Description of the EER in Assignment1a

I would explain the structure of the EER by going through the scenarios which a user lands up while using a social website and also maps with the information mentioned in the description of the assignment.

User Information:

1. User Maintains a Profile:
   - The entity set **Users** maintains a set of users in the database with "*userid*" and "*email*" forming the *primary key*.
   - Users maintain their profile which has all the personal details and is depicted my **Profile** entity set which has "*pid*" as its *primary* key.
   - Each user has to maintain a single profile and a profile should be maintained by a single user, this is brought out using the "Maintained_By" relationship having **total participation** and **1-1 relationship** on both the sides.
     (Assumption: a user can have only one profile in the social networking site)
   - Each profile has a "home_town " which is maintained as a multi-valued attribute since it has to take values for "state", "city", and "country", and the current address is maintained as a relationship "Has_Current_Address_At" with **Location** entity set.
   - A "Location" entity set, with "zip_code" as the primary key, has all the details of an address which could be used by many other entity sets to define the location.
   - A profile has an "**Access Privacy**" which is represented as a weak entity set as it has no primary key and the attributes like close_friends, relatives and others could be accessed only in conjunction with a profile.
   - A user can access many profile, and a profile could be accessed by many users which is represented using an "Accesses_By" relationship as ***many-many***.
   - The **Work_Education** entity set is a super class which maintains the start dates, end dates and the names of the work places or education places.
     Each user profile can have many work and/or education places and those places can be selected by many users, this is expressed by a ***many-many*** relationship between "Profile" and "Work_Education" entity sets.
     - The entity sets **Work** and **Education** are the subclasses of the Work_Education class which inherit the parent attributes and also define their specific achievement levels as wlevel and elevel respectively .
     - Work and Education names are selected from the page type **Places** having category as work or education. This is depicted using an aggregation of Work and education being related to "Places" Entity set.

2. User makes friends:
    - Each user can make friends by sending a request and also by accepting the requests arrived. This is represented by using a relationship "Makes_Friendship" between entities in the same entity set "Users".
    - Users entities take **roles (requestor or acceptor)** based on whether he sends a request or accepts a request.
    - This relationship is a weak relationship , and fid (friendship id) is a partial descriptive attribute of the relationship.
    - The group of friends like relatives, close friends are maintained as descriptive attributes of the relationship.

<u>Post Information:</u>

1. User creates Post:
    - User entity set is related to a **Post** entity set which maintains a set of attributes having pcode and sender as the primary key by "Publishes" relationship. The same relationship is used by User to receive posts based on the receiver attribute set by the sender.
    - Each post is published by a sender, but a sender can publish many posts, and each post should have a sender and receiver who could be the same as the sender, this is depicted by using a **one - many** relationship between "Post" and "User" entity sets, and also by having "Post" **totally participating** in the "Publishes" relationship.
    - Users can also access the posts posted by a sender based on the privacy level which is maintained as an attribute of each post. This is depicted by a many - many relationship, "Accesses_A" between "User" and "Post" entity sets.
2. User Comments and likes Posts:
    - Each user might comment on a post which has a text area with the details of the person who is commenting on it. Each post must have an owner where as an user can have many posts.
    - This is picturised using the "Posts_A" relationship being a **one - many** between user and "Comments", and "Comments" is described as a relationship itself as it doesn't have to maintain a separate Entity Set in order to showcase the comment details.
    - "Posts_A" has a time and date being captured as descriptive attributes for each comment being posted by the user.
    - Posts and Comments both maintain "likes" attribute to count for the number of likes for both posts and comments of each user.

Photo Information:

1. User Posts Photos:
    - Photos are to be owned by a user and it has to be contained in a post, this is described by using a "piccode" and "owner" primary attributes for the "**Photos**" entity set which is being related to "Posts"
    - Users can share photos only as posts, each post may contain many photos but each photo must be contained in a post with an owner.
      This is brought out by maintaining **one - many** relationship, "**Contains**" between "Posts" and "Photos", and also "Photos" having a **total participation** in the relationship.
    - Photos can have users being tagged in them, this is depicted using a "Tagged_With" relationship with "Users" Entity set. This also maintains a list of users tagged as a descriptive attribute for each picture.
2. User Specifies Photo Location:
    - The location details of the photo, along with the date and time of capture of the photo could be accommodated in the "Photos: entity set by having a "Has_A" relationship with "Location" entity set.
    - Each photo may have a location where as a location might be specified in many photos, this is the reason to use a **one - many** relationship between "Photos" and "Location".

Pages Information:

1. User can create and own a page:
    - An user can create and be an admin of a page having a unique title and a description.
      It is represented by having a "**Pages**" entity set being related to "Users" by "Admins_A" relationship.
    - An user can have many pages where as a page should be owned by a single admin and an owner, and each page should be having an admin, this is depicted by using a **one - many** and **total participation** between "Users" and "Pages".
    - A page can a Movie, Place or an Event, the concept of subclasses is used and having "Pages" as a superclass, three subclasses **Movies, Entities and Places** are created using an **ISA** relationship. The subclasses are going to inherit the attributes of the superclass.

2. User adds a "Movies" page to his watched list:
   - "Movies" subclass has all the attributes to describe a movie page, and it is being referenced by another entity set **Watched_Movies** which is being maintained for each user as a list of watched movies and rating of each one of them.
   - Each "Watched_Movies" belongs to "Movies" pages where the vice-versa is not true. this is being highlighted using a **total participation** between "Watched_Movies" and "Movie" along with a **one - one** mapping. The "rating" attribute maintains the movie rating for each watched movie.
3. User Handles an Event:
   - "Events" is a subclass of "Pages" having attributes to describe the details of an event, a **relationship**, "Maintains_A_List_Of" is used to maintain a list of "userids", and their "status", i.e., if they are ready, may attend, not decided or not attending the event.
4. User might access a page:
   - We have already considered "Movies" and "Event" pages, "Places" are another type of pages that user might access, check-in, and based on the category of the page.
   - The "Accesses_A" relationship allows to represent that users can access pages, its a **many - many** relationship as many users can access many pages.
   - "Places" as a subclass of "Pages" has a location defined for it and it is again a **one - many** relationship between "Places" and "Location" entity sets.
5. User might check-in a location or a place:
   - Users can check-in "Places " which anyway has a location along with a time and date being the descriptive attributes. A place can be checked-in by many users and a user can check in many places, this is represented using a **many - many** relationship between "Users" and "Places" through a "Check_In" relationship.
   - "Places" can have many categories like Education, Airport, Work, etc., and a multi-valued attribute "category" is being used to represent the variety of categories, and an user can check-in the "Places" as a location, or as a "Work" or as an "Education" based on the category.