    commentDate  DATE,
    commentOwnedBy     INTEGER,
    commentedUnder       INTEGER NOT NULL,
    PRIMARY KEY (commentID),
    FOREIGN KEY (commentOwnedBy)
    REFERENCES Users (user_id),
    FOREIGN KEY (commentedUnder)
    REFERENCES Pictures (picture_id)
);
```

Use Case assumptions, implementations, limitations, etc.

User Management

- Becoming a Registered User
    - All fields except hometown and gender are required.
- Adding and Listing Friends
    - This feature can be accessed by (only) logged in users via the Find and View Friends hyperlink on the home page. The search is done on the email. If the user submits the form with *text*, the query uses WHERE email LIKE %*text*% so that all users with emails that contain the specified text are returned. If no results are returned from the database, a message appears to the user saying that no users matched the search. The user can add friends from the list of users matching their search by selecting the Add Friend hyperlink next to each user's name. All of the user's friends are displayed under Current Friends.
- User Activity
    - Click on user activity to access the feature. Implemented using a search query where we found the number of photos and the number of users and added them together. Sorted in descending order.

Album and Photo Management

- Photo and Album Browsing
    - This can be accessed with the "View all Photos" and "View all Albums" for all users. Only logged in users can access "View your Photos" and "View your Albums." This was implemented using a search query where the view all photos and view all albums returned everything in the photo and album database. The view your photos and view your albums were made with a similar query that was filtered by the user_id.

- Photo and Album Creating
    - Upload button to upload a photo which must be put into an album. Tags are optional for photo uploading and must be separated by commas. Albums are displayed as a list and the user must pick 1 to upload under.

Tag Management

- Assumption: Tags are one word
- Viewing Your Photos by Tag Name
    - To see all user's tags, the user must select the View Tags hyperlink from the home page. The list of tags is displayed in no particular order. Selecting one of the hyperlinks displays all the user's photos associated with the selected tag. This feature is only viewable for logged in users.
- Viewing All Photos by Tag Name
    - To see all tags, the user must select the View Tags hyperlink from the home page. The list of tags is displayed in no particular order. Selecting one of the hyperlinks displays all the photos associated with the selected tag. This feature is viewable for logged in and non logged in users.
- Viewing the Most Popular Tags
    - To see popular tags, click on popular tags hyperlink. The returned tags can be clicked which will display all photos correlating with that tag. Looks through taggedWith data and groups by tags. It then sorts them by count and returns the top 5 most popular tags.
- Photo Search
    - Users can access this feature by selecting the Search Photos hyperlink from the home page. The submitted text is trimmed and split by whitespaces and a SELECT query is sent to the database to find photos containing all the specified tags. This feature is available for logged users and non logged in users.

Comments

- Leaving Comments
    - Users can click the Comment hyperlink below a photo to comment on the photo and to see all the photo's comments. Photo owners will not see the text field to submit a comment but can see the history of comments. Comments are preceded by the comment owner and the date the comment was created. The comment feature is available for non logged in users. For non logged in users, the query to the Comments table in the database omits a userID for the commentOwner attribute and their comments are displayed with the name Anonymous on the web page.
- Like Functionality

    - Users can like photos by clicking the Like hyperlink. Currently after liking a picture, the user is redirected back to the profile page. For future improvement, it would be more user friendly to not be directed to another page after liking a photo. Users can like their own photos. The query for inserting likes in the database uses REPLACE instead of INSERT so that no error is thrown when a user clicks the like button of the same photo more than once. Users must be logged in to like photos. Selecting the View Likes hyperlink below a photo displays the emails of all the users that liked the photo as well as the total number of likes the photo has.
- Search on Comments
    - Users can enter a text query and it will return all users who have made comments which exactly match that text along with how many times each user commented that specific text query. We joined the Users and Commented table and grouped by the comments text and user email. After sorting we then return the results in descending order.

Recommendations

- Friend Recommendation
    - This feature is accessible via the Friend Recommendations hyperlink. This feature is available only for logged in users. If the user has no friends, a message appears saying there are no friend recommendations. Otherwise, friends are recommended using the friends of friends approach.
- You-may-also-like functionality
    - This feature is accessible via the You May Also Like the hyperlink. This feature is only available for logged in users. If the user has no uploaded photos with tags, then a message appears saying that there are no recommendations at this time. Otherwise, the implementation is as specified in the assignment where all photos containing the user's 5 most popular tags are displayed in order from all 5 tages to 1 tag with ties broken by photos with the least number of total tags.