Joseph Kwan
Justin Wong

Relational Schema

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

DROP TABLE IF EXISTS Friends CASCADE;
DROP TABLE IF EXISTS Tagged CASCADE;
DROP TABLE IF EXISTS Likes CASCADE;
DROP TABLE IF EXISTS Comments CASCADE;
DROP TABLE IF EXISTS Photos CASCADE;
DROP TABLE IF EXISTS Albums CASCADE;
DROP TABLE IF EXISTS Tags CASCADE;
DROP TABLE IF EXISTS Users CASCADE;


CREATE TABLE Users(
 user_id INTEGER AUTO_INCREMENT,
 first_name VARCHAR(100)NOT NULL,
 last_name VARCHAR(100) NOT NULL,
 email VARCHAR(100) UNIQUE NOT NULL,
 birth_date DATE NOT NULL,
 hometown VARCHAR(100),
 gender VARCHAR(100),
 password VARCHAR(100) NOT NULL,
 score INTEGER,
 PRIMARY KEY (user_id)
 );

 CREATE TABLE Friends(
 user_id1 INTEGER,
 user_id2 INTEGER,
 PRIMARY KEY (user_id1, user_id2),
 FOREIGN KEY (user_id1)
 REFERENCES Users(user_id),
 FOREIGN KEY (user_id2)
 REFERENCES Users(user_id),
 CONSTRAINT notFriendsWithSelf
        CHECK(user_id1 <> user_id2)
);

CREATE TABLE Albums(
 albums_id INTEGER AUTO_INCREMENT,
```

```
 name VARCHAR(100),
 date DATE,
 user_id INTEGER NOT NULL,
 PRIMARY KEY (albums_id),
 FOREIGN KEY (user_id)
 REFERENCES Users(user_id)
);

CREATE TABLE Tags(
 tag_id INTEGER AUTO_INCREMENT,
 name VARCHAR(100),
 quantity INTEGER,
 PRIMARY KEY (tag_id)
);

CREATE TABLE Photos(
 photo_id INTEGER AUTO_INCREMENT,
 caption VARCHAR(100),
 imgdata LONGTEXT,
 albums_id INTEGER NOT NULL,
user_id INTEGER NOT NULL,
 PRIMARY KEY (photo_id),
 FOREIGN KEY (albums_id) REFERENCES Albums (albums_id)
 ON DELETE CASCADE,
FOREIGN KEY (user_id) REFERENCES Users (user_id)
);

CREATE TABLE Tagged(
 photo_id INTEGER,
 tag_id INTEGER,
 PRIMARY KEY (photo_id, tag_id),
 FOREIGN KEY(photo_id)
 REFERENCES Photos (photo_id),
 FOREIGN KEY(tag_id)
 REFERENCES Tags (tag_id)
);

CREATE TABLE Comments(
 comment_id INTEGER AUTO_INCREMENT,
 user_id INTEGER,
 photo_id INTEGER NOT NULL,
 text VARCHAR (100),
 date DATE,
 PRIMARY KEY (comment_id),
```

```
 FOREIGN KEY (user_id)
 REFERENCES Users (user_id),
 FOREIGN KEY (photo_id)
 REFERENCES Photos (photo_id)
 -- CONSTRAINT noSelfComment
--         CHECK( ( (SELECT C.user_id
--                   FROM Comments C
--         WHERE C.user_id =
--         (SELECT P.user_id
--         FROM Photos P
--         WHERE C.photo_id = P.photo_id))))
);

CREATE TABLE Likes(
 photo_id INTEGER,
 user_id INTEGER,
 PRIMARY KEY (photo_id,user_id),
 FOREIGN KEY (photo_id)
 REFERENCES Photos (photo_id),
 FOREIGN KEY (user_id)
 REFERENCES Users (user_id)
);
```

Assumptions:

-When creating an account, the user fills out all fields.

- Tags can only be assigned when viewing your pictures from Album

- You cannot remove comments and likes after submitting.

- Like and comment functionality is only available when searching by Tag

- Friendships are one-directional. For example, Friend A can be friends with Friend B while Friend B is not friends with Friend A.

- No special characters (e.g !|?|#) are submitted onto the website.

- The number of tags on a photo is unlimited.

- You cannot like a photo unless you are logged in.

- Photo recs are made based off the most popular tags site-wide.

- All data fields have a maximum 100 character limit.