**THING TO DO:**

1. **1.SQL SCHEMA**
   1. (as Data section on assignment describes – maybe use the solution we have)
2. **MYSQL QUERIES** (not sure if all of them have to be implemented this ways) for the following Use Cases:
   1. **becoming a registered user (K)**
   2. **adding or listing friends (K)**
      1. **add a user to friend list**
      2. **search for other users in the system**
      3. **list his/her friends**
   3. **user activity** (list top 10 users based on their #photos+#

comments) (B)

* 1. **browse all albums+photos (**not sure if mysql is required for this) (B)
  2. **photo and album creating (**MySQL, but mostly html work here**)**
  3. **photo and album deleting (**for albums, on delete cascade “photos”)
  4. **a user can view his/her photos by tag names**
  5. **a user can view ALL photos by tag name**
  6. **view the most popular tags**
  7. **conjunctive search photos by tag** (e.g. “friends boston” should display all photos with tags friends AND boston)
  8. **registers and anonymous users can leave comments**
  9. **a user can not leave comments on his/her photo 🡪 CONSTRAINT**
  10. **like a photo (**mostly HTML work, just add to database)
  11. how many **likes** a photo has, users that liked this photo
  12. **comment query-list users based on how many times their comments match this query** (what about anonymous users?)
  13. **recommend possible NEW friends to a user (friends of friends in descending order)**
  14. **photo recommendation** (take 5 most used tags. Recommend photos based on # common tags.. all 5, 4 out of 5, 3 out of 5 a.s.o. Resolve ties by ordering photos with smaller # of tags first.

1. **HTML PAGES (maybe most difficult part?)**

Here we have to find 1. How many html pages we need. 2.functionality of this page

(ONLY IF TIME ALLOWS: CSS ENRICHMENT)