The final report with the code and the additional assumptions that you made and the limitations of your system. In particular, you should submit using Gradescope the following:

1. The final report that should contain the final schema and any additional assumptions and constraints you made.
2. A zip file with the photoshare directory (all your code).

Report (15 pts):
==========
The final report that should contain the final schema and any additional assumptions and constraints you made.
==========

Final Schema:

```
CREATE DATABASE IF NOT EXISTS photoshare;
USE photoshare;
DROP TABLE IF EXISTS Photos CASCADE;
DROP TABLE IF EXISTS Users CASCADE;

CREATE TABLE Users (
   user_id INTEGER NOT NULL AUTO_INCREMENT,
   fname VARCHAR(50) DEFAULT NULL,
   lname VARCHAR(50) DEFAULT NULL,
   DOB DATE DEFAULT NULL,
   gender VARCHAR(50) DEFAULT NULL,
   hometown VARCHAR(50) DEFAULT NULL,
   email VARCHAR(255) UNIQUE,
   password VARCHAR(255) NOT NULL,
   CONSTRAINT users_pk PRIMARY KEY (user_id)
);


CREATE TABLE Photos
(
 photo_id INTEGER NOT NULL AUTO_INCREMENT,
 user_id INTEGER DEFAULT NULL,
 imgdata longblob,
 caption VARCHAR(255) DEFAULT NULL,
 INDEX upicture_id_idx (user_id),
 CONSTRAINT Photos_pk PRIMARY KEY (photo_id)
);

CREATE TABLE Albums
```

```
(
   album_id INTEGER NOT NULL AUTO_INCREMENT PRIMARY KEY,
   title VARCHAR(75)  DEFAULT NULL,
   user_id INTEGER DEFAULT NULL, /*change to default null?*/
   date_created DATE DEFAULT(CURRENT_DATE),
   FOREIGN KEY(user_id) REFERENCES
       Users(user_id) ON DELETE CASCADE

);

CREATE TABLE Tags (
   tag_id INTEGER NOT NULL AUTO_INCREMENT PRIMARY KEY,
   tag VARCHAR(50) NOT NULL
);

CREATE TABLE is_tagged
(
       tag_id INTEGER NOT NULL, /*should we change ot tag_id? depeneds if we need
<string:tag> or <int:tag_id> constraint*/
       photo_id INTEGER NOT NULL,
       PRIMARY KEY(tag_id, photo_id),
       FOREIGN KEY(tag_id) REFERENCES
           Tags(tag_id),/*if change to tag_id, need to change this*/
       FOREIGN KEY(photo_id) REFERENCES
           Photos(photo_id) ON DELETE CASCADE
   );

CREATE TABLE liked_photos
(
   user_id INTEGER NOT NULL,
   liked_photo INTEGER NOT NULL,
   PRIMARY KEY(user_id, liked_photo),
   FOREIGN KEY(user_id) REFERENCES
       Users(user_id),
   FOREIGN KEY(liked_photo) REFERENCES
       Photos(photo_id)
);


CREATE TABLE are_friends
(
user_id INTEGER NOT NULL,
```

```
friend_id INTEGER NOT NULL,
PRIMARY KEY(user_id, friend_id),
FOREIGN KEY(user_id) REFERENCES
Users(user_id),
FOREIGN KEY(friend_id) REFERENCES
   Users(user_id),
CHECK (user_id <> friend_id)
);

CREATE TABLE Comments
(
   cid INTEGER NOT NULL AUTO_INCREMENT PRIMARY KEY, /*required*/
   text VARCHAR(255) NOT NULL, /*required*/
   commenter_id INTEGER DEFAULT NULL, /*not required, bc if anon user*/
   date_commented DATE DEFAULT(CURRENT_DATE),
   poster_id INTEGER DEFAULT NULL,
   photo_id INTEGER NOT NULL,
   CHECK (commenter_id <> poster_id),
   FOREIGN KEY(commenter_id) REFERENCES
       Users(user_id),
   FOREIGN KEY(poster_id) REFERENCES
       Users(user_id) ON DELETE CASCADE,
   FOREIGN KEY(photo_id) REFERENCES
       Photos(photo_id) ON DELETE CASCADE
);

INSERT INTO Users (email, password) VALUES ('test@bu.edu', 'test');
INSERT INTO Users (email, password) VALUES ('test1@bu.edu', 'test');
```

Additional assumptions and constraints:

We assume people know other user's emails, and thus search for them by their emails (bc they wouldn't know their user id, and there can be duplicates with names)

- Ideally, can do SELECT name by from Users where name = [user input from form], (select all people with matching names), and create hyperlinks to their profiles (that contain their albums) that the user can choose from

Merged photo and tag search: searching by tag allows conjunctive taq queries, so you can specify multiple words in the search by tag field and it will show you the photos that have all of the tags in query