(a) Describe some basic assumptions that you will make in your design. Describe any extra features you plan to add to the description, and any things you are planning to not support because they seem too complicated or useless (or you ran out of time). Discuss why you made these decisions.

PicBook is a social media website that allows its users to share interesting photos which can be something they found interesting while browsing the web or something that they clicked from their smartphone.

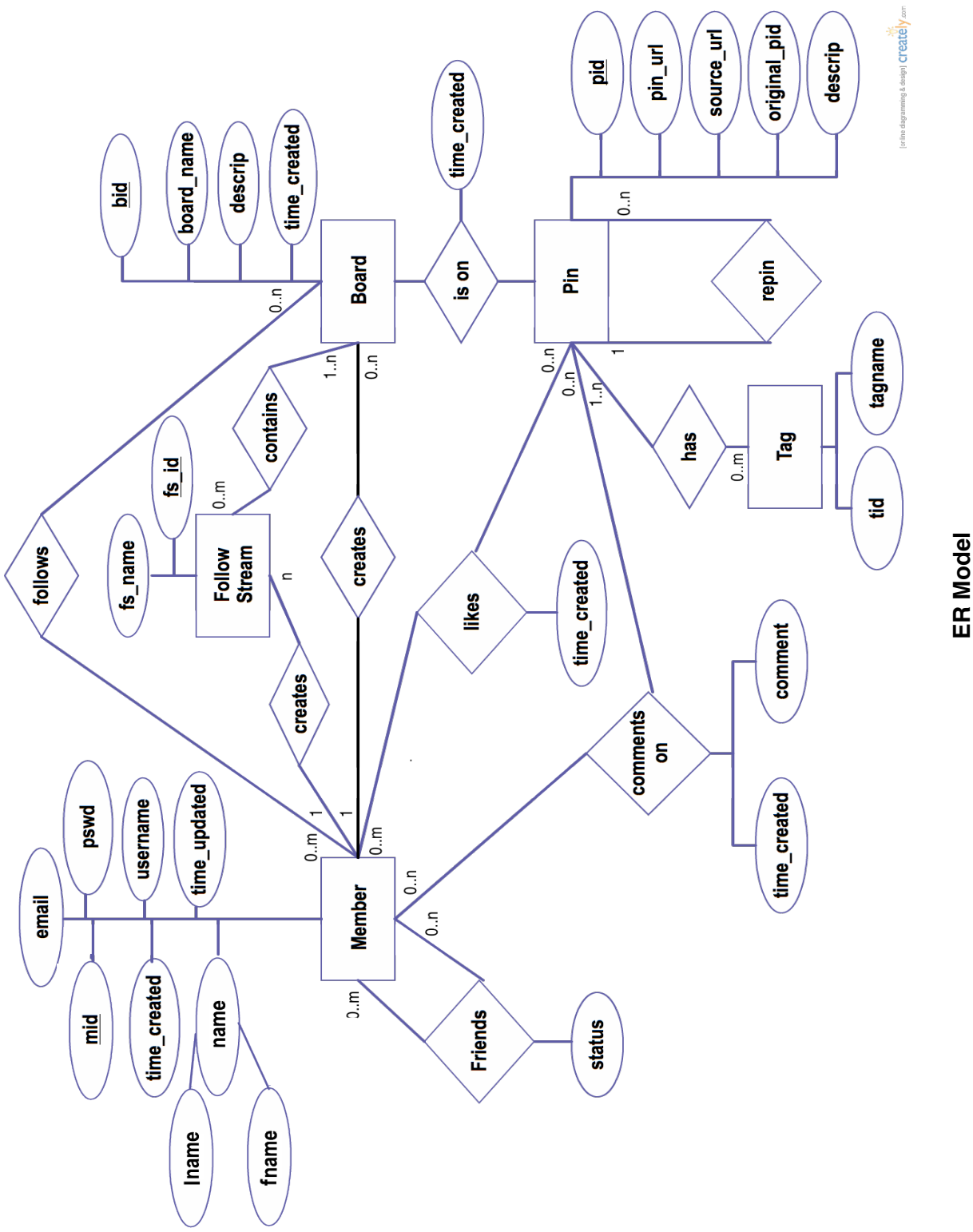
For this project we chose MS SQL Server as the backend because we plan to use ASP.NET as the frontend and it plays well with SQL Server.

The basic assumptions for the Project:

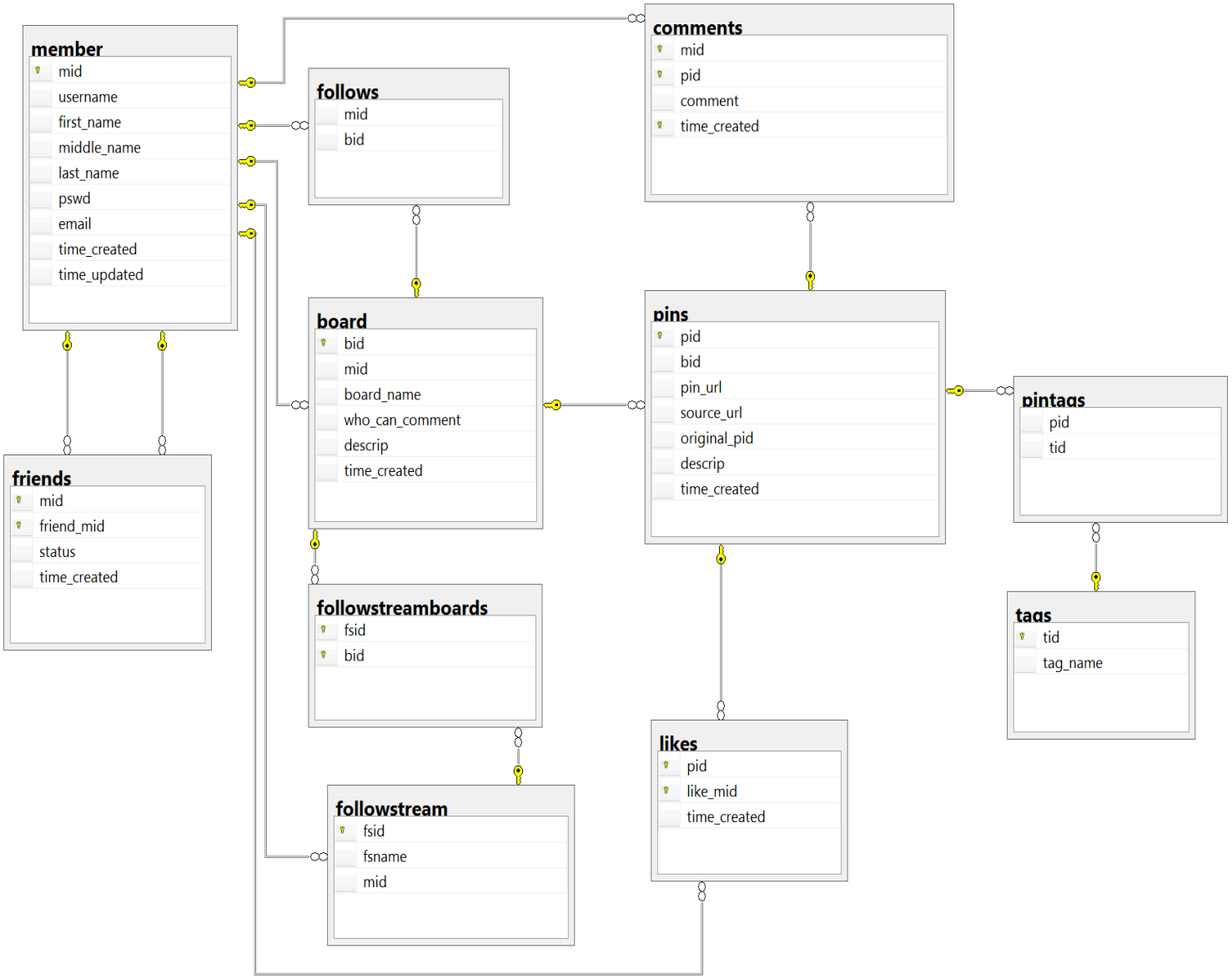
- Users signup as members for the website by giving some basic details (like: Name and Email) and get to choose a Username and Password to access their account in the future.
- Once logged in a user can:
    - View pins from all the other users,
    - Like them,
    - Comment on them (only if the owner user permits it. Details discussed later),
    - Create pins (either by copying a link from the web or uploading a picture from their PC),
    - Repin pins (copying pins created by some other user),
    - Delete pins (this includes both original pins and repins)
    - Create boards,
    - Decide who can comment on their boards (Only them, only friends or everyone. This is a setting they decide for each board separately.),
    - Follow boards (so that it is displayed in their default stream),
    - Create custom followstreams (which is a page which displays results from other's boards which have been followed and grouped by the user as followstream),
    - Send other users friend requests,
    - Approve friend requests which others have sent to them,
    - Decline friend requests which others have sent to them,
    - Tag pins
- When an original pin is deleted all the repins that originated from that pin get deleted.
- When a Repin is deleted only that Repin is deleted.
- When a pin is liked the like is attributed to the original pin.
- When a pin (Original pin or Repin) is commented on, the comment is attributed to that pin only.

- Permission for commenting on a board is by default “Everyone”. It can be changed to “Friends only” or “Me only” by the user later on.
- ***Pin Images are not stored in the Database as BLOBs.***
- ***When an image is uploaded by the user for creating a new pin the image is stored on the file server and a url is generated to refer to the image.***
- ***When the user gives a pin url of a image on the web the image is again copied to the file server and a url is generated to refer to the image. This url is stored in the pin_url column of the Pins table. We always have the source_url field which points to the original webpage from where the pin image was copied.***
- A followstream can contain only those boards which have been followed by the user separately.
- When a user deletes a followstream, only the followstream gets deleted and he/she keeps on following the individual boards that were the part of that followstream.

(b) Design, justify, and create an appropriate database schema for the above situation. Make sure your schema is space efficient and suitably normalized. Show an ER diagram of your design, and a translation into relational format. Identify keys and foreign key constraints. Note that you may have to revisit your design if it turns out later that the design is not suitable. Discuss in particular how you model users, boards, pictures, pins, comments, likes, repins etc.



NOTE: Primary keys and Foreign keys are denoted in the diagram.



Relational Database Schema

(c) Use a relational database system to create the schema, together with key, foreign key, and other constraints.

Member Table
The member table consists of the following fields:

mid	This field stores the member ID of the members. It is generated by Auto-Increment and the user has no control over what ID is assigned to him. This field is also the primary key for member table.
username	Username of users choice. Right now we are allowing users to have identical usernames. This might change if we feel we need unique username for each user.
first_name	First name of the member.
middle_name	Middle name of the member.
last_name	Last name of the member.
pswd	Stores the password set by user while creating account.
email	Email address of the member. A member uses his email address to log into the website and therefore the email address has to be unique for each member.
time_created	Timestamp for account creation.
time_updated	Timestamp for account updation.

```
create table member
(
  mid int NOT NULL identity(1,1),
  username varchar(25) NOT NULL DEFAULT '',
  first_name varchar(50) NOT NULL DEFAULT '',
  middle_name varchar(50) NOT NULL DEFAULT '',
  last_name varchar(50) NOT NULL DEFAULT '',
  pswd varchar(max) NOT NULL,
  email varchar(30) NOT NULL,
  time_created datetime2 NOT NULL DEFAULT CURRENT_TIMESTAMP,
  time_updated datetime2(7) NOT NULL DEFAULT CURRENT_TIMESTAMP,
  constraint pk_member primary key(mid),
  constraint uq_member unique(email)
)
```

Board Table
The board table consists of the following fields:

bid	This field stores the board ID for every board created. It is generated by Auto-Increment and the user has no control over what ID is assigned to a board. This field is also the primary key for board table.

mid	This field tells us which member created this board. This field is a foreign key which refers to the mid field of the Member table. Also this field has a on delete cascade property which essentially means that whenever an member row in the member table (parent table) is deleted, all the boards related to that mid are are deleted from the board table (child table).
board_name	Name of the board. Different boards can have the same name but one member can only have one board of a particular name.
who_can_comment	This field is used as the permission tracker for each board. If this field says ‘e’, then anyone who is a member of the website can comment on the pins on this board. If it says ‘f’, then only the friends of the member who owns this board may comment on the pins on this board. If it says ‘m’, then only the owner may comment on the pins on this board. The default setting is ‘e’ which means any one can comment on the pins on a board by default.
descrip	A few lines to describe what the board is about.
time_created	Timestamp for board creation.

```
create table board
(
  bid int NOT NULL identity(1,1),
  mid int NOT NULL,
  board_name varchar(25) NOT NULL,
  who_can_comment varchar(1) NOT NULL DEFAULT 'e',
  descrip text NOT NULL,
  time_created datetime2 NOT NULL DEFAULT CURRENT_TIMESTAMP,
  constraint pk_board primary key(bid),
  constraint fk_board foreign key(mid) references member(mid) ON DELETE CASCADE,
  constraint uq_board unique(mid, board_name),
  constraint ck_who_can_comment check (who_can_comment in ('f','e','m'))
)
```

Pins Table
The pins table consists of the following fields:

pid	This field stores the pin ID for every pin created. It is generated by Auto-Increment and the user has no control over what ID is assigned to a pin. This field is also the primary key for pin table. Every time a pin is repinned a new entry is made to the pins table and a new pid is assigned.
bid	This field tells us which which board does this pin belong to. This field is a foreign key which refers to bid in the board table. Owner of the pin is found by looking up the owner of the board it is pinned on. Also this field has a on delete cascade property which essentially means that whenever a board row in the board table (parent table) is deleted, all the pins related to that bid are are deleted from the board table (child table).
pin_url	URL containing only the image of the pin.
source_url	URL containing the webpage from where the pin was captured.

original_pid	This field tells us if a pin is original or a repin of some other pin. **If the original_pid field is the same as the pid field, this means that the pin is original. If they don't match then the original_pid refers to the original pin and this pin is a repin.**
descrip	A short description of the pin.
time_created	Timestamp for pin creation.

```
create table pins
(
pid int NOT NULL identity(1,1),
 bid int NOT NULL,
 pin_url text NOT NULL,
 source_url text,
 original_pid int NOT NULL DEFAULT Ident_current('pins'),
 descrip text NOT NULL,
 time_created datetime2 NOT NULL DEFAULT CURRENT_TIMESTAMP,
 constraint pk_pins primary key(pid),
 constraint fk_p_bid foreign key(bid) references board(bid) ON DELETE CASCADE
);
```

Likes Table
The likes table consists of the following fields:

pid	This field refers to the pin which is being liked. This field is a foreign key which refers to the pid in the pins table. This field has a on delete cascade property which essentially means that whenever a pin row in the pin table (parent table) is deleted, all the likes related to that pin are are deleted from the board table (child table).
like_mid	This field refers to the member who is liking the field. This field is a foreign key which refers to the mid in the member table.
time_created	Timestamp of when the like was made.

pid, like_mid fields together form the primary key.

```
create table likes
(
 pid int NOT NULL,
 like_mid int NOT NULL,
 time_created datetime2 NOT NULL DEFAULT CURRENT_TIMESTAMP,
 constraint fk_l_pid foreign key(pid) references pins(pid)  ON DELETE CASCADE,
 constraint fk_l_lm foreign key(like_mid) references member(mid),
 constraint pk_like primary key(pid, like_mid)
 );
```

Comments Table
The comments table consists of the following fields:

mid	The mid of the member who commented on the pin. This field is a foreign key which refers to the mid in the member table.
pid	The pid of the pin being commented on. This field is a foreign key which refers to the pid in the pins table. This field has a on delete cascade property which essentially means that whenever a pin row in the pin table (parent table) is deleted, all the comments related to that pin are are deleted from the board table (child table).
comment	The actual comment text.
time_created	Timestamp for comment creation.

mid, pid, time_created fields together form the primary key.

```
 create table comments
(
 mid int NOT NULL,
 pid int NOT NULL,
 comment text NOT NULL,
 time_created datetime2 NOT NULL DEFAULT CURRENT_TIMESTAMP,
 constraint pk_comments primary key(mid, pid, time_created),
 constraint fk_c_mid foreign key(mid) references member(mid),
 constraint fk_c_pid foreign key(pid) references pins(pid) ON DELETE CASCADE
);
```

Friends Table
The friends table consists of the following fields:

mid	Member who receives the friend request.
friend_mid	Member who sends the friend request.
status	Status of the friend request. A friend request can either be pending (‘p’) or approved (‘y’). When a member declines the friend request the record is deleted from the table so that they can send the request again in the future.
time_created	Timestamp

mid, friend_mid fields together form the primary key.

```
create table friends
(
 mid int NOT NULL,
 friend_mid int NOT NULL,
 status varchar(1) NOT NULL,
 time_created datetime2 NOT NULL DEFAULT CURRENT_TIMESTAMP,
```

```
  constraint pk_friend primary key(mid, friend_mid),
  constraint fk_fr_mid foreign key(mid) references member(mid) ON DELETE CASCADE,
  constraint fk_fr_friend_mid foreign key(friend_mid) references member(mid),
  constraint ck_status check (status in ('y','p'))
);
```

Followstream Table
The followstream table consists of the following fields:

fsid	This field stores the Followstream ID for every followstream created by every user. It is generated by Auto-Increment and the user has no control over what ID is assigned to a followstream. This field is also the primary key for Followstream table.
fsname	Name of the followstream.
mid	Member ID of the member who created this followstream. This field is a foreign key which refers to the mid field of the Member table. This field has a on delete cascade property which essentially means that whenever a member row in the member table (parent table) is deleted, all the followstream created by that member are deleted from the followstream table (child table).

```
create table followstream
(
  fsid int NOT NULL identity(1, 1),
  fsname varchar(50) NOT NULL,
  mid int NOT NULL,
  constraint pk_follow primary key(fsid),
  constraint fk_f_mid foreign key(mid) references member(mid) ON DELETE CASCADE
);
```

Followstreamboards table
The followstreamboards table consists of the following fields:

fsid	Followstream ID. This field is a foreign key that refers to the fsid in the followstream table. This field has a on delete cascade property which essentially means that whenever a followstream row in the followstream table (parent table) is deleted, all the followstreamboards contained in that followstream are deleted from the followstreamboards table (child table).
bid	Board ID of the boards associated with that fsid. This field is a foreign key which refers to bid in the board table.

fsid, bid fields together form the primary key.

```
create table followstreamboards
(
fsid int NOT NULL,
bid int NOT NULL,
constraint fk_fol_fsid foreign key(fsid) references followstream(fsid) ON DELETE CASCADE,
constraint fk_fol_bid foreign key(bid) references board(bid),
constraint pk_follows primary key(fsid, bid)
```

);

Follows table

The follows table consists of the following fields:

mid	Member ID who is following the board. This field is a foreign key which refers to the mid field of the Member table. This field has a on delete cascade property which essentially means that whenever a member row in the member table (parent table) is deleted, all the follow entries in the follows table associated with that member are deleted from the follows table (child table).
bid	Board ID of the board that is being followed. This field is a foreign key which refers to the bid field of the Board table.

mid, bid fields together form the primary key.

```
create table follows
(
mid int not NULL,
bid int not NULL,
constraint fk_folwg_mid foreign key(mid) references member(mid) ON DELETE CASCADE,
constraint fk_folwg_bid foreign key(bid) references board(bid),
constraint pk_folwg primary key(mid, bid)
)
```

Tags Table

The tags table consists of the following fields:

tid	This field stores the tag ID for every tag created. It is generated by Auto-Increment and the user has no control over what ID is assigned to a tag. This field is also the primary key for tags table. Every time a tag is created a new entry is made to the tags table and a new tid is assigned. If a user goes on to tag a picture with a tag that already exists the system makes sure that the existing tag is referred to and a new tag is not created.
tag_name	Name of the tag. Tag names have to be unique.

```
create table tags
(
tid int identity(1,1),
tag_name varchar(50) NOT NULL UNIQUE,
constraint pk_tags primary key(tid)
);
```

Pintags Table

The Pintags table consists of the following fields:

pid	Pin ID of the pin that is being tagged. This field is a foreign key which refers to the pid field of the Pins table.

tid	Tag ID. This field is a foreign key which refers to the tid field of the Tags table.

pid, tid fields have a unique constraint that means a particular pin can be tagged by a particular tag only once.

```
create table pintags
(
pid int NOT NULL,
tid int NOT NULL,
constraint fk_pt_pid foreign key(pid) references pins(pid),
constraint fk_pt_tid foreign key(tid) references tags(tid)
constraint uq_pt_pidtid unique(pid, tid)
);
```

(d) Write SQL queries (or sequences of SQL queries) for the following tasks.

We chose to embed each query into procedures so that it is easy to call them once we develop the frontend.

1) Signing Up, Creating Boards, and Pinning: Write queries that users need to sign up, to login, to create or edit their profile, to create pinboards, to pin a picture, and to delete a pinned picture.

Signing Up

```
CREATE PROCEDURE CreateAccount(
        @username varchar(25),
        @first_name varchar(50),
        @middle_name varchar(50),
        @last_name varchar(50),
        @pswd varchar(max),
        @email varchar(30))
AS
BEGIN
        insert into member(username,first_name,middle_name,last_name,pswd,email)
        values (@username,@first_name,@middle_name,@last_name,@pswd,@email);
END
```

Login

```
CREATE PROCEDURE CheckPassword(
   @username VARCHAR(30),
   @password varchar(max))
AS
BEGIN
        IF EXISTS(SELECT * FROM member WHERE email = @username AND pswd =
@password)
                SELECT 'true' AS UserExists
        ELSE
                SELECT 'false' AS UserExists
```

```
END
```

Update Profile

```
create procedure UpdateProfile(
        @firstname varchar(50),
        @middle_name varchar(50),
        @last_name varchar(50),
        @username varchar(30))
as
Begin
        update member
        set first_name = @firstname, middle_name = @middle_name, last_name = @last_name
        where email = @username;
end
```

Create Pinboards

```
create procedure CreateBoard(
        @board_name varchar(25),
        @discrip text,
        @mid int)
as
begin
        insert into board(mid, board_name, descrip)
        values(@mid, @board_name, @discrip);
end
```

Create Pins

```
create procedure PinPicture(
        @bid int,
        @pin_url text,
        @source_url text,
        @descrip text)
as
begin
        insert into pins(bid, pin_url, source_url, descrip)
        values(@bid, @pin_url, @source_url, @descrip);
end
```

Create Repins

```
create procedure RepinPicture(
        @bid int,
        @original_pid int,
        @descrip text)
as
begin
        declare @pin_url varchar(max);
```

```
        declare @source_url varchar(max);
        select @pin_url = pin_url, @source_url = source_url from pins where pid =
@original_pid;
        insert into pins(bid, pin_url, source_url, original_pid, descrip)
        values(@bid, @pin_url, @source_url, @original_pid, @descrip);
end
```

Delete pins and repins

```
create procedure DeletePicture(
        @pid int)
as
begin
        declare @original int;
        set @original = (select pid from pins where original_pid = @pid and pid = @pid);
        if(@original is not NULL)
                begin
                        delete from pins
                        where original_pid = @pid;
                end
        else
                begin
                        delete from pins
                        where pid = @pid;
                end
end
```

2) Friends: Write queries for asking another user to be friends, and for answering a friend requests.

Sending Friend Request

```
create procedure SendFriendRequest(
        @mid int,
        @friend_mid int,
        @status varchar(1))
as
begin
        insert into friends(mid,friend_mid,status)
        values(@mid,@friend_mid,@status);
end
```

Updating(Approving/Declining) Friend Request

```
create procedure UpdateFriendRequest(
        @mid int,
        @friend_mid int,
        @status varchar(1))
as
```

```
begin
	if(@status='y')
		begin
			update friends
			set status='y' where mid=@mid and friend_mid=@friend_mid;
		end
	else
		begin
			delete from friends where mid=@mid and friend_mid=@friend_mid
		end

end
```

3) Repinning and Following: Write queries for repinning a picture and for creating a follow stream. Also, write a query that given a follow stream, displays all pictures belonging to that follow stream in reverse chronological order.

Repinning Pin

```
create procedure RepinPicture(
	@bid int,
	@original_pid int,
	@descrip text)
as
begin
	declare @pin_url varchar(max);
	declare @source_url varchar(max);
	select @pin_url = pin_url, @source_url = source_url from pins where pid =
@original_pid;
	insert into pins(bid, pin_url, source_url, original_pid, descrip)
	values(@bid, @pin_url, @source_url, @original_pid, @descrip);
end
```

Creating a Followstream

```
create procedure CreateFollowStream(
	@mid int,
	@fsname varchar(50)
)
as
begin
	insert into followstream(fsname,mid)
	values(@fsname,@mid);
end
```

Adding to Followstream

```
create procedure AddToFollowStream(
	@fs_id int,
```

```
        @bid int
)
as
begin
        insert into followstreamboards(fsid,bid)
        values(@fs_id,@bid);
end
```

Display pins belonging to follow stream in reverse chronological order

```
select pin_url from pins
where bid in (select bid from followstreamboards
                where fsid in(select fsid from followstream
                                where mid=@mid))
order by time_created desc
```

4) Liking and Commenting: Write queries to like a picture, and to add a comment to a picture (while making sure the user is allowed to comment on this picture).

Liking a pin

```
create procedure LikePin(
        @pid int,
        @like_mid int)
as
begin
        insert into likes(pid,like_mid)
        values(@pid,@like_mid);
end
```

Commenting on a pin

```
create procedure CommentPin(
        @mid int,
        @pid int,
        @comment text)
as
begin
        declare @bid int;
        set @bid = (select bid from pins where pid=@pid);
        declare @permission varchar(1)
        set @permission = [dbo].GetPermission(@bid);
        if(@permission = 'f')
                begin
                        declare @friend_mid int;
                        set @friend_mid = (select mid from board where bid = @bid);
                        declare @isFriend varchar(1);
                        set @isFriend = [dbo].IsFriend(@mid, @friend_mid);
```

```
                        if(@isFriend ='y')
                                begin
                                        insert into comments(mid,pid,comment)
                                        values(@mid,@pid,@comment);
                                end
                        else
                                return;
                end
        else if(@permission = 'e')
                begin
                        insert into comments(mid,pid,comment)
                        values(@mid,@pid,@comment);
                end
        else
                begin
                        if( @mid = (select mid from board where bid = @bid))
                                insert into comments(mid,pid,comment)
                                values(@mid,@pid,@comment);
                        else
                                return;
                end
end
```

GetPermission() returns the permission set for the board on which the pin is pinned.

```
create function GetPermission(
        @bid int
)
returns varchar(1)
as
begin
        declare @permission varchar(1);
        select @permission = who_can_comment from board where bid = @bid;
        return @permission;
end
```

IsFriend(mid1, mid2) returns ‘y’ or ’n’ depending if two mid1 and mid2 are friends or not respectively.

```
create function IsFriend(
        @mid int,
        @friend_mid int)
returns varchar(1)
as
begin
        declare @isfriend int;
        declare @retval varchar(1);
        select @isfriend = mid from friends where ((mid=@mid and friend_mid=@friend_mid) or
(mid=@friend_mid and friend_mid=@mid)) and status = 'y';
```

```
        if(@isfriend is not NULL)
                begin
                        set @retval = 'y';
                end
        else
                begin
                        set @retval = 'n';
                end
                return @retval;
end
```

5) Keyword Search: Write a query to perform a keyword search for pictures whose tags match the keywords. Use the contain operator to do so.

```
declare @tagname varchar(50);
set @tagname='nice';
select pid,pin_url from pins
where pid in(select pid from pintags
                where tid in (select tid from tags
                                where contains(tag_name,@tagname)))
```

e) Populate your database with some sample data, and test the queries you have written in part (d). Make sure to input interesting and meaningful data and to test a number of cases. Limit yourself to a few users and a few pictures and boards, but make sure there is enough data to generate interesting test cases. It is suggested that you design your test data very carefully. Draw and submit a page with the content of all the tables in easily understandable form (not a long list of insert statements) and discuss the structure of the data.

Member table - Member entries for 5 members

mid	username	first_name	middle_name	last_name	pswd	email	time_created	time_updated
1	wall	Anshul		Mehra	123456	ansh.mehra@me.com	2013-11-25 13:59:51.2600000	2013-11-25 13:59:51.2600000
2	sru	Srujan		Saggam	654321	srujan.saggam1@gmail.com	2013-11-25 13:59:51.2600000	2013-11-25 13:59:51.2600000
3	Therock	Pratik		Patel	123123	pratik.patel@me.com	2013-11-25 13:59:51.2630000	2013-11-25 13:59:51.2630000
4	chotu	Komal		Ruparel	321321	komal.ruparel@nyu.edu	2013-11-25 13:59:51.2630000	2013-11-25 13:59:51.2630000
5	Don	Ashish		Parekh	222222	ashish@nyu.edu	2013-11-25 13:59:51.2630000	2013-11-25 13:59:51.2630000

Board table -

- bid 1,2 and 3 belong to mid 1 (Anshul Mehra)
- bid 4,5 belongs to mid 2 (Srujan Saggam)
- bid 6 belongs to mid 3 (Pratik Patel)
- bid 1 has ‘f’ in who_can_comment column. This means only friends of the owner (Anshul Mehra) can comment on the pins on this board.
- bid 4 has ‘m’ in who_can_comment column. This means only the owner of that board (Srujan Saggam) can comment on the pins on this board.
- All other pins have ‘e’ in the who can comment column. This means that anybody can comment on the pins on this board.

bid	mid	board_name	who_can_comment	descrip	time_created
1	1	Cars	f	Fun car photos	2013-11-25 14:01:24.7100000
2	1	Bikes	e	Fun bike photos	2013-11-25 14:01:24.7100000
3	1	Places to visit	e	I want to go here	2013-11-25 14:01:24.7100000
4	2	Cars & Bike	m	Speed vehicles	2013-11-25 14:01:24.7130000
5	2	Trolls	e	Funny Animated GIFs	2013-11-25 14:01:24.7130000
6	3	Monsters	e	Monster photos	2013-11-25 14:01:24.7130000
7	4	Babies	e	Baby photos	2013-11-25 14:01:24.7130000
8	5	Babies	e	New Born	2013-11-25 14:01:24.7130000

Pins Table

with original pins

- Notice the pid and original_pid is same in case of original pins.

pid	bid	pin_url	source_url	original_pid	descrip	time_created
1	1	http://www.extremetech.com/w...	http://www.extremetech....	1	My Dream Car	2013-11-25 14:02:42.1700000
2	2	http://www.hdwallpapers.in/wall...	http://www.hdwallpapers....	2	My Dream Bike	2013-11-25 14:02:42.1700000

with both original pins and repins

- The pid and original_pid in different in case of repins. The original_pid in this case refers to the original pin pid in the pins table.
- For eg. in the table below the 3rd and 4th row have different pids and original pids, which means they are repins.

pid	bid	pin_url	source_url	original_pid	descrip	time_created
1	1	http://www.extremetech.com/wp-content/uploads/20...	http://www.extremetech.com/extreme/143025-audi-v...	1	My Dream Car	2013-11-25 14:02:42.1700000
2	2	http://www.hdwallpapers.in/walls/yamaha_r6_bike-H...	http://www.hdwallpapers.in/yamaha_r6_bike-wallpa...	2	My Dream Bike	2013-11-25 14:02:42.1700000
3	4	http://www.extremetech.com/wp-content/uploads/20...	http://www.extremetech.com/extreme/143025-audi-v...	1	Anshul's Dream Car	2013-11-25 14:03:21.0400000
4	4	http://www.hdwallpapers.in/walls/yamaha_r6_bike-H...	http://www.hdwallpapers.in/yamaha_r6_bike-wallpa...	2	Anshul's Dream Bike	2013-11-25 14:03:21.0400000

Like table

pid	like_mid	time_created
1	3	2013-11-25 14:06:45.9430000
1	4	2013-11-25 14:06:45.9430000
2	3	2013-11-25 14:06:45.9430000
2	4	2013-11-25 14:06:45.9430000
3	3	2013-11-25 14:06:45.9430000
3	4	2013-11-25 14:06:45.9430000
3	5	2013-11-25 14:06:45.9430000
4	4	2013-11-25 14:06:45.9430000
4	5	2013-11-25 14:06:45.9430000

Comment table

- Only members who have permission to comment on the board are allowed to make comments on pins on that board. This is achieved through the CommentPin procedure which has already been discussed.

mid	pid	comment	time_created
2	1	nice	2013-11-25 14:07:28.2170000
2	2	Awesome	2013-11-25 14:07:28.2170000
2	3	Aweso...	2013-11-25 14:07:28.2170000
3	2	nice bike	2013-11-25 14:07:28.2170000

Friend table

with pending friend requests

mid	friend_mid	status	time_created
1	2	p	2013-11-25 14:04:16.2830000
1	3	p	2013-11-25 14:04:16.2870000
1	4	p	2013-11-25 14:04:16.2870000

after approving one and declining one friend requests

mid	friend_mid	status	time_created
1	2	y	2013-11-25 14:04:16.2830000
1	3	p	2013-11-25 14:04:16.2870000

Followstream table

When a user creates a followstream entry is made here.

fsid	fsname	mid
1	Vehicles	3

Followstreamboards table

When a user adds boards to followstream entry is made here. Only boards which have been followed can be added to the follow stream.

fsid	bid
1	1
1	2
1	4

Follows table
When user follows a board, entry is made here.

mid	bid
1	5
3	1
3	2
3	4

Tags table
When a tag is created entry is made here. The table is checked before a new entry is made into the table to ensure duplicate tags are not created. The reason on having a separate tags table is save memory space and ensure normalization.

tid	tag_name
1	Cars
2	Fast

Pintags Table
Keeps track of which pin has what tags associated with it.

pid	tid
1	1
1	2
2	2
3	1

NOTE: This design is an initial draft which is subject to change as we move forward in the project. If there is some functionality that is deemed important when developing the frontend of the project, necessary changes will be made to the database schema.