Photoshare Project Report
Camille Christie and Jeremy Bui

The final report should include the final schema, additional assumptions that you make during the implementation, and the limitations of your system (what functions are implemented and what are not). You need to submit to Gradescope the following:

Final Schema:

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

CREATE TABLE IF NOT EXISTS Users (
    user_id int4 AUTO_INCREMENT,
    email VARCHAR(255) UNIQUE,
    password varchar(255) NOT NULL,
   dob DATE,
    first_name CHAR(27),
    last_name CHAR(27),
    hometown VARCHAR(255),
    gender VARCHAR(255),
   CONSTRAINT users_pk PRIMARY KEY (user_id)
);


CREATE TABLE IF NOT EXISTS Albums(
    album_id int4 AUTO_INCREMENT,
    owner_id int4,
    album_name VARCHAR(255),
    date_of_creation DATE,
    CONSTRAINT album_pk PRIMARY KEY (album_id),
   FOREIGN KEY (owner_id) REFERENCES Users(user_id) ON DELETE CASCADE

);


CREATE TABLE IF NOT EXISTS Pictures
```

```
(
  picture_id int4  AUTO_INCREMENT,
  user_id int4,
  num_likes INT,
  album_id int4,
  imgdata longblob ,
  caption VARCHAR(255),
  INDEX upid_idx (user_id),
  CONSTRAINT pictures_pk PRIMARY KEY (picture_id),
  FOREIGN KEY (album_id) REFERENCES Albums(album_id) ON DELETE CASCADE,
  CONSTRAINT num_likes_check CHECK (num_likes >= 0)
);

CREATE TABLE IF NOT EXISTS Comments(
   comment_id int4 AUTO_INCREMENT,
    picture_id int4,
    user_id int4,
    text_comment VARCHAR(255),
    date_of_comment DATE,
   CONSTRAINT comment_pk PRIMARY KEY (comment_id),
   FOREIGN KEY (picture_id) REFERENCES Pictures(picture_id) ON DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS Tags(
    tag_id int4 AUTO_INCREMENT,
    tag_description VARCHAR(255),
    CONSTRAINT tags_pk PRIMARY KEY (tag_id)
);

CREATE TABLE IF NOT EXISTS Photo_contain(
    picture_id int4,
    tag_id int4,
    PRIMARY KEY (picture_id, tag_id),
    FOREIGN KEY (picture_id) REFERENCES Pictures(picture_id) ON DELETE CASCADE,
    FOREIGN KEY (tag_id) REFERENCES Tags(tag_id)
);

CREATE TABLE IF NOT EXISTS Liked_by(
    user_id int4,
    picture_id int4,
```

```
    PRIMARY KEY (user_id, picture_id),
    FOREIGN KEY (user_id) REFERENCES Users(user_id),
    FOREIGN KEY (picture_id) REFERENCES Pictures(picture_id) ON DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS Friends(
    super_user_id int4,
    sub_user_id int4,
    PRIMARY KEY (super_user_id, sub_user_id),
    CONSTRAINT friendship_super FOREIGN KEY (super_user_id) REFERENCES
Users(user_id),
    CONSTRAINT friendship_sub FOREIGN KEY (sub_user_id) REFERENCES Users(user_id)
);
```

Additional Assumptions to Schema:

- The comments table is not a relation to the users table. If that was implemented, then it would not be able to set the value of an anonymous user commenting to null. The TF listed on piazza that it is acceptable to set the values to null.
    - Another possible solution that we did not approach is setting a default userid in the users table to -1 to represent anonymous users.
- Users can not comment on their own photos
- The pictures table depends on the albums table, because a picture must be in exactly one album. ON DELETE CASCADE was added to the table to assure all photos of a deleted album are purged as well.
- Comments and likes only belong to one picture, ON DELETE CASCADE added to assure their records are deleted when the picture is.
- The password entered can not be NULL.

Additional Functions in Python:

The list of functions for the following project descriptions below are more high-level. Due to the nature of this report, it is inefficient to list all the functions that were created in our python code. For a more indepth description of functions, please refer to our app.py.

- Allow users to enter multiple tags by separating with comma: tags.split(‘,’) and utilize for loop
- Get all the picture recommendations in the sql order by how many tags the user shares: getPictureRecsInOrder()
- Handle searching for friends, check if that friend exists, if so then allow the user to be able to add a friend or view their profile: hello_friend_handler()
- Recommend friends based on how many times user is found in another user’s friend list: friends_of_friends()
- Allow a user to view, delete, or create their own albums: album_handler()
- Show the top 3 users that have contributed (comments + posts) the most to the photoshare application: activity()
- Allow all users to view specific pictures of a user, in the picture page ours contains the photo, name of poster, comments section, tag sections, and likes: picture(picture_id).
    - Allows user to like a photo, delete it (if they are the owner), click on tags or leave comments: picture_handler()
        - A user can not comment on their own photo: conditional statement to check owner_id of picture and compare to current.user_id
- Users (and anonymous) can search for other users that posted a certain comment: commentSearch()
- Users (and anonymous) can search for pictures that have a certain tag. The tag can be one or many: tagSearch_handler()

Limitations:

In accordance with the project description and the rubric, we were able to implement every function required. However, this does not mean that our project is flawless. Here are some additional functions that we can implement to improve our application.

- Although we check if an email is unique in our database, it does not check the internet to actually see if an email exists.
- Following the project guidelines, a photo can not exist without an album. In some cases, a user would not prefer this.
- Friends are automatically added without a request form.
- A user is not able to private their profile, which may arise some security issues .
- Due to limited HTML knowledge, some of the photos stretch.