FALL 2013

**PINBOARDS**

**PROJECT for CS6083**

**MIT PARIKH (0487919)**

**AKSHAY PAI (0491264)**

Contents

[1. Introduction: 2](#_Toc373154576)

[2. Functionalities implemented: 2](#_Toc373154577)

[3. Design: 3](#_Toc373154578)

[3.1. Conceptualization: 3](#_Toc373154579)

[3.2. Entity Relationship Diagram: 4](#_Toc373154580)

[3.3. Assumptions: 6](#_Toc373154581)

[4. Relational Schema 6](#_Toc373154582)

[5. Queries and tests on data: 10](#_Toc373154583)

[6. Conclusion and Further steps: 18](#_Toc373154584)

7.Screenshots........................................................................................................................................19

Pinboards

# Introduction:

This project is an attempt to recreate a web-based service to the currently existing “Pinterest”. The service would allow users to share their interests visually by “pinning” pictures on their virtual “boards”. Users can sign up for this service by providing their email address and also selecting a unique username and password. They can then create boards pertaining to different interests and for each board they have the ability to pin pictures on them.

It is also a social platform, so users can also have friends who can see their boards and the pictures they pin on them. Users can also like and comment on pictures pinned by other users. Apart from “liking” and “commenting” on other people’s pictures, users can also “repin” picture from other users on their own boards. Users can also create streams, where they can follow pictures from various boards. Ability to search for pictures pertaining to various tags is also provided.

# Functionalities implemented:

Following is the list of features or functionalities that the service would be able to provide:

* Users can join by selecting a unique username. They need to provide a password and also a valid email address with that.
* Each user has a profile that they can edit. This includes details like first name, last name, date of birth, summary etc.
* Users can then create boards with unique names amongst their boards. That is, a user cannot have two boards with same name, but two different users can have boards with same name.
* Users have ability to pin pictures to boards. They can do this by two ways, either through the web or uploading from computer.
* Pictures can have tags associated with them to describe what they are for.
* Users can be friends with other users on pinboards.
  + They can add friends by sending requests to other users.
  + Users can accept or deny friend requests.
* Users are able to view pictures pinned by all users on pinboards.
* Apart from viewing, a user can “repin” any “pinned picture” on any of their boards.
* Users can also “like” pictures. Each “like” is on the original picture that is pinned.
* Users can also “comment” on pictures. Users are also given the option to put restrictions such that, for some boards, only the friends of that user can comment on the pictures pinned on those boards.
* In contrast to “likes” being on the original picture, “comments” are associated with individual “pins”.
* Users can create “streams” which would allow them to follow boards of other users. Thus, whenever the user follows any board of any other user, that board would be added to any stream that the user wants.
* Users also have an option for searching a picture by its tags. Thus pictures containing the same tags (similar pictures) can be found.

# Design:

## Conceptualization:

On a conceptual stage, we tried to keep the database design as simple as possible. The database concentrates on the strong entities involved in the database. These strong entities are Users, boards, pictures and pin. Each of these entities would have their detailed attribute set. There were 2 weak entity sets “userinfo” and “stream” which are both dependent on existence of users. After entities we thought of the relationships amongst these entities. We found that friendship, likes and comments are relationship sets.

“User” table will have details regarding username, password and the email address associated with the user account.

“Userinfo” will contain all the personal information like the first name, last name, date of birth, hometown etc. User can edit these whenever he/she wishes.

“Friendship” table contains all the relations of friendships amongst the users of pinboards. Whenever a user sends a friend request to another user, a new entry with id of the first user is saved in sendFriend and the id of the other user is saved in the column acceptfriend. If the other user accepts the friend request then a new entry is created with the ids reversed. This makes it easier to differentiate friend requests and also to find various things like how many friends each user has, how many pending requests etc.

“Boards” contain information of each board like it’s unique boardid, the name of the board, username of the user who created the board and also a Boolean field which keeps track whether the user has restricted commenting on that board to only his/her friends.

“Pictures” records all the information for each picture that is uploaded on the system. This includes the tags associated with that picture, its location on the server and also number of likes on that picture.

“Pin” is a table that keeps information about unique id of each pin, id of the picture that is being pinned/repined. It also has username of the person pinning/repinning the picture, the board on which it is being pinned/repinned and the timestamp to store its timestamp. There is also a “repinid” which contains the id of the original “pin” that is being repinned. If it is a “pin” then “repinid” would remain null.

“Likes” keeps a record of which user likes which picture and at what time. There is a trigger on this table which after inserting any row into this table, automatically updates the numOfLikes attribute of the same picture in the picture table.

“Comments” table keeps information about who commented what on what “pin” and at what time. The content of the comment is also stored in the table.

“Stream” table is used to keep record of which streams a user is creating and for each stream, picture from which boards are being followed.

## Entity Relationship Diagram:

For the diagram below:

Strong Entities:

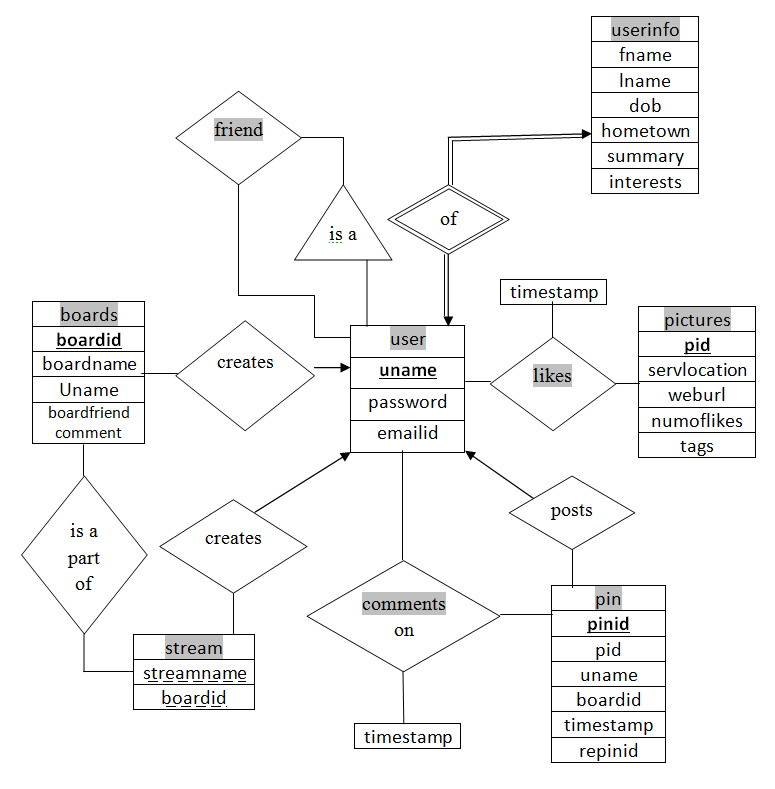
* User
* Boards
* Pictures
* Pins

Weak Entities:

* Userinfo
* Stream

Relationship Sets:

* Friendship
* Likes
* Comments



Keys, constraints and cardinalities are shown in the diagram.

## Assumptions:

We consider the following assumptions for the database:

* A username would be unique to each user. And would be the unique identifier for users throughout the database schema.
* User would not be creating more than one board with the same name.
* Each picture is stored on the server at some location and only its path/address is stored in the database. The actual content of the picture is stored at that path/address.
* There is unique “pinid” for each act of pinning/repinning.
* Tags are stored as an attribute to the picture and would remain constant, even when it is repinned by some other user. To search for pictures having same/similar tags we use the “contains” operator in the “tag” attribute of the picture table.
* All pins/repins are public and can be seen by any user who is a member of pinboards.
* The same concept is true for likes. Any user can like a picture. Each “like” is a “like” on the original picture. This means even if the picture is repinned and someone likes it, it still counts as a like on the original picture.
* As for commenting, a comment is on the “pin” and not on the “picture”. Thus any user “repin” a picture and there would be a new set of comments for that.
* A user also has chance to put restrictions on whether to allow only his friends to comment or any user on pinboards.

# Relational Schema

From the E-R diagram, we can create a relational schema as below:

**USER** (uname, password , emailid)

**USERINFO** (uname, fname, lname, dob, hometown, summary, interests)

uname referenced from USER table

**BOARDS** (boardid, boardname, uname,boardfriendcomment)

uname referenced from USER table

**FRIENDSHIP** (sendFriend, acceptFriend)

sendFriend, acceptFriend referenced from USER.uname

**PICTURES** (pid, servlocation, weburl, numoflikes, tags)

**PIN** (pinid, pid, uname, boardid, pin\_timestamp, repinid)

pid referenced from PICTURES.pid ; uname referenced from USER.uname;

boardid referenced from BOARDS.boardid; repinid reference from PIN.pinid

**LIKES** (pid, uname, like\_timestamp)

pid referenced from PICTURES.pid; uname referenced from USER.uname;

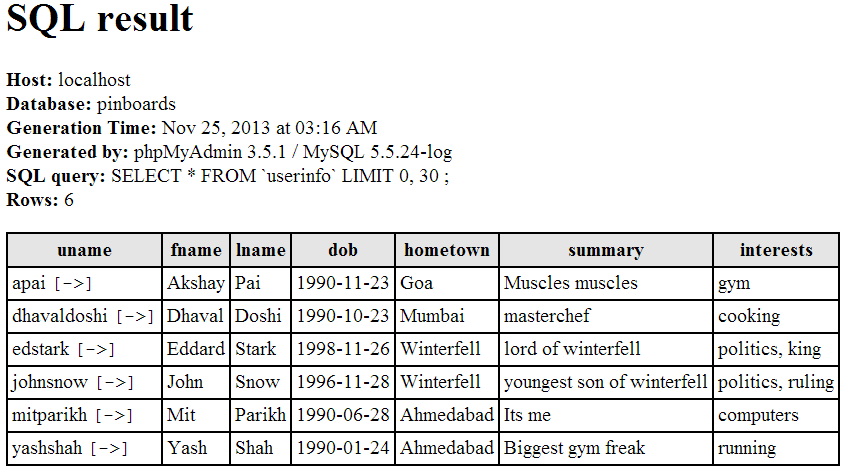
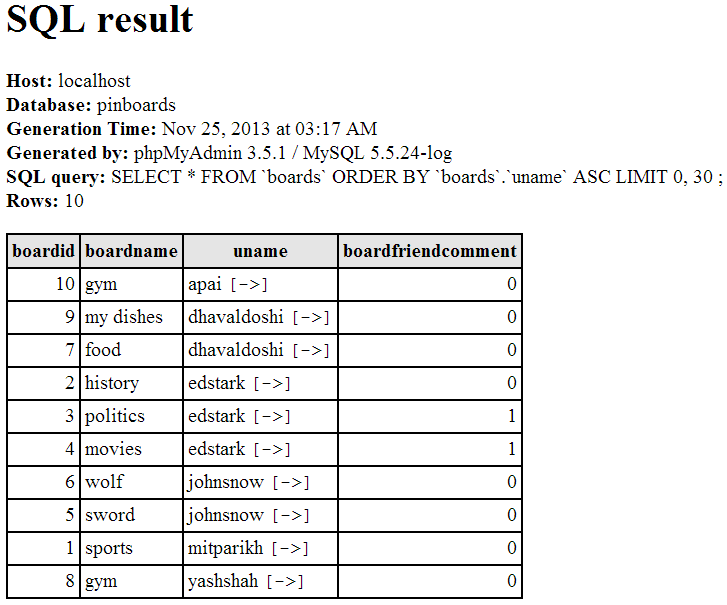
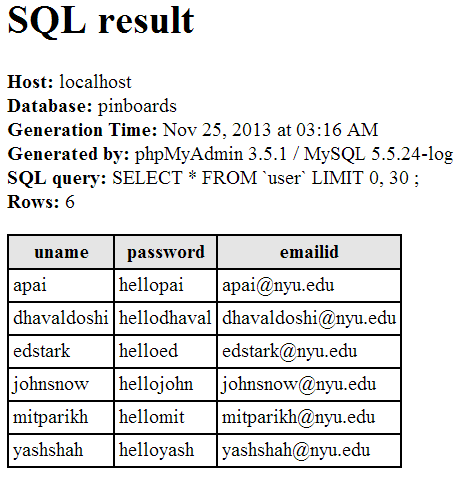
**COMMENTS** (commentid, pinid, uname, comment, comment\_timestamp)

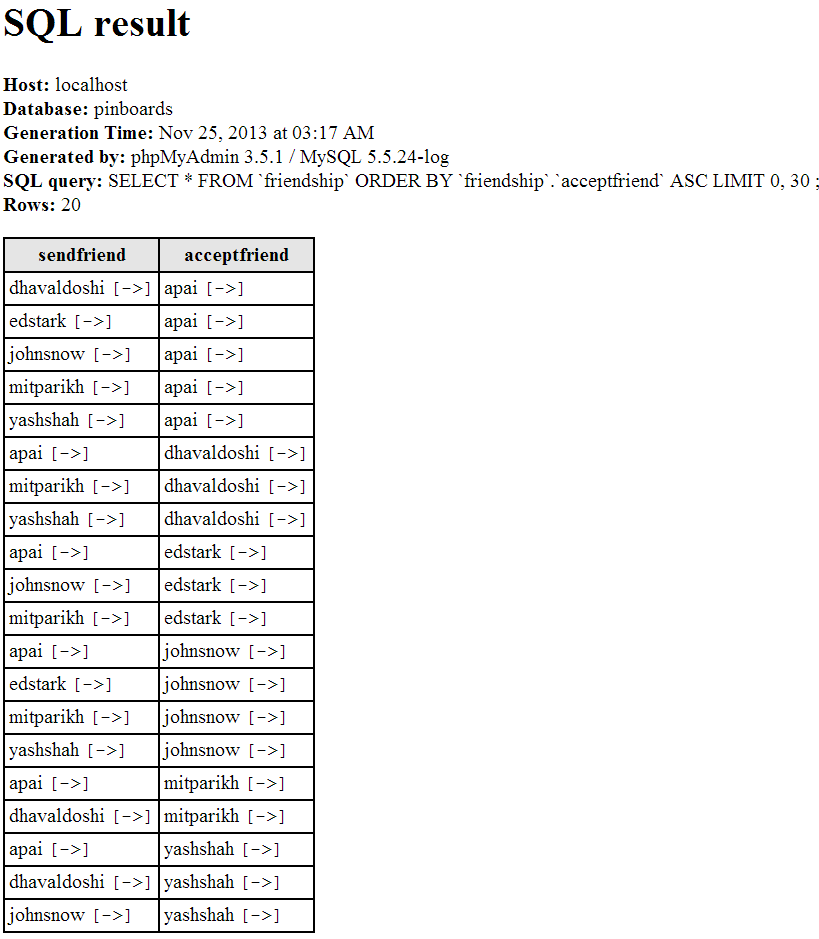
pinid referenced from PIN.pinid; uname referenced from USER.uname;

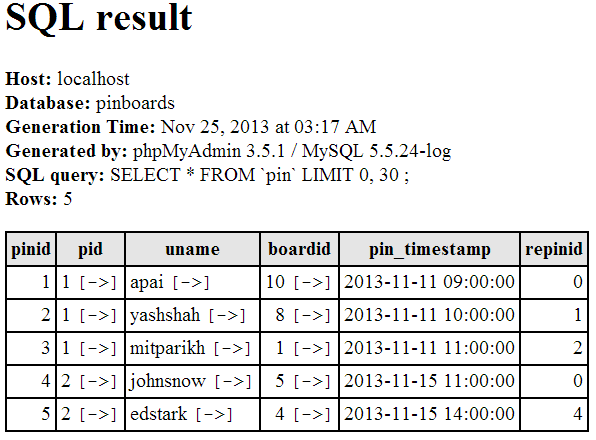
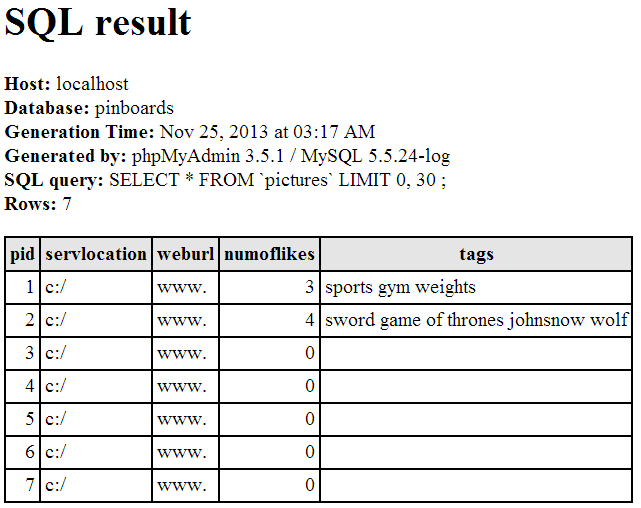
**STREAM** (streamname, uname, boardid)

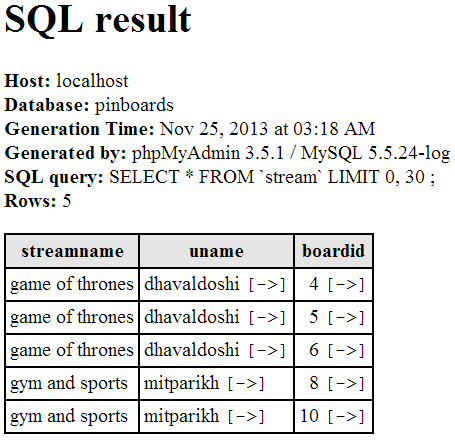
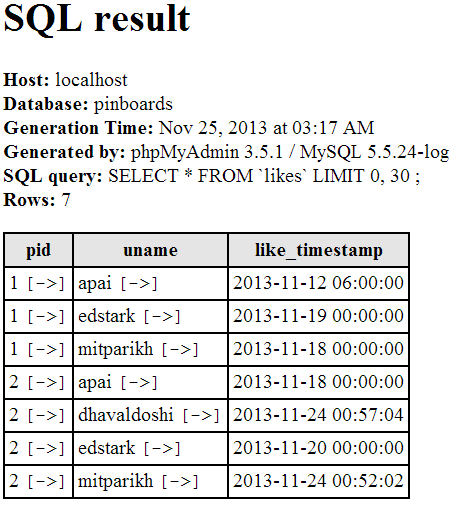
uname referenced from USER.uname; boardid referenced from BOARDS.boardid

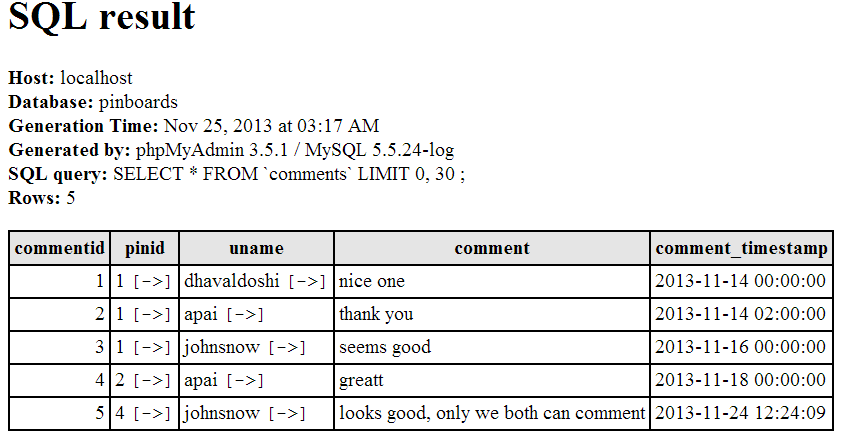
Find below the created tables with sample data:











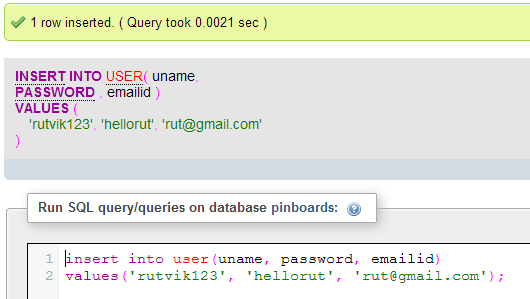
# Queries and tests on data:

Below are the queries for all the tasks that the system would perform. We are taking sample data at this stage, but after the front end is integrated, the sample data would be replaced by the data that would be sent on the server from the browser. This is to show that all the tasks can be fulfilled using these queries.

**Query to sign up:**

Inserting into the user table, the values of username, password and emailid, that we get as input from the from:

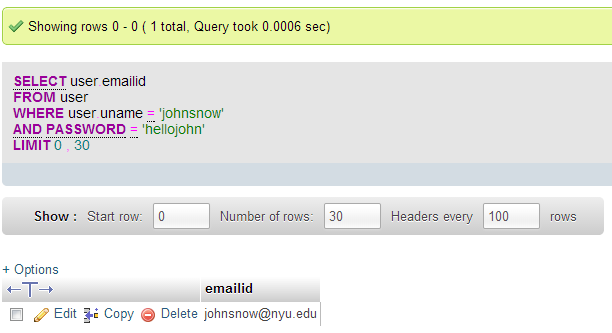
*insert into user(uname, password, emailid) values('rutvik123', 'hellorut', 'rut@gmail.com');*



**Query to login:**

Displaying emailid of the user whose signin details (username and password) are taken from the form, if no record is found, we could disallow the sign in:

*select user.emailid from user where user.uname='johnsnow' and password='hellojohn'*;

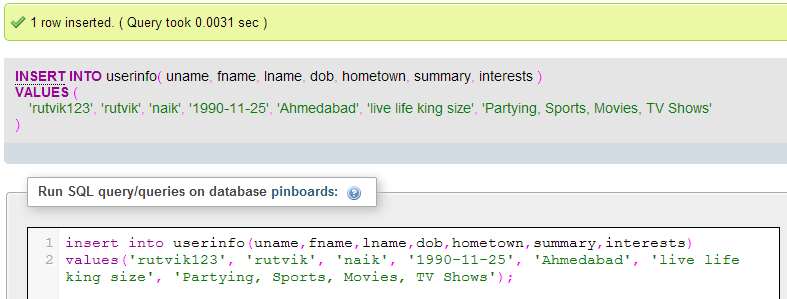


**Query to create profile:**

Inserting the details of user, that we get from the form, into the userinfo table:

insert into userinfo(uname,fname,lname,dob,hometown,summary,interests)

values('rutvik123', 'rutvik', 'naik', '1990-11-19', 'Ahmedabad', 'live life king size!', 'Partying, Sports, Movies, TV Shows');



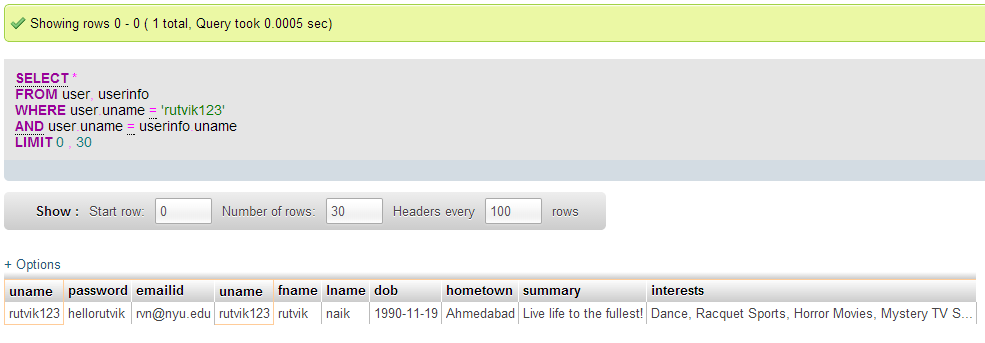
**Query to edit profile:**

We are allowing the user to change everything other than the username. Updating the password, emailid, and other details that the user wishes to change, into the user and userinfo tables respectively:

*update user set password='hellorutvik', emailid='rvn@nyu.edu' where uname='rutvik123';*

*update userinfo set summary='Live life to the fullest!', interests='Dance, Racquet Sports, Horror Movies, Mystery TV Shows' where uname='rutvik123';*

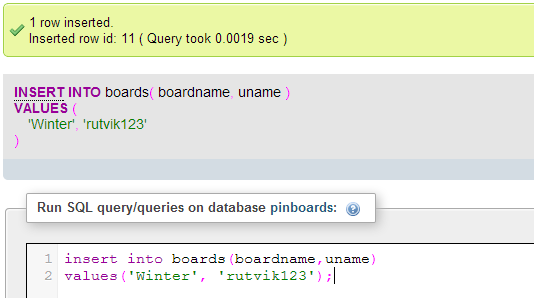
*select \* from user, userinfo where user.uname='rutvik123' and user.uname = userinfo.uname;*



**Query to create a new board:**

Inserting into boards table, the name of board, which we get from the form, and the name of user who created that board:

*insert into boards(boardname,uname) values('Winter', 'rutvik123');*

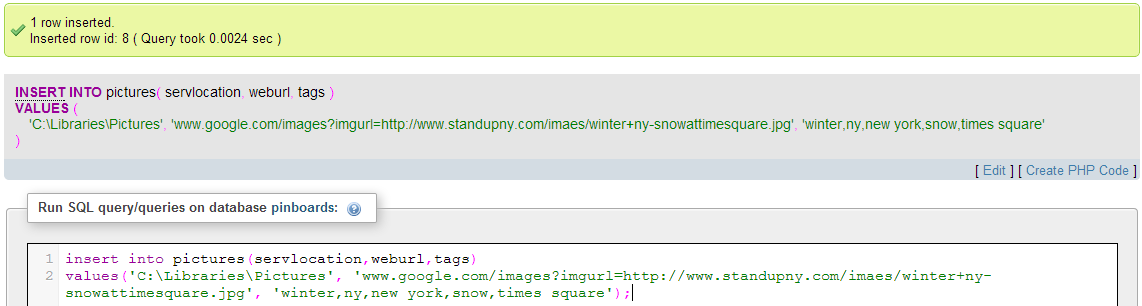


**Query to pin a picture:**

Inserting into pictures table, the weburl pasted into the form by the user, and any tags the user wishes to add, along with the server location where the picture is downloaded (in order to retain it even if it is removed from the website)

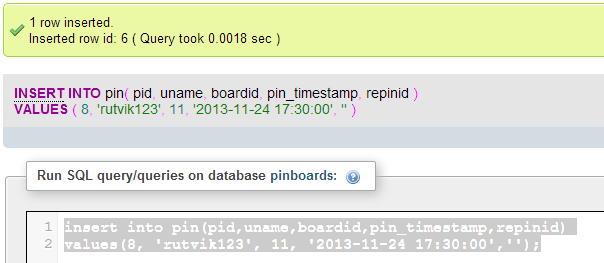
A corresponding entry is created in the pin table, with details of the pin created by the user:

*insert into pictures(servlocation,weburl,tags) values('C:\Libraries\Pictures', 'www.google.com/images?imgurl=http://www.standupny.com/images/winter+ny-snowattimessquare.jpg', 'winter,ny,new york,snow,times square');*



*insert into pin(pid,uname,boardid,pin\_timestamp,repinid)*

*values(8, 'rutvik123', 11, '2013-11-24 17:30:00','');*

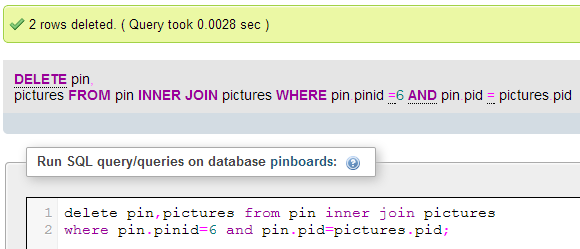


**Query to delete picture:**

Deleting corresponding records from both, pictures and pin tables:

*delete pin,pictures from pin inner join pictures*

*where pin.pinid=6 and pin.pid=pictures.pid;*

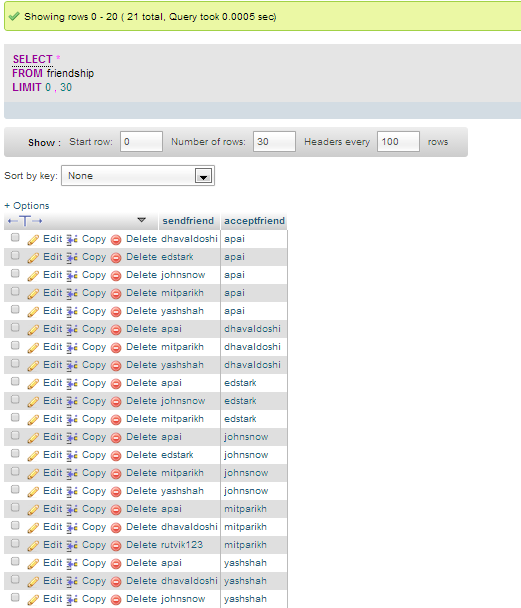


**Query to send friend request:**

rutvik123 sends friend request to mitparikh, creating an entry in friendship table, with rutvik123 as sendfriend and mitparikh as acceptfriend:

*insert into friendship(sendfriend,acceptfriend) values('rutvik123', 'mitparikh');*

*select\* from friendship;*



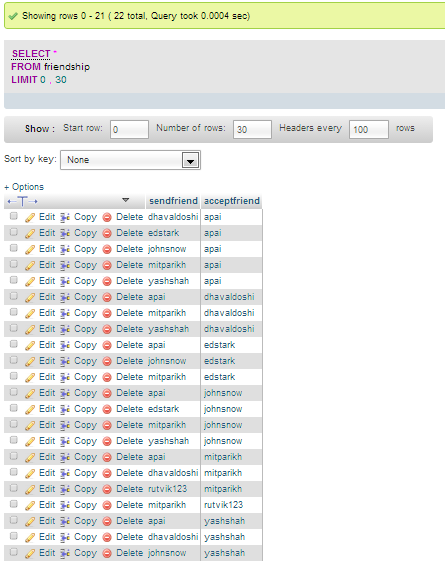
**Corresponding query to accept friend request:**

mitparikh accepts friend request from rutvik123, creating an entry with mitparikh as sendfriend and rutvik123 as acceptfriend. If the friend request was not accepted, this entry with interchanged sendfriend and acceptfriend attributes, would not be created, indicating a pending request. If the request was denied, the previous entry, indicating a friend request sent, would be deleted.

*insert into friendship(sendfriend,acceptfriend)*

*values('mitparikh', 'rutvik123');*

*select \*from friendship;*

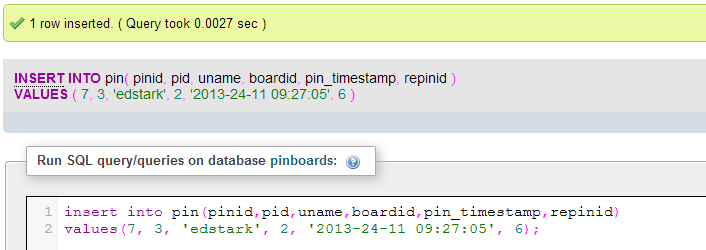


**Query for repinning a picture:**

edstark repins johnsnow's picture(pid-3) from pin(pinid-6) to his own board(boardid-2):

*insert into pin(pinid,pid,uname,boardid,pin\_timestamp,repinid)*

*values(7, 3, 'edstark', 2, '2013-24-11 09:27:05', 6);*



**Query for creating a follow stream and adding boards to it:**

creating the stream 'food and health' for the first time, while adding board(boardid-7) to it:

*insert into stream(streamname,uname,boardid)*

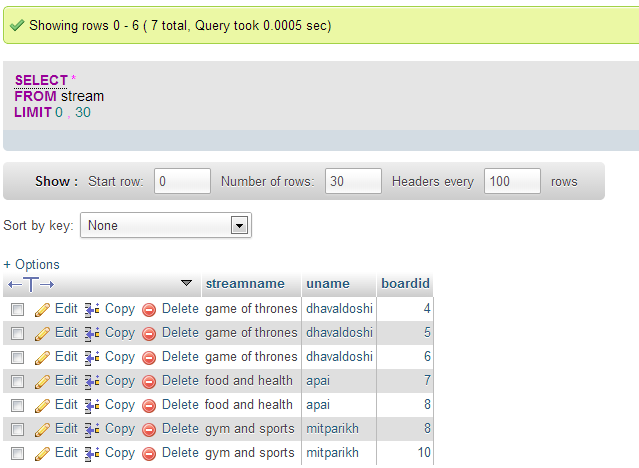
*values('food and health', 'apai', 7);*

adding another board(boardid-8) to the stream:

*insert into stream(streamname,uname,boardid) values('food and health', 'apai', 8);*

displaying the result:

*select\* from stream;*



**Query for displaying pictures belonging to the boards in a follow stream:**

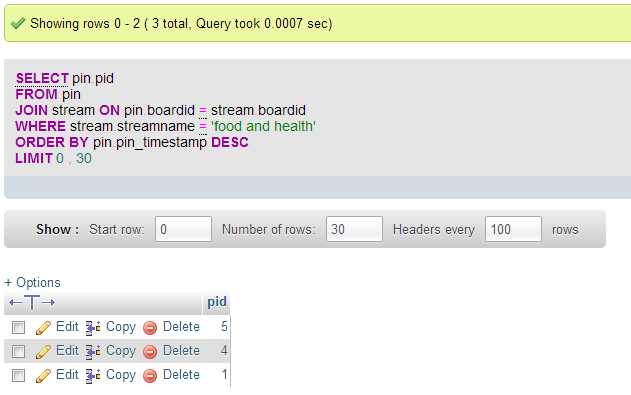
display pictures(pid) which belong to all the boards in the stream 'food and health' in reverse chronological order in which they were added:

*select pin.pid from pin join stream*

*on pin.boardid=stream.boardid*

*where stream.streamname='food and health'*

*order by pin.pin\_timestamp desc;*



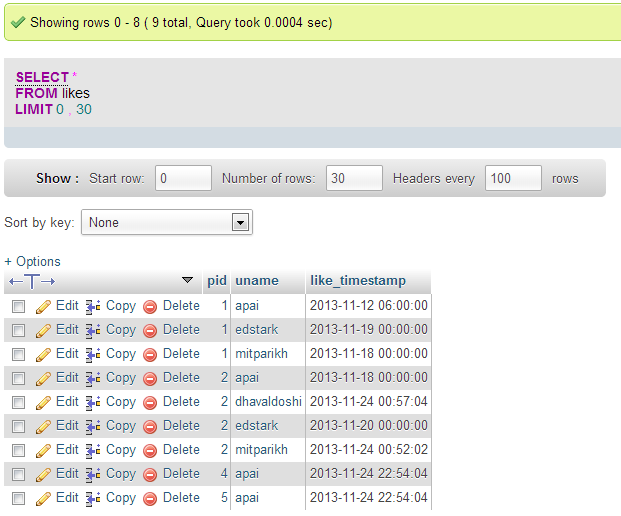
**Queries to like pictures:**

Inserting records in the likes table. We have added a trigger to increment the numof likes field in the pictures table for the respective picture:

*insert into likes(pid,uname) values(4, 'apai');*

*insert into likes(pid,uname) values(5, 'apai');*

*select \*from likes;*



**Query to comment on pictures:**

We are planning to check the condition of whether the user is allowed to comment on the picture or not at front end level. For now we have tried using this logic, but it does not work. We would work upon it in the second part of the project.

*insert into comments(pinid,uname,comment) values(4,'apai','cool pic!')*

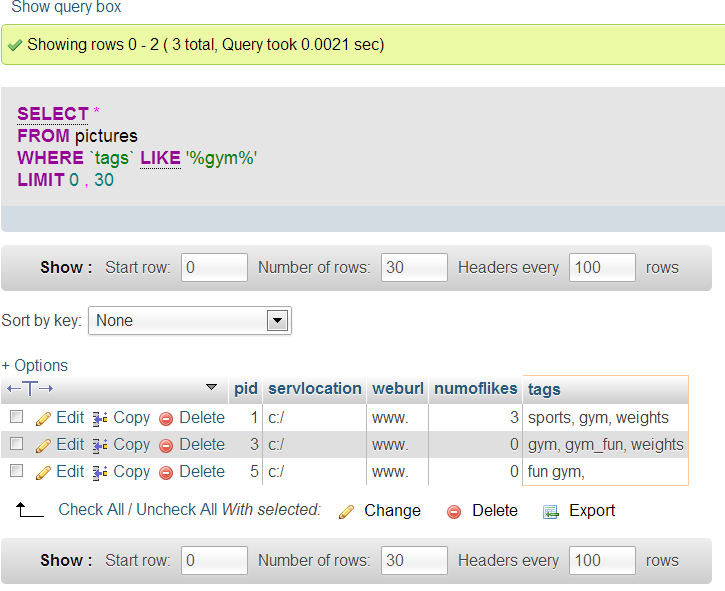
*where (('apai'=friendship.sendfriend and friendship.acceptfriend=(select pin.uname from pin where pin.pinid=4)as friend2) and (friend2=friendship.sendfriend and 'apai'=friendship.acceptfriend)or friend2='apai');*

**Query to do keyword search for pictures using tags:**

Here we have shown a query that for a given keyword would return all the pictures having that keyword associated with it in the form of “tag”. Various tests regarding what would happen if the keyword was a single letter, or a white space or “\_”, have been performed and during all those conditions the query outputs the perfect result.

*SELECT \* FROM pictures WHERE `tags` LIKE '%keyword%'*;

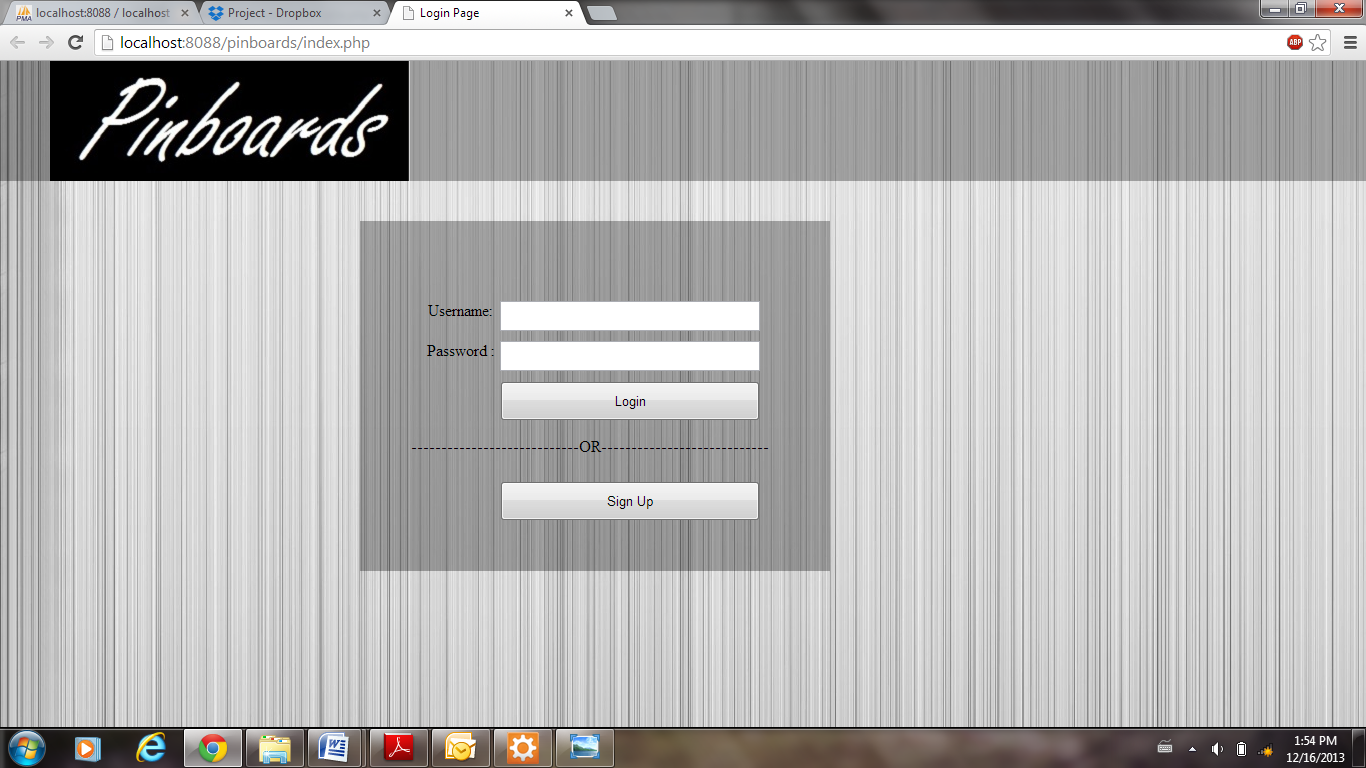
Here we use the keyword as *gym*



# Screenshots

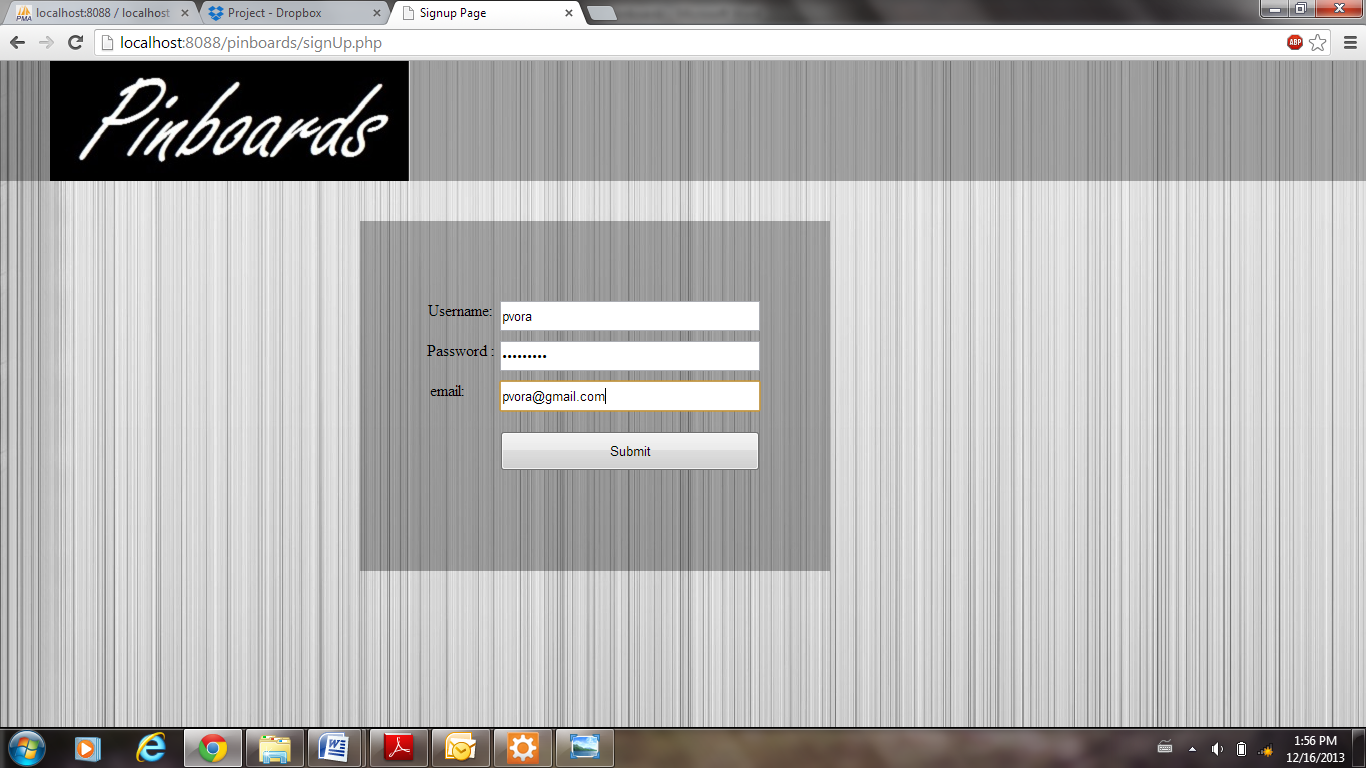
Login Page:

This is the first page of the web service. It takes login details from the user, which include username and password, both of which are mandatory for a login, without which the login fails. A Sign Up option is also provided.



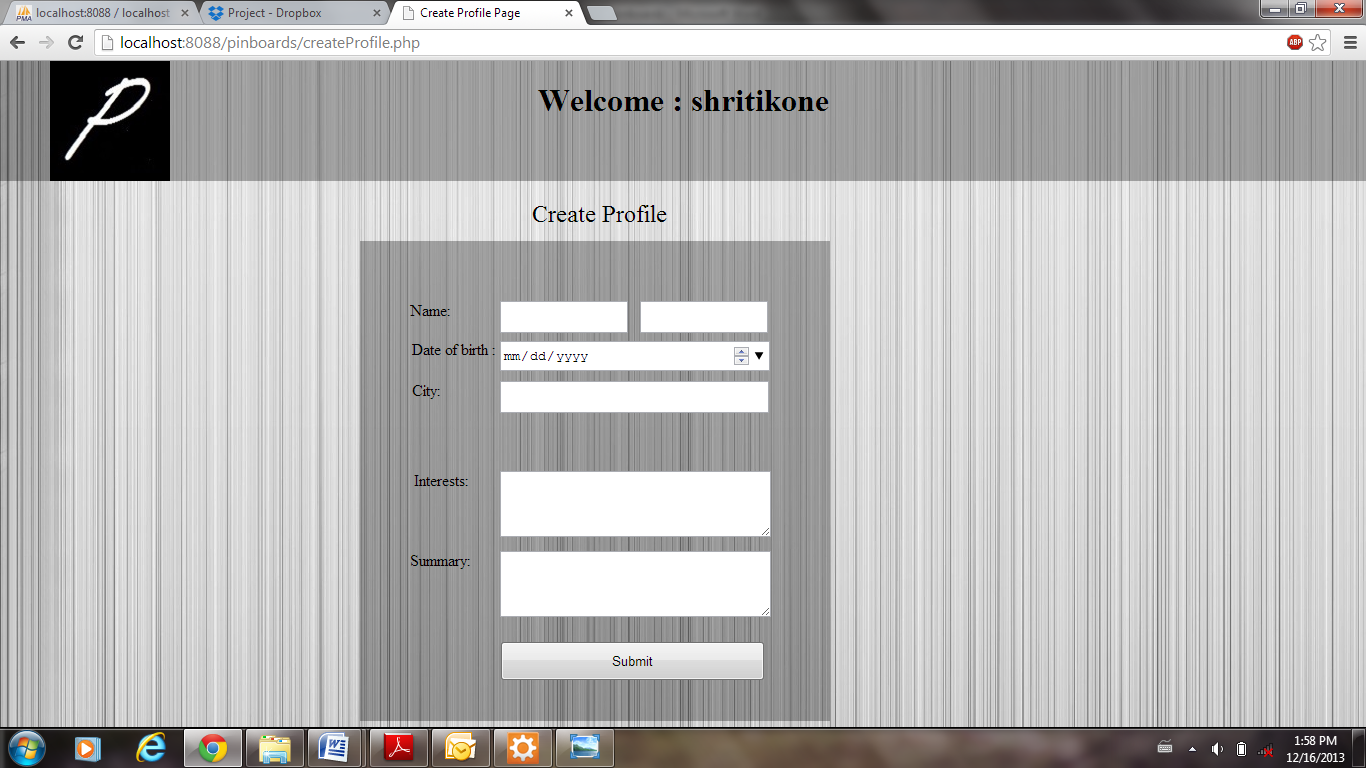
Sign Up Page:

This page requires the user to choose a username and a password, along with an email address to associate with it, in order to create a new account with "Pinboards". Email address is an optional field here. Validation of email address is included, in order to check the format of text entered.



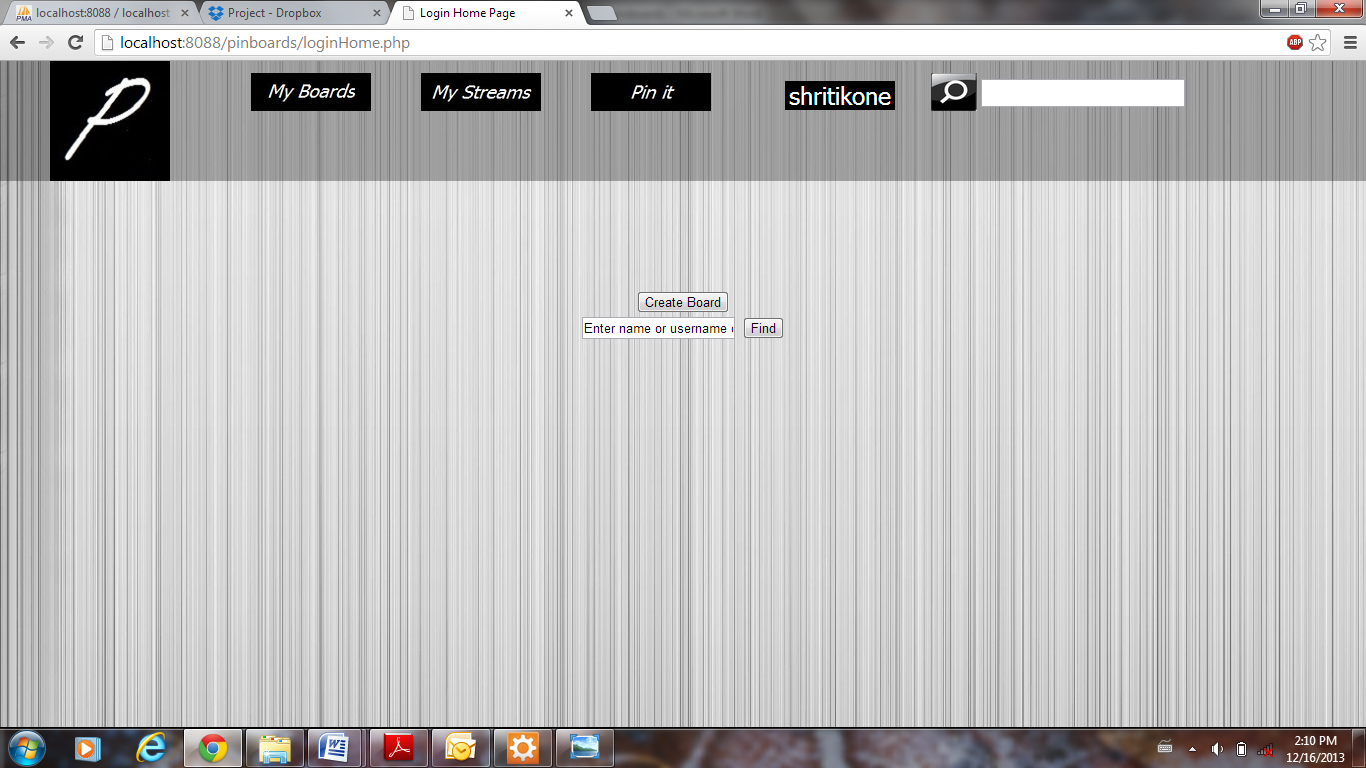
Create Profile Page:

After Signing Up, the user is directed to the create profile page, so that user information could be recorded. First Name and Last Name fields are mandatory.



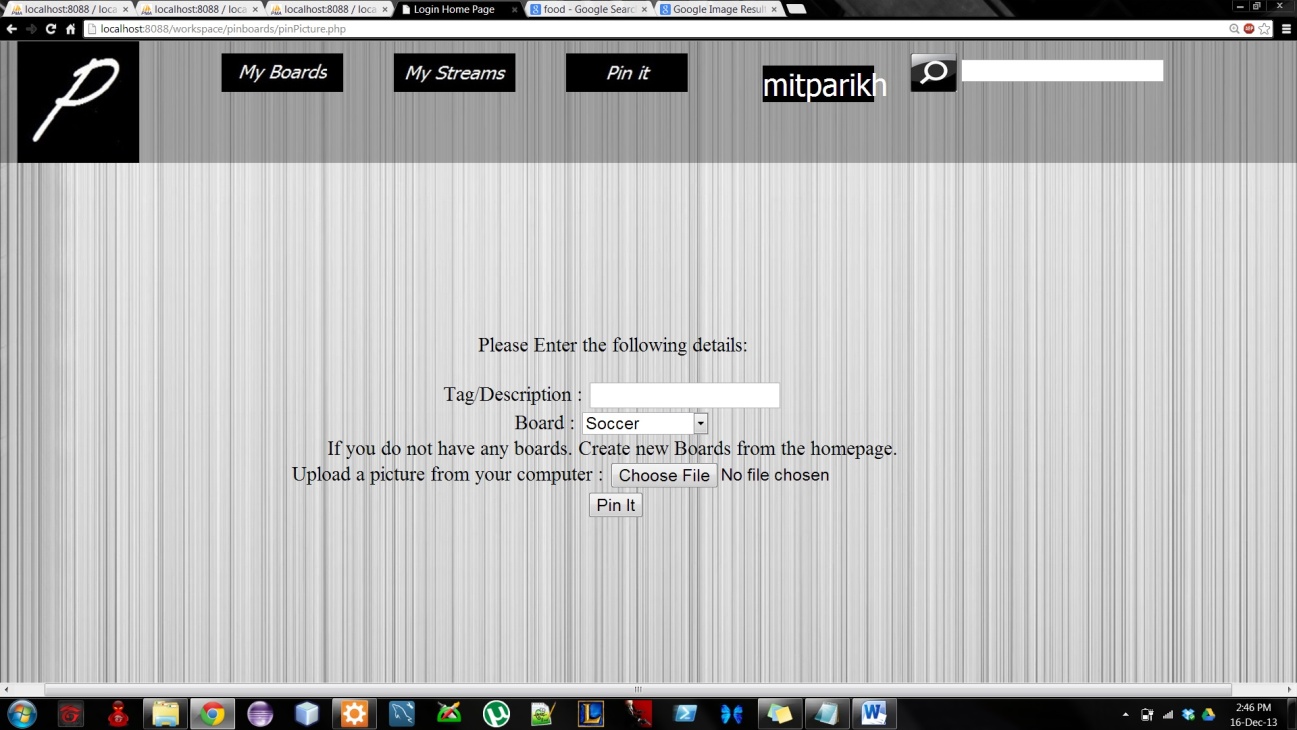
Login Home Page:

After successfully creating the profile, user is directed to the login home page. This is the base returning point from all other pages, and provides links to many activities, like viewing boards, creating boards, viewing streams, pinning pictures, searching pins, finding new friends by their first name/ last name or username.



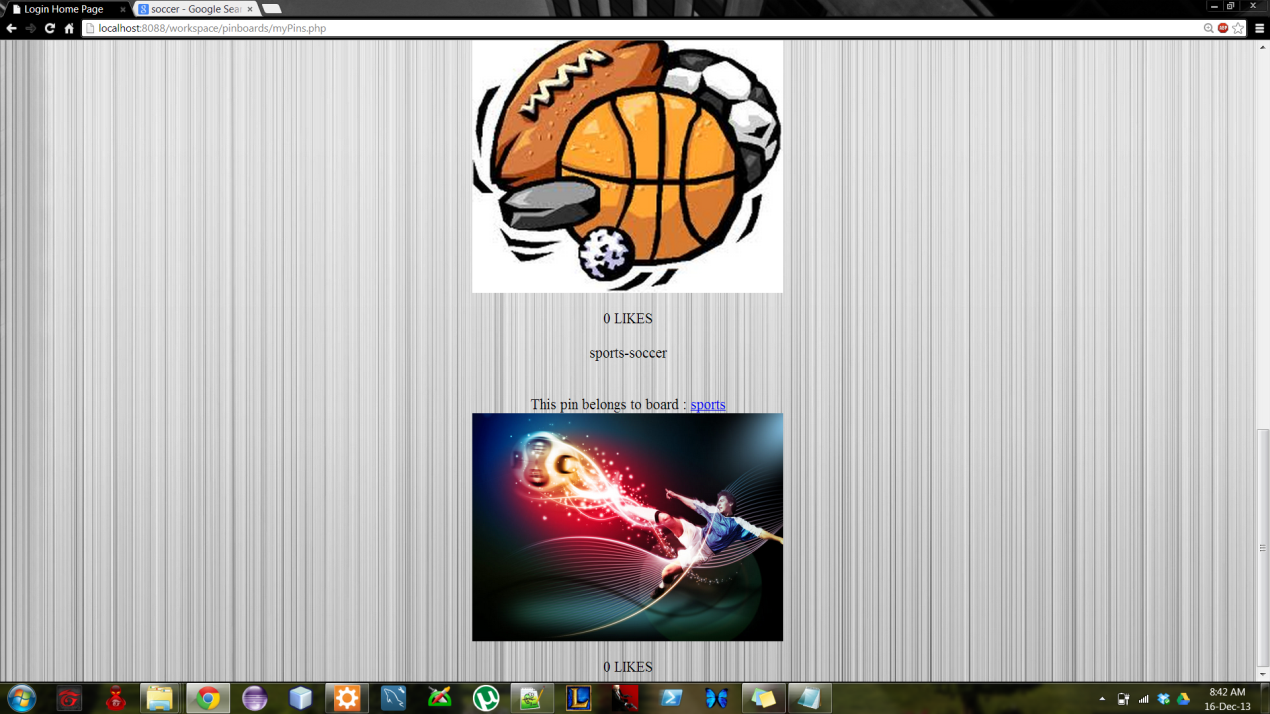
Pin Picture Page:

This page takes you to a form which takes in the tags, allows user to choose file to upload, board name to which it is to be pinned.



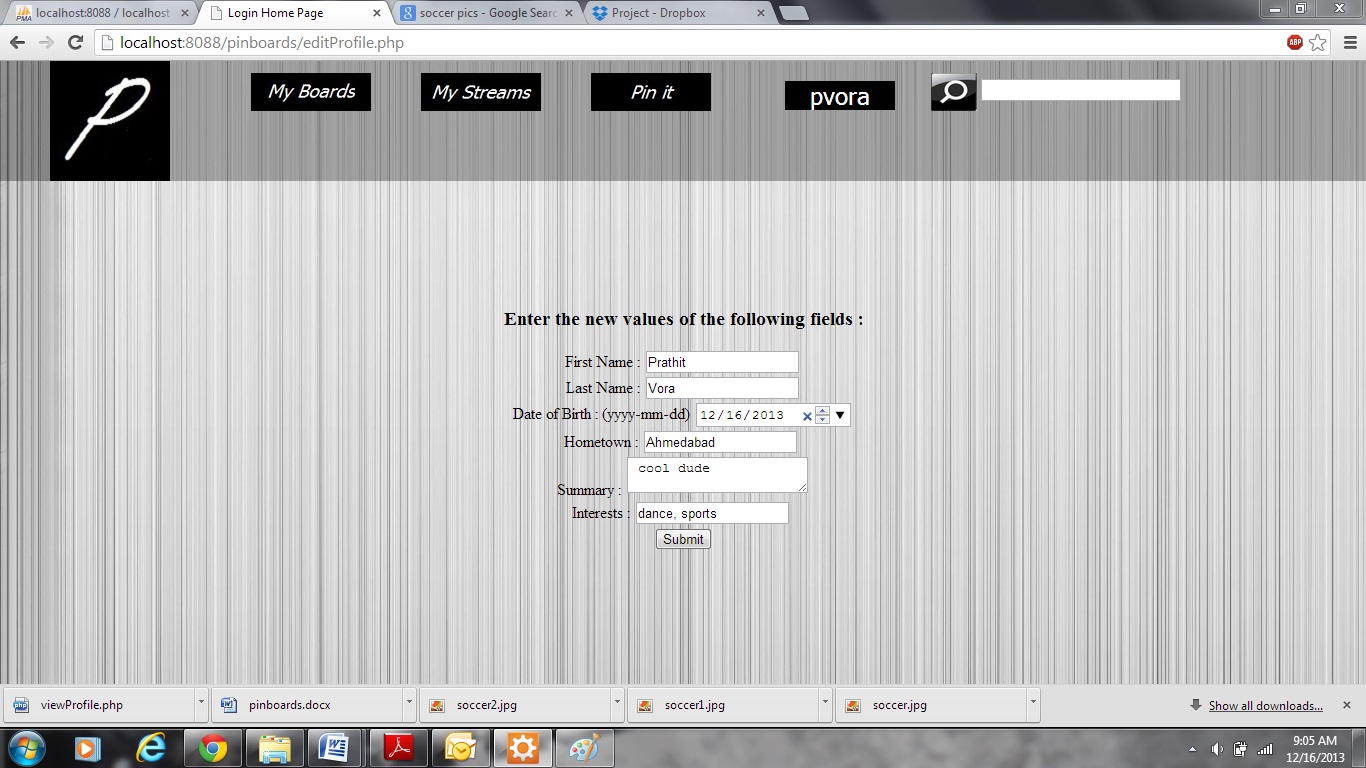
My Boards Page:

This page display all the boards belonging to the user, with thumbails. On clicking on these thumbnails, it redirects you to the page which has all pictures of that board.



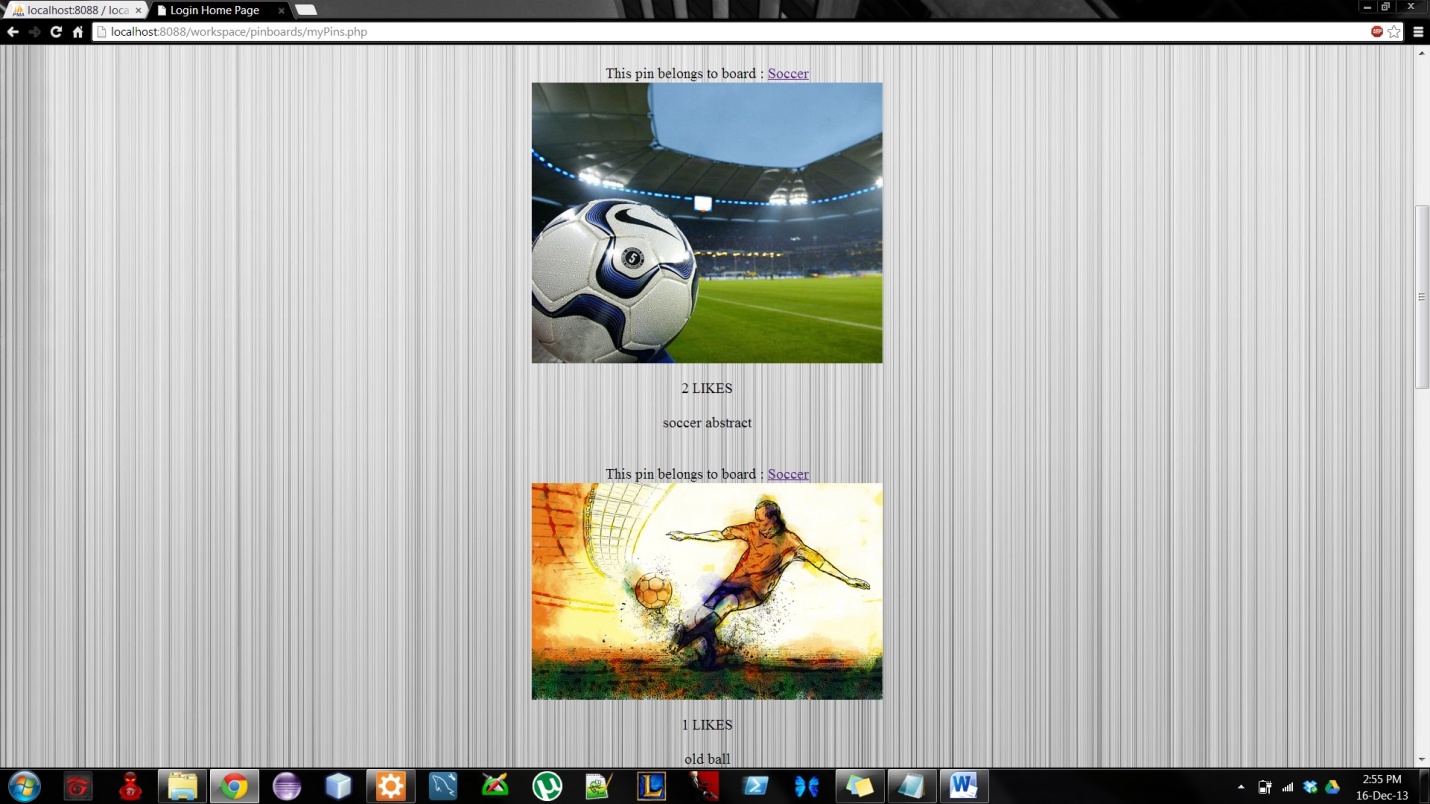
Edit Profile Page:

This page allows the user to change the user data previously entered.



My Pins:

Takes the user to the page containing all pins posted by him/her.



# Conclusion

Here, we conclude with the database as well as the user interface design and implementation phase of the web-service.