CS460 PA1 Final Report
By Seungwon Burm and Yuchan Kim

Schema for PA1

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

--
-- Table structure for table `albums`
--
DROP TABLE IF EXISTS `albums`;
CREATE TABLE `albums` (
  `album_id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(40) NOT NULL,
  `user_id` int(11) NOT NULL,
  PRIMARY KEY (`album_id`),
  KEY `user_id` (`user_id`),
  CONSTRAINT `albums_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `users` (`user_id`) ON
DELETE CASCADE
)

--
-- Table structure for table `friends`
--
DROP TABLE IF EXISTS `friends`;
CREATE TABLE `friends` (
  `user_id_1` int(11) NOT NULL DEFAULT '0',
  `friend_id` int(11) NOT NULL DEFAULT '0',
  PRIMARY KEY (`user_id_1`,`friend_id`),
  KEY `friend_id` (`friend_id`),
  CONSTRAINT `friends_ibfk_1` FOREIGN KEY (`user_id_1`) REFERENCES `users` (`user_id`) ON
DELETE CASCADE,
  CONSTRAINT `friends_ibfk_2` FOREIGN KEY (`friend_id`) REFERENCES `users` (`user_id`) ON
DELETE CASCADE
)

--
-- Table structure for table `likes`
--
DROP TABLE IF EXISTS `likes`;
CREATE TABLE `likes` (
  `like_id` int(11) NOT NULL AUTO_INCREMENT,
  `picture_id` int(11) NOT NULL,
  `user_id` int(11) NOT NULL,
  PRIMARY KEY (`like_id`)
)
```

```
--
-- Table structure for table `pictures`
--
DROP TABLE IF EXISTS `pictures`;
CREATE TABLE `pictures` (
  `picture_id` int(11) NOT NULL AUTO_INCREMENT,
  `user_id` int(11) DEFAULT NULL,
  `imgdata` longblob NOT NULL,
  `caption` varchar(255) DEFAULT NULL,
  `album_id` int(11) DEFAULT NULL,
  PRIMARY KEY (`picture_id`),
  KEY `upid_idx` (`user_id`),
  KEY `album_id` (`album_id`),
  CONSTRAINT `pictures_ibfk_1` FOREIGN KEY (`album_id`) REFERENCES `albums` (`album_id`) ON
DELETE CASCADE,
  CONSTRAINT `pictures_ibfk_2` FOREIGN KEY (`user_id`) REFERENCES `users` (`user_id`) ON
DELETE CASCADE
)

--
-- Table structure for table `replys`
--
DROP TABLE IF EXISTS `replys`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `replys` (
  `reply_id` int(11) NOT NULL AUTO_INCREMENT,
  `picture_id` int(11) NOT NULL,
  `user_id` int(11) NOT NULL,
  `contents` text,
  PRIMARY KEY (`reply_id`)
)

--
-- Table structure for table `tagged`
--
DROP TABLE IF EXISTS `tagged`;
CREATE TABLE `tagged` (
  `picture_id` int(11) NOT NULL DEFAULT '0',
  `tag_id` int(11) NOT NULL DEFAULT '0',
  PRIMARY KEY (`picture_id`,`tag_id`),
  KEY `tag_id` (`tag_id`),
  CONSTRAINT `tagged_ibfk_1` FOREIGN KEY (`picture_id`) REFERENCES `pictures` (`picture_id`),
  CONSTRAINT `tagged_ibfk_2` FOREIGN KEY (`tag_id`) REFERENCES `tags` (`tags_id`)
)
```

```
--
-- Table structure for table `tags`
--
DROP TABLE IF EXISTS `tags`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `tags` (
  `tags_id` int(11) NOT NULL AUTO_INCREMENT,
  `tags_name` varchar(100) NOT NULL,
  PRIMARY KEY (`tags_id`)
)

--
-- Table structure for table `users`
--
DROP TABLE IF EXISTS `users`;
CREATE TABLE `users` (
  `user_id` int(11) NOT NULL AUTO_INCREMENT,
  `email` varchar(255) NOT NULL,
  `password` varchar(255) NOT NULL,
  `dob` date NOT NULL,
  `first_name` varchar(40) NOT NULL,
  `last_name` varchar(40) NOT NULL,
  `hometown` varchar(40) NOT NULL,
  `gender` varchar(6) NOT NULL,
  PRIMARY KEY (`user_id`),
  UNIQUE KEY `email` (`email`)
)
```

Assumptions

Previous Assumptions:

- One user can have multiple albums, an album must be owned by exactly one user
- Once user1 becomes friend with user2, user2 automatically becomes friend with user1
- One album can have multiple photos, a photograph can only belong to one album, not multiple albums.
- One user can write multiple comments, a comment must be written by exactly one user
- A photo can have multiple comments, a comment must be posted in exactly one photo

Additional Assumptions:

- Albums created by users (who are currently signed in) are under "Here are your albums"
- Albums created by users (who are currently signed in) plus others are under "Here are public albums"
- User can view photos by tags by searching tags
- User can view photos by comments by searching them
- User cannot leave comments on their own photos
- You-may-also-like function is known as 'Best Photos' that displaying popular photos associated with tags
- "Replys" serves as comments
- Contribution score is calculated by number of photos uploaded plus number of comments posted
- Friend Recommendation is done by introducing user A's common friends

Limitations

Functions Implemented:

Registration by providing their first name, last name, email address, date of birth, and a password.
Duplicate email check
Add Friends
Search Friends
Show friends
Calculate contribution score and display top 10 contributors
Public album browsing (public albums for non-registered users and other users)
Personal album browsing
Album creation by Users
Album/photo deletion created by users
Viewing photos by tag name
Viewing all photos by tag name
Leaving comments except own photos
Count number of likes
List users who liked photos
Search comments
Friends recommendation (friends-of-friends)
You-may-also-like (implemented as best photos)

Functions Not Implemented:

Viewing your photos by tag name