Report

**Group Name COMP 353\_group\_12**

**Student ID First Name Last Name Email**

1707000 Jason SOTZKY j\_sotzk@encs.concordia.ca

1941097 Michal WOZNIAK m\_wozni@encs.concordia.ca

1955802 Francis BAYARD f\_bayard@encs.concordia.ca

6676138 Daniel HACKL d\_hackl@encs.concordia.ca

9649727 Sebastian Rafique PROCTOR-SHAH s\_procto@encs.concordia.ca

Table of Contents  
  
Project Description……………………………………………………………………………………………………………………3

Assumptions……………………………………………………………………………………………………………………………..4

The Limitations………………………………………………………………………………………………………………………….5

The Application Supported………………………………………………………………………………………………………..6

Architectural Decisions………………………………………………………………………………………………………….7-8

E-R Model………………………………………………………………………………………………………………………………….9

Relational Database Design………………………………………………………………………………………………………10

3NF Solution……………………………………………………………………………………………………………………....11-12

Member Responsibilities…………………………………………………………………………………………………….13-14

Gantt chart………………………………………………………………………………………………………………………………15

Detailed Analysis of Coding the Website..…………………………………………………………………………..16-18

The Interface Design Rational…………………………………………………………………………………………………..19

Queries……………………………………………………………………………………………………………………………….20-23

User Manual……………………………………………………………………………………………………………………….24-29

Website Demo…………………………………………………………………………………………………………………….30-33

Installation Instructions……………………………………………………………………………………………………………34

TEAM Participation using Facebook Group………………………………………………………………………...35-36

GIT Log……………………………………………………………………………………………………………………………….......37

Improvements since Demo…………………………………………………………………………………………………38-40

**Project Description:**This project aims at creating a relational database system in the form of a private social networking website. The POWON (acronym for Private Online What’s On Now) System as the project is named has many of the characteristics associated to social networking websites and forums. Amongst the notable features available in POWON is the ability to add friends and family, to create groups, the ability to send messages and emails, to change privacy settings and the ability to receive status updates on groups or individuals. To describe POWON in terms of existing systems, it resembles a cross between Facebook and internet forums. Over the course of two months, the project has flourished from a single posting page to a network of groups and friends with features such as gift giving and event creations and voting.

**Assumptions:**  
The project has been changed multiple times since its inception at the beginning of the fall semester. The original design for POWON was inspired by the Facebook model. As the project progressed, a shift was made towards a forum look rather than a social networking design due to time limitations.

Many productivity tools where used such as GitHub where used and eventually scraped during production. We considered using TortoiseSVN but ended up using the GitHub instead due to familiarity of the members of the group with its simplistic design.

**The Limitations:**

Limitations exist when we consider features that where part of POWON’s original design but where either abandoned or simplified during the implementation phase of the project. Due to the Concordia clipper servers which was used to deploy the POWON project, concession where made in order to allow the website’s full functionalities. For instance, we were not able to post very large videos to the group threads. As a matter of fact, any kind of input to the Concordia servers where limited in size. With the time constraints from having other projects and assignment for this course, many features where scoped out as well or reduced to simpler versions of their original design. Examples of these are:

* The news feed which was reduced to look more like forum threads. Only the site admin now has the power to make public posts which appear on the front page/news feed, and was implemented as a thread.
* We left the events functionality to finish before the demo. The deployment of project to the servers and testing/fixing bugs took more time than anticipated, so it was not implemented for in time for the demo but for the final submission.
* The original design made prevented active users from viewing inactive ones
* Website security and input validation was originally planned but many input boxes where changed to radio buttons and check boxes
* GitHub was originally used to merge coding but was abandoned three quarters through the project production because of issues with merging code based on an older/un-updated versions of the project with newer versions of the project.

**The Applications Supported**

The project was mainly aimed at making a website working on browsers from the Widows environment. Once the project was transferred to the servers we were able to test it on Firefox and Windows Explorer. Both worked as expected with the exception with issues regarding JavaScript scripts which were fixed before the early demo of the project.

Since we used the CSS bootstrap framework, the website was found to scale properly on mobile environments such as the Android version of Google Chrome and iPhone’s Safari. The scaling was tested on a Google Nexus 5 smartphone, on a Samsung Galaxy SIII and on an iPhone 5.

**Architectural Decisions:**

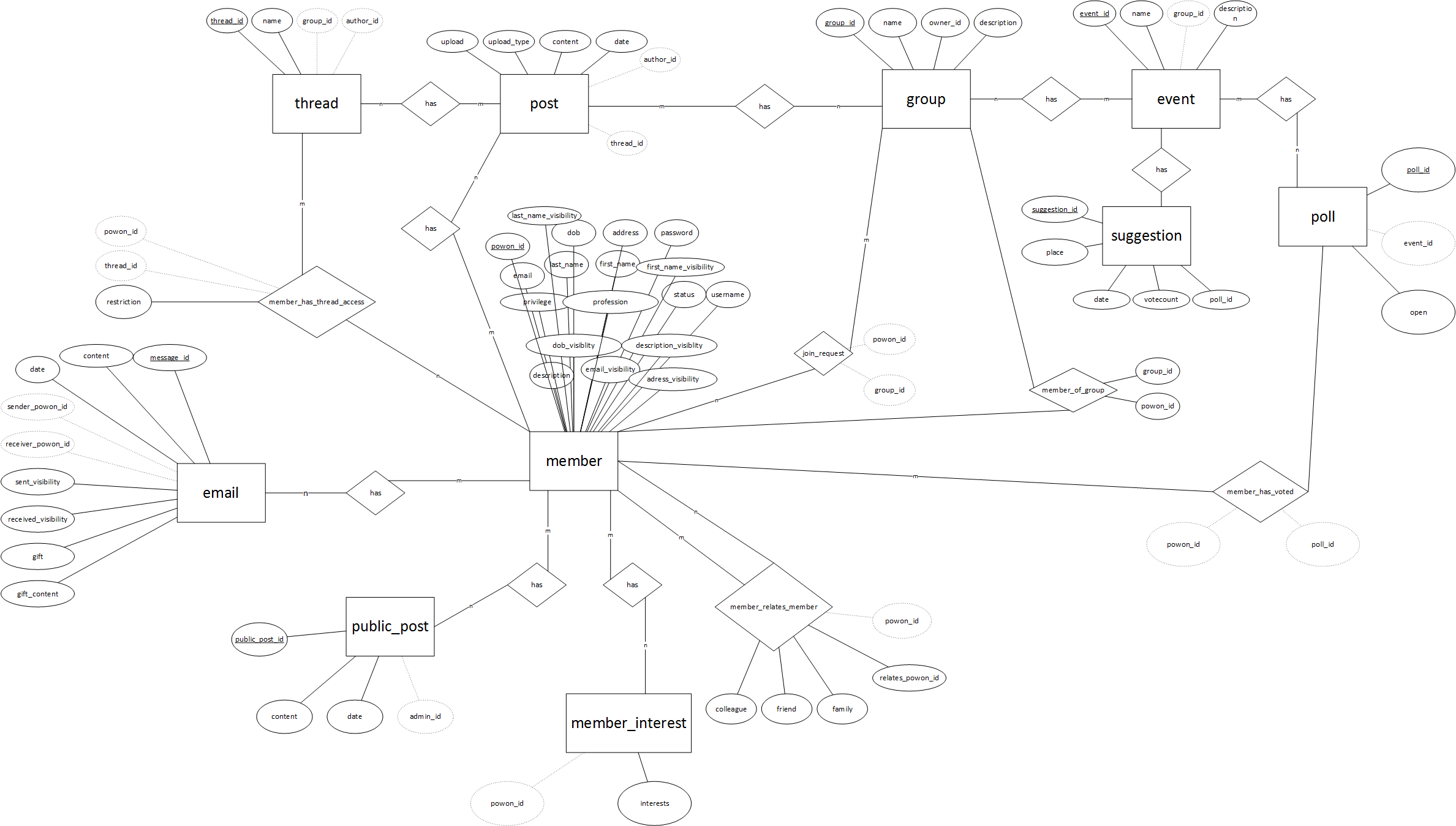
The original model had eight different Views:

* The basic heads up display: Links to all the other templates
* The home view: Has the links to all the other views and the input for login
* The person info view: Has the info user info (name, email, etc.)
* The personal setting view: Has the privacy setting and personal setting (change name, address etc.)
* The group info page: Has group info (determines who is in the group, public/private setting etc.)
* The group setting page: The owner can set admins and admins (including the owner) can add members, change the group privacy setting etc.
* Group post page: Display of all the post inside the group by group members or public
* Mail page: Displays the mail sent and receive from other users

Due to time restrictions the complexity of the project the views where split into different pages to allow easier testing. The main pages changed to the following:

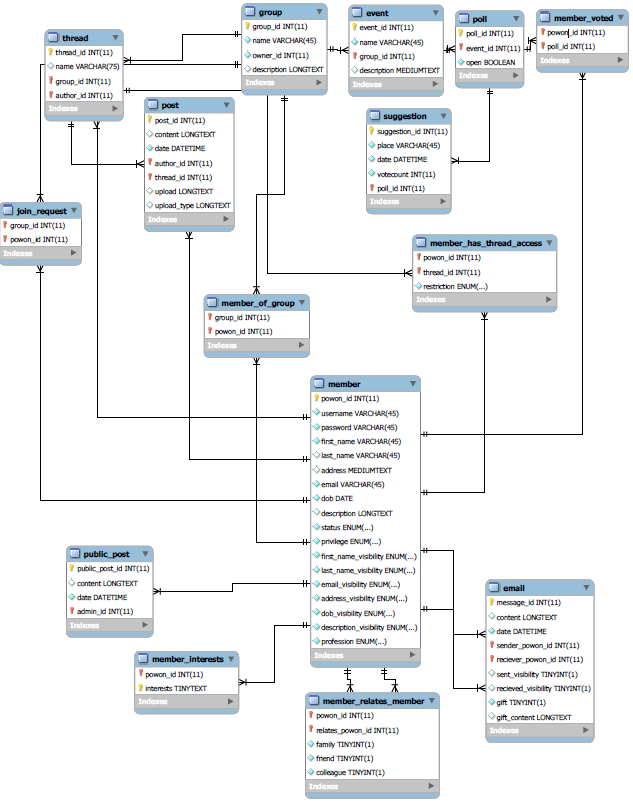
* Home view: The home page is now the index page of the website and has access to all the possible pages and links available to the user or admin. It also includes the Public Posts, the Threads and links to the Email page, the Profile page, the groups and the Member Options (View Profile view, Create Group view, Search Groups view, Search Members view)
* Email view: Instead of having its own page, the email view overlaps the current view to displays in tool bar.
* Profile Page: The personal info page and the personal settings page where merged together. Additionally, a section to add a profession and interests was added.

**E-R MODEL**



*Larger version in ERMODEL.pdf*

**Relational Database Design**



**3NF Solution**

**Conversion to NF3**

1

* + Email ( message\_id, content, date, sender\_powon\_id, receiver\_powon\_id, sent\_visibility, received\_visibility, gift, gift\_content)
  + Public\_Post ( public\_post\_id, content, date, admin\_id)
  + Member ( powon\_id, username, password, first\_name, last\_name, address, email, dob, description, status, privilege, first\_name\_visibility, last\_name\_visiblity, email\_visibility, address\_visibility, dob\_visibility, description\_visibility, profession)
  + Member\_interests (interests ,powon\_id)
  + Group (group\_id, name, owner\_id, description)
  + Post (post\_id, content, date, author\_id, thread\_id, upload, upload type)
  + Thread(thread\_id, name, group\_id, author\_id)

2 Relation 1 – Member of group between member and group

Many members belong to many different groups. This represents a many to many relationship. We created a table member\_of\_group with foreign keys group\_id and powon\_id.

* + Member\_of\_group(group\_id, powon\_id)

Relation 2 – Member requests to join group between member and group

Many members request to join many different groups. This represents a many to many relationship. We created a table join\_request with foreign keys group\_id and powon\_id.

* + Join\_request(group\_id, powon\_id)

Relation 3 – Member is related to another member between member and member

Many members are related to other members. This represents a many to many relationship. We created a table member\_relates\_member with foreign keys powon\_id and relates\_powon\_id.

* + Member\_relates\_member(powon\_id, relates\_powon\_id, family, friend, colleage)

Relation 4 – Member has access to threads between member and thread

Many members have access to ceratin to certain threads other threads. This represents a many to many relationship. We created a table member\_relates\_member with foreign keys powon\_id and relates\_powon\_id.

• Member\_has\_thread\_access(powon\_id, thread\_id, restriction)

For each of these relations, it turns out the PrimaryKey determines all other information so they are all BCNF which includes being all in 3NF.

**Member Responsibilities**

Jason:

* Design of database
* Add profile visibility features (private, group, public)
* Add interests/profession features
* Add admin reports page for user statistics
* Add relationships features (ability to add friends, family, colleagues)
* Add event creation page and functionality
* Add poll voting for event suggestions
* Installation Manual
* Edit/Delete admin post
* Server Administration

Sebastian:

* Design of database
* Login page and functionality
* Registration page and functionality
* Add active/inactive/suspended member features to site
* Add admin option to delete member
* Add admin option to change privilege/status for any member
* Create group and group functionality
* Add feature for member to send group join request
* Add feature for group owner to invite member to group
* Option for group owner to delete group and group member to leave group
* Templates for home page, public, group page
* UI layout

Daniel:

* Implement threads
* Implement posts
* Implement thread accessibility by members
* Implement ability for thread author to manage thread access
* User Manual
* Gantt Chart

Michal:

* User interface with CSS Bootstrap
* Email pages and functionality
* Send an email to group members when new thread is created
* Create E/R model 3NF
* Add file upload for pictures and video to posts
* Add gift exchange functionality
* Add invite member to POWON functionality

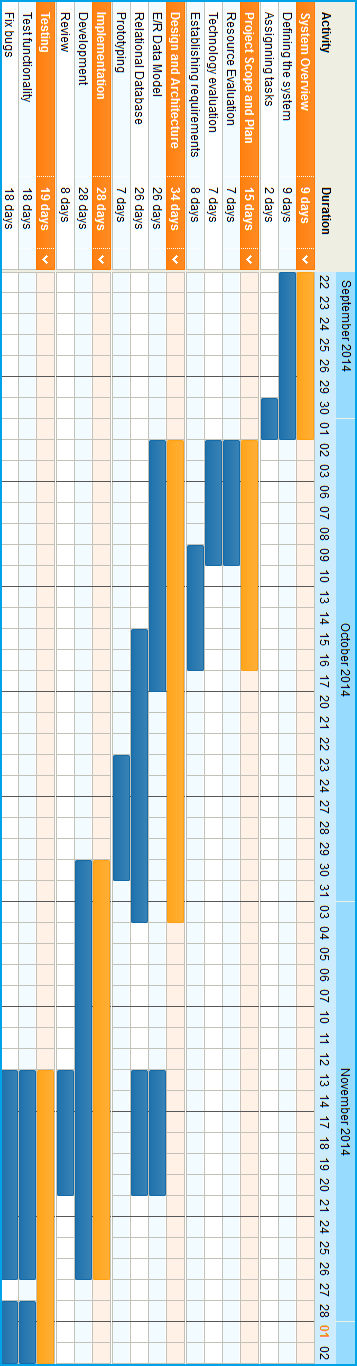
Francis:

* User interface design
* QA testing
* Bug fixing
* Note taking
* Documentation compilation & editing

Joint Contributions:

* Discussing technology to be used
* Discussing database design
* Information gathering
* Peer review
* Testing
* Bug Fixing
* Documentation

**Gantt Chart**

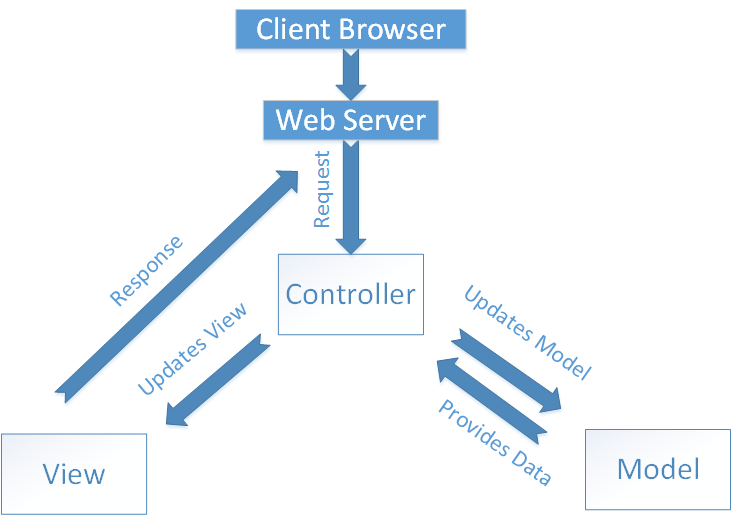
****

**Detailed Analysis of coding the Website**  
POWON is based on the model view controller (MVC) framework. Using CodeIgniter as our PHP framework we are able to separate the logic between our view, controller and model. This separation of logic helps simplify group work as all three sections can be worked on at the same time without interfering with each other. This makes it less difficult to test and solve problems to the related logic domain.

**The Model** handles data access, the business, and validation logic. The model will be retrieving from and storing data into the database. The handling of the interaction between our SQL database and the application is included here.

**The View** handles displaying the application data. Views are created from the model application data. The view will also accept user input in which the controller will handle.

**The Controller** handles user input and responses. The controller reads user input and handles the events from the view to the model sending the user input data to the model when appropriate. It will also validate user input and provide error handling messages.

****

Reference: http://betterexplained.com/wp-content/uploads/rails/mvc-rails.png

The MVC framework is fully integrated and supported in CodIgniter. All three domains are coded with PhpStorm and coded in PHP language. There will be integration of javascript, and jquery for the view. The view was further developed with knowledge of bootstrap  (framework using javascript, html, and css).

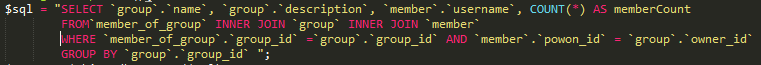
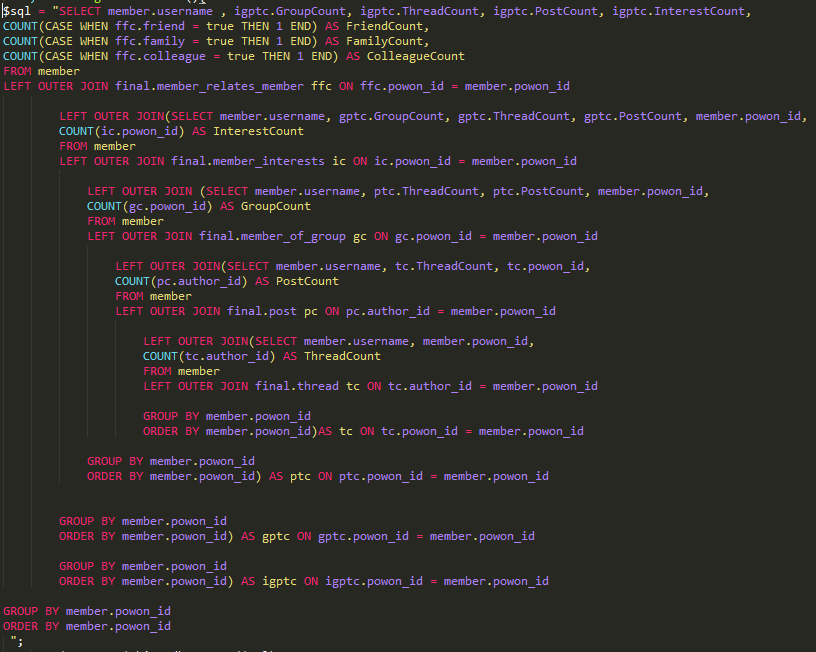
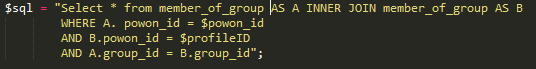
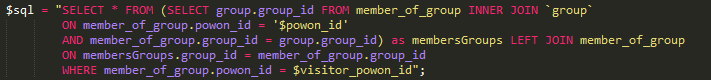
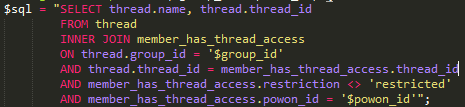
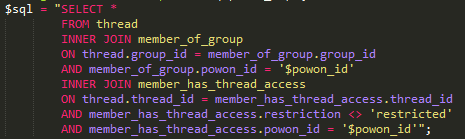
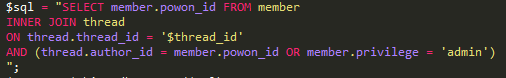
**List of Technologies Used And Rational:**

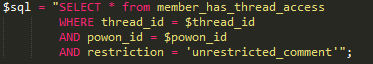
|  |  |
| --- | --- |
| **Technology** | **Rational** |
| **PhpStorm** | We chose the PhpStorm IDE because it was meant to be a lite application for the programming language PHP, and works well with managing classes and CodeIgniter MVC. |
| **PHP** | Required coding language. |
| **CodeIgniter** | Suggested framework. We chose it because we were familiar with the MVC pattern. It’s libraries were useful for database functions and access from our model, form validation, managing session information,  uploading files, managing the helper methods from the url and forms. It was simple and easy to use. Many video tutorials are supported for it. |
| **Bootstrap** | As a HTML, CSS, and JS framework, we were able to use it for the design of our views. It sped up the design process so we can share views between multiple pages at the same time. |
| **jQuery** | A prebuilt javascript library. Aided in visual animations, event handling, and pre built functions that help us design nicer views. |
| **MySQL** | Required relational database technology. |

**The Interface Design Rational**

Much of the interface model is based on the forum and Facebook designs. We wanted to make the user interface as simple and as user-friendly as possible. A user using our version of POWON will notice there are many drop down and radio buttons rather than direct input or image uploads as can be seen in the exchange gift option. Although our original design was to leave the user option to upload their own gift pictures, issues with the different file format pushed us to limit the gifts to select pre uploaded pictures only.

**Queries**

* Get every public post  
  + 
* Get group statistics  
  + 
* Get user statistic
  + 
* Get all received emails  
  + 
* Get the number of received emails  
  + 
* Get the status of the user we are trying to send a message.   
  + 
* Get all group member   
  + 
* Get all the group that the member is part of   
  + 
* Get the member’s profile information  
  + 
* Verify if the user can see the selected member’s profile information  
  + 
* Find is the user is suspended   
  + 
* Get the privacy of the group  
  + 
* Get all member that are part of your family
  + 
* Get all the users that requested to join the group  
  + 
* Check for any group restriction
  + 
* Get threads from all the members related groups, it will not show any thread where he is restricted   
  + 
* Check if user is an admin or a an author   
  + 
* Check if user can comment on a thread



**User Manual**

**WELCOME**

In the home page there are three options available to a visitor or returning member. From those three, register will bring the user to the register page, and login directs the user to the login page.

The top bar is accessible from any pages making the register and login option always in reach.

**VISITOR**

***REGISTRATION***

In order to register, a user has to enter an existing member’s first name, email, and date of birth in the format (yyyy-mm-dd). The user will then enter a username, password, first name, last name (optional), address (optional), email, date of birth in the format (yyyy-mm-dd), and a description (optional).

**USER**

***USER HOME PAGE***

Once a user as logged in, from his member options, he can view his profile, create a group, search a group, search members, view emails, or enter his privacy settings. The user can also see public posts from administrators, all the groups the user belongs to, and the threads from all their groups. The user can enter a group or a thread by clicking on them. The top bar, available to every page while logged in, holds the links to the user’s home page, profile, and email to the left. To the right of the top bar is the users username, which when clicked gives the option to log out.

**VIEW PROFILE (USER’S OWN PROFILE)**

Once a user as clicked on “View Profile” from the user’s home page or “Profile” from the top bar, they will enter their profile page.  The profile is composed of his user information, relationships, professions, and interests. In the update relation section the user can select a member from the dropdown box, and put yes or no to each corresponding relation below. When the update button is clicked, that member will be added to the user’s chosen relations. Under each relationship there is a list of members that belong to it. The user can click on a member name to be redirected to that members profile page, where he can view their profile information.

The user can also go to the profession and interests section where they can enter a profession or interest. When clicking submit the professions and interests will be updated.

**VIEW PROFILE (ANOTHER MEMBER’S PROFILE)**

Once a user has clicked on a member name from their home page or group page, they will be redirected to that members profile page, where he can view their profile information.  The profile is composed of that member’s information, and their relationships. Under each relationship there is a list of members that belong to it. The user can click on a member name to be redirected to that members profile page, where he can view their profile information.

**VIEW EMAIL**

Once a user has clicked on view email from their member options in their home page, they will be redirected to the email page, where he can view, send, and receive, emails.

**SEARCH MEMBER**

From the user’s home page, the user must go to the member options section and choose

“Search Member” option, where he will be redirected to the search member page and see all members of POWON and their member information.

**SEARCH GROUP**

Once a user has clicked on “Search Group” from the user’s home page, he will be redirected to the group search page. The user will see a list of all groups and can make a request to join a chosen group by clicking on the “Send Join Group Request” button. When clicked it will send a request to the group owner where the owner can choose to accept or reject the request. If accepted, the user will be a member of that group.

**CREATE GROUP**

Once a user has clicked on “Create Group” from the user’s home page, he will be redirected to the group creation page. In the group details section, the user can input the group name in the provided text box, and input the group description (optional) in the text box below. Once the information has been entered the user will click the “Submit” button where the group will be created and the user will be redirected to the new group page.

**GROUP PAGE**

Once a user has clicked on a group name from the user’s home page, he will be redirected to that group page. In the group page the user will see the group info, threads, and members belonging to the group. The user can enter a thread or view a member profile by clicking on them. From the group options section the user can create a group. If the user is the owner of the group then he will also see the owner’s option section. In the owners option section he will see links to view join requests, invite members, or delete this group. Clicking on the links will redirect the user to their respective pages.

**CREATE THREAD**

Once a user has clicked on “Create Thread” from a group page, he will be redirected to the thread creation page. In the thread details section, the user can input the thread name in the provided text box, and input the post content in the text box below. Below the post content text box the user can enter public or custom restrictions. If public, once the information has been entered the user will click the “Submit” button where the thread will be created and the user will be redirected to the new thread page. If custom, the thread will be created and the user will be redirected to the thread access page.

**THREAD ACCESS**

Once a user, who is a thread owner, has clicked on “Thread Access” from a thread page, he will be redirected to the thread access page. In the Update Restrictions section the user can choose a member from a drop down list and change the member’s restriction level to unrestricted, no comments, or restricted. When the user clicks update, the member restriction level will be updated and the thread access page will refresh.

**ADMIN**

The admin is a member that also has access to all thread owner options for every thread and can edit or delete members from POWON.

**ADMIN REGISTRATION**

In order to register with the administrator privileges, the original admin has to add administrator privileges to the user from the “Edit/Delete Member” page which is accessible from the admins home page.

**EDIT/DELETE MEMBER**

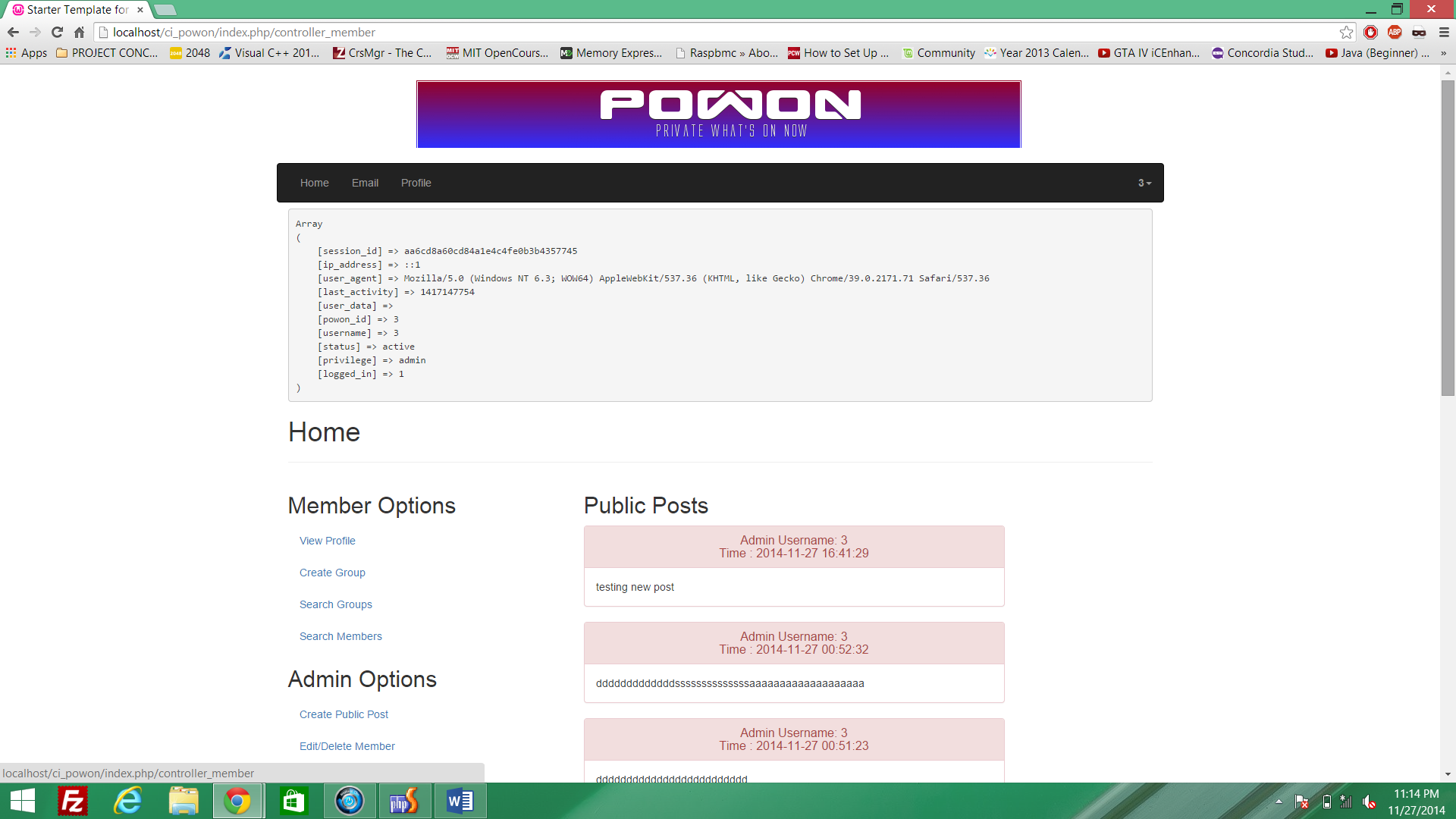
From the admin home page, the admin must go to the admin options section and choose “Edit/Delete Member” option, where he will see all members of POWON, their member information, update any member’s privilege to admin or member, status to active, inactive, or suspended, or the user can delete that member.

**CREATE PUBLIC POST**

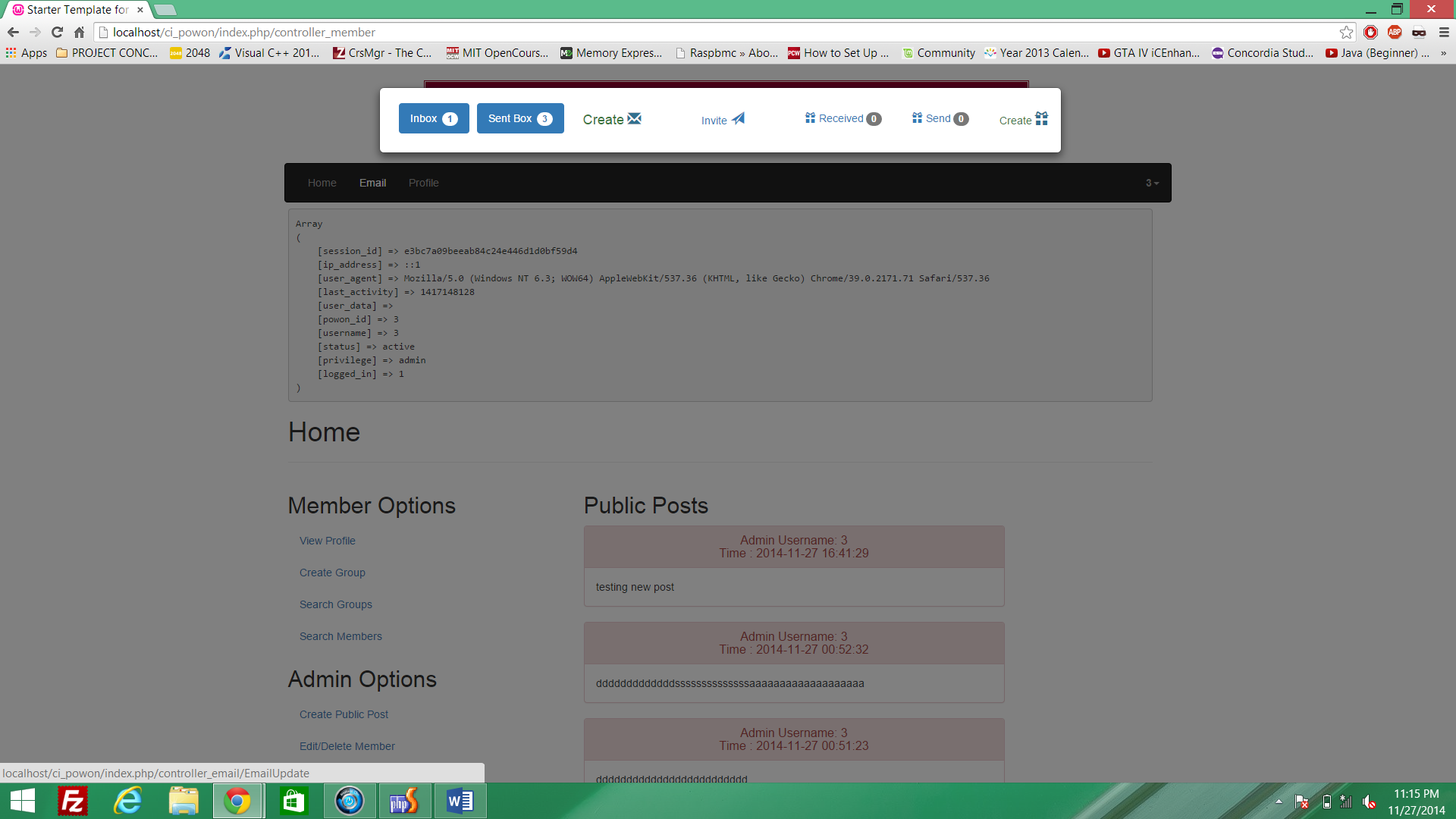
Once an admin has clicked on “Create Public Post” from the admin home page, he will be redirected to the public post creation page. In the content section, the user can input the content of the public post in the provided text box. Once the content has been entered the user will click the “Create Public Post” button where the public post will be created and the user will be redirected to the admin's home page.

**Website Demo**

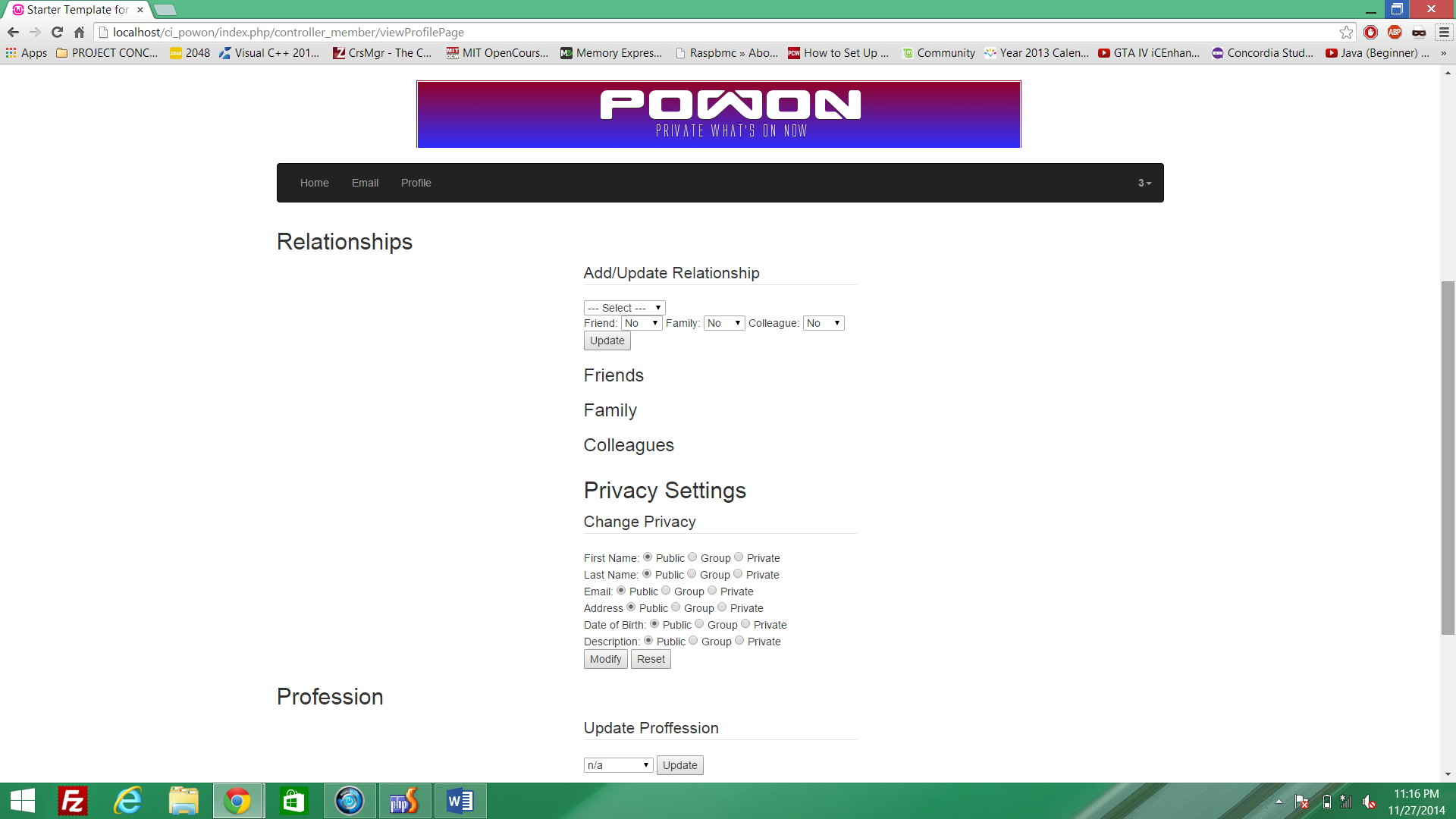
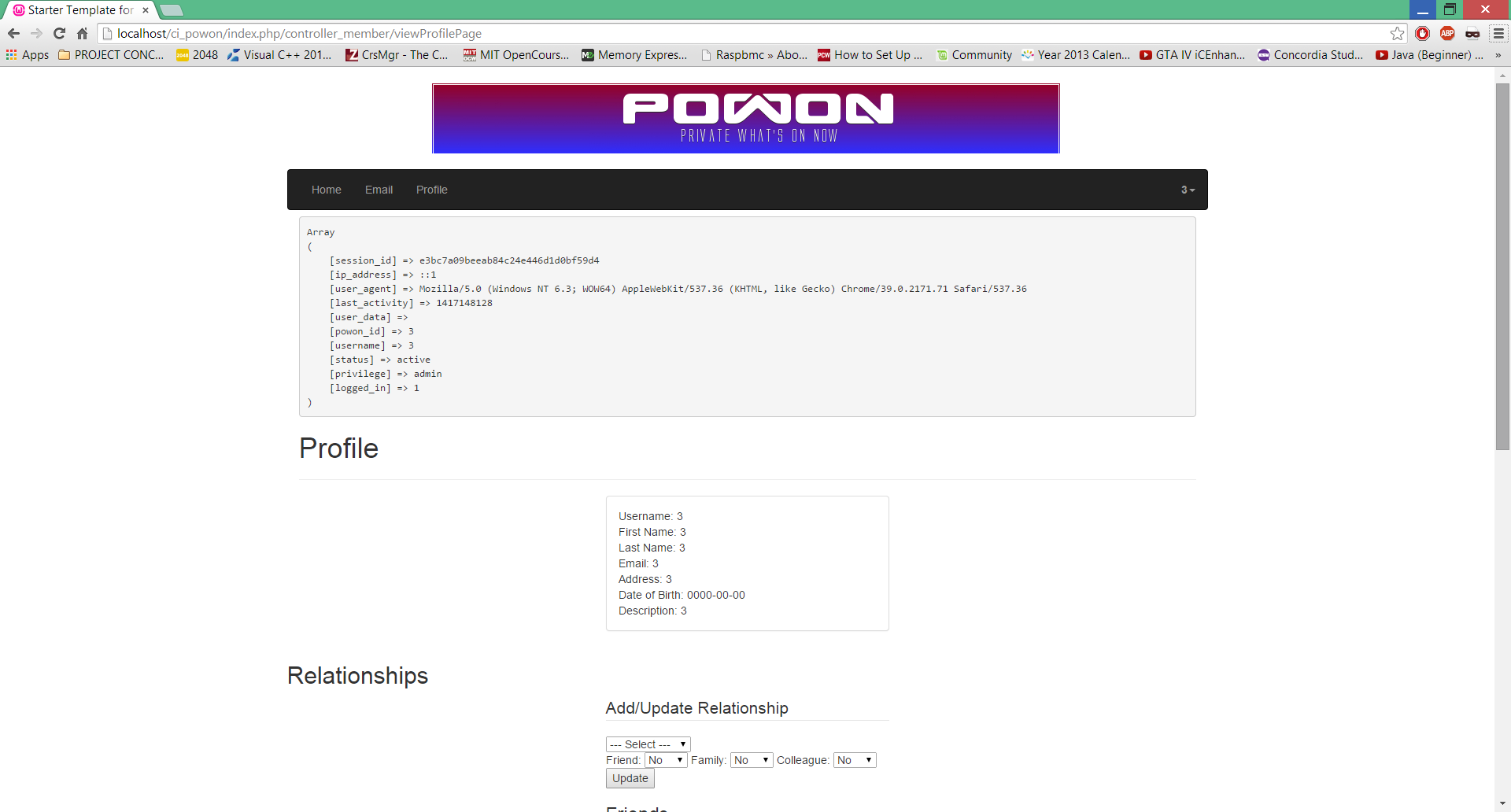
*Home Page (user is logged)*



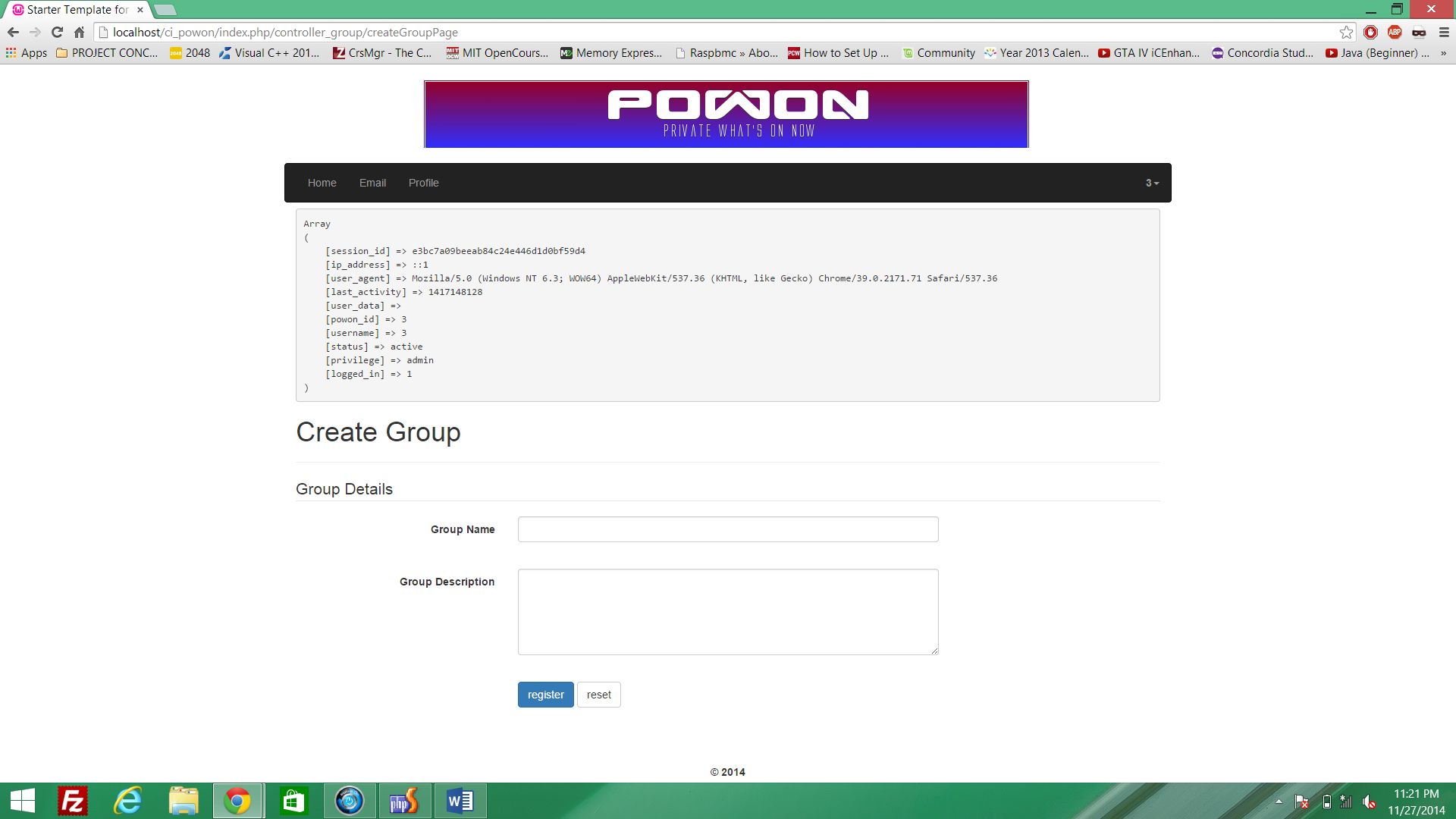
*Email Popup*



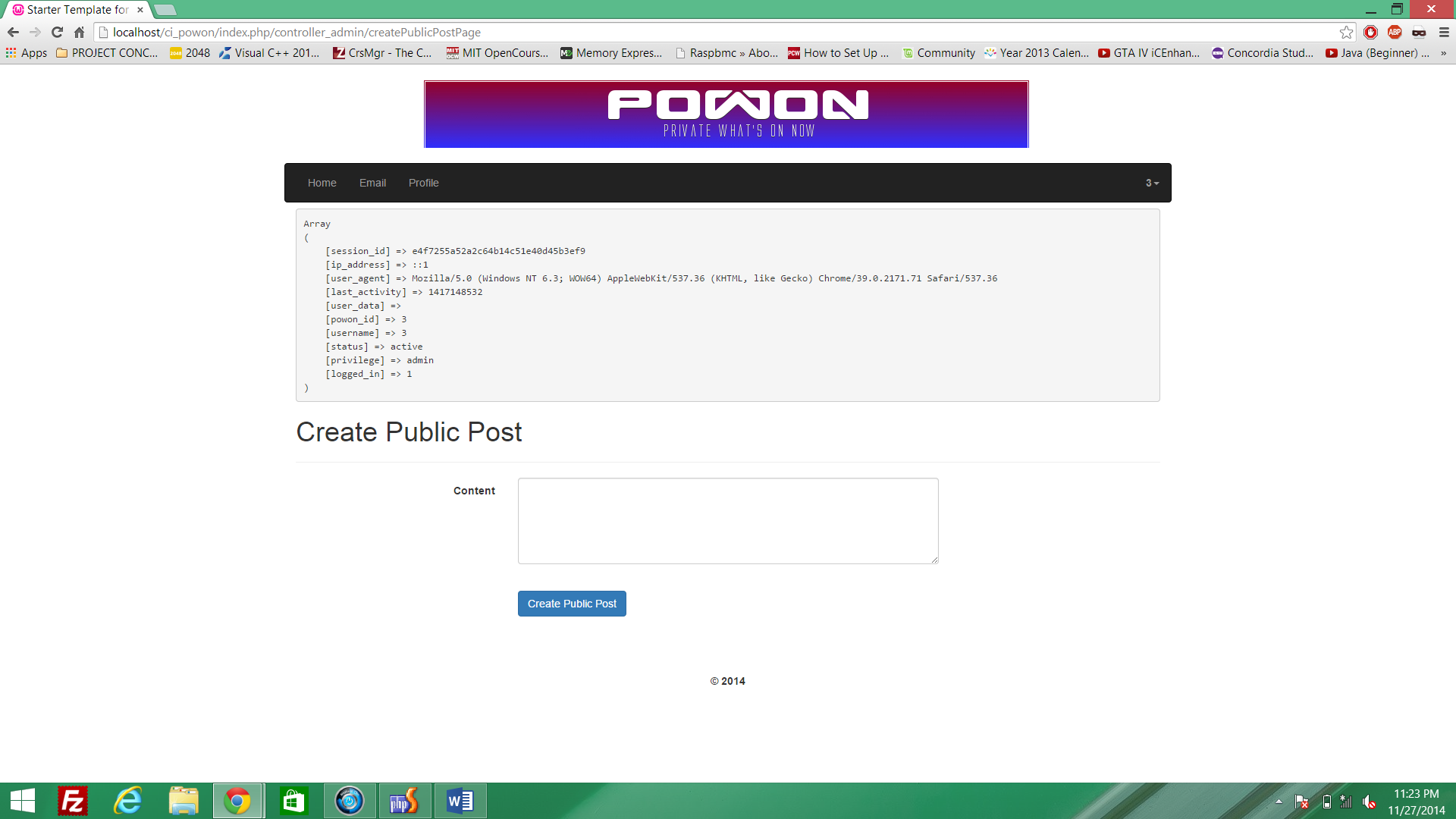
*Profile (user can set his privacy setting)*



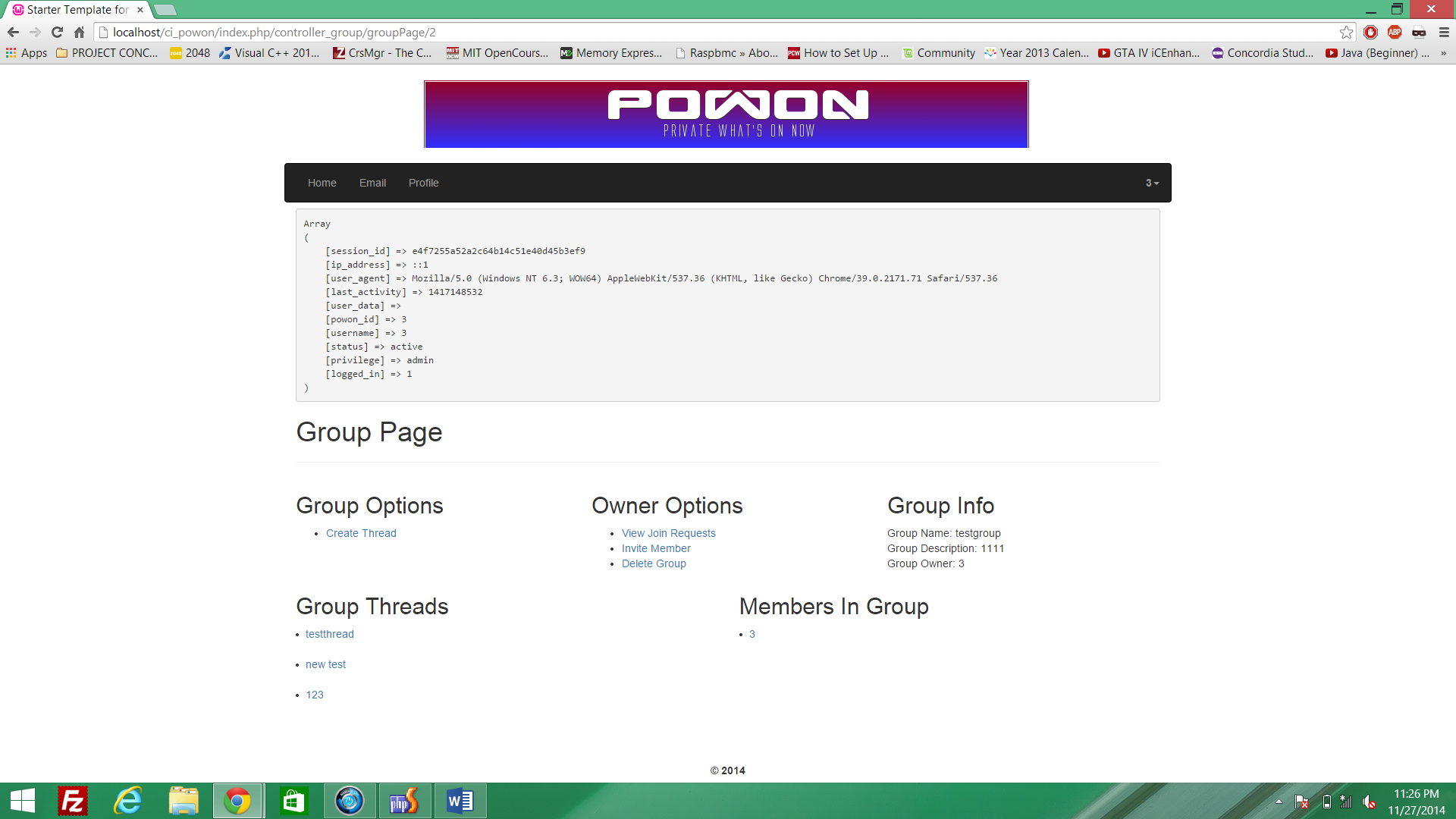
*Create Group*

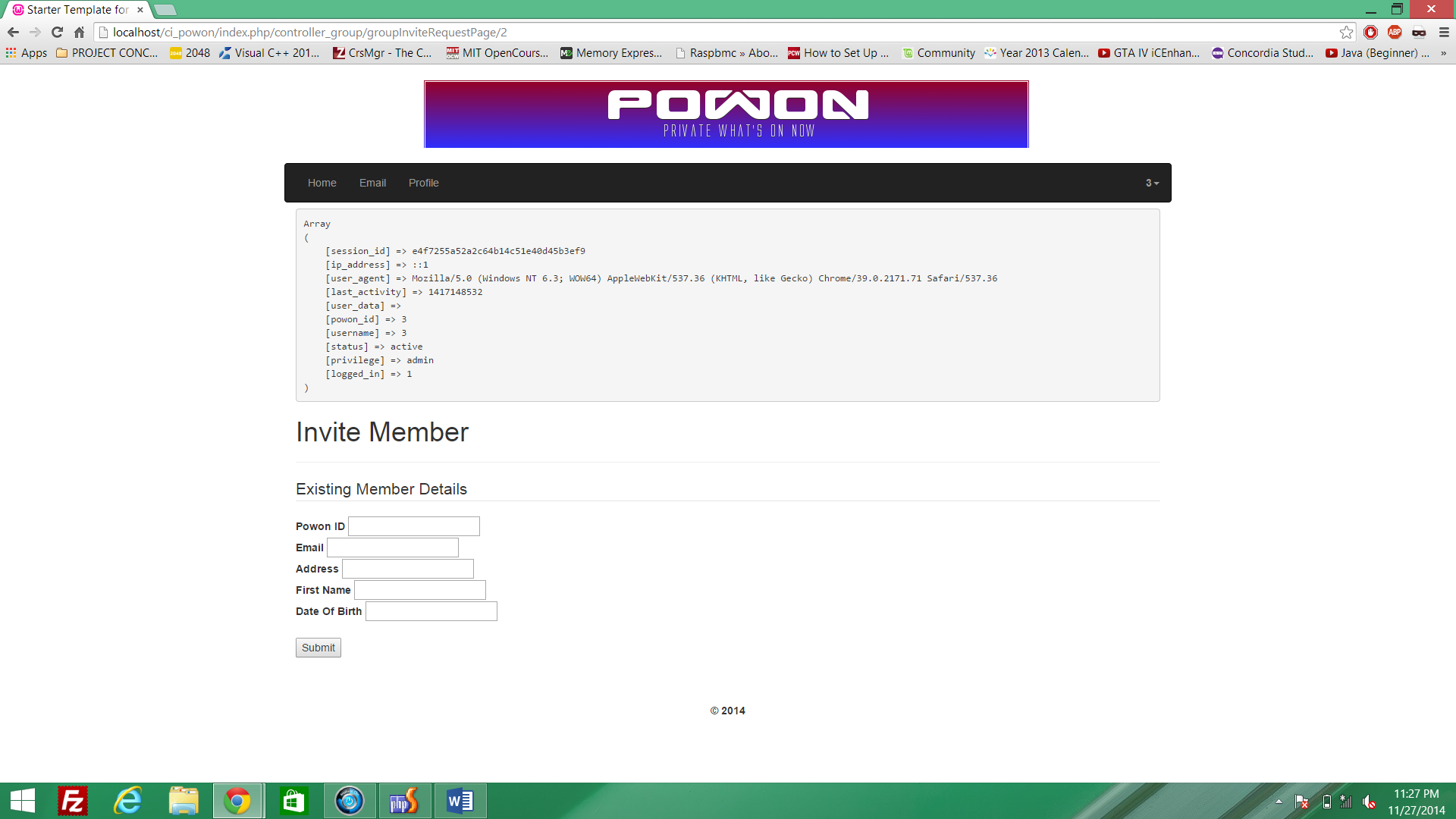


*Admin (edit/delete members)*



*Group Page Options*

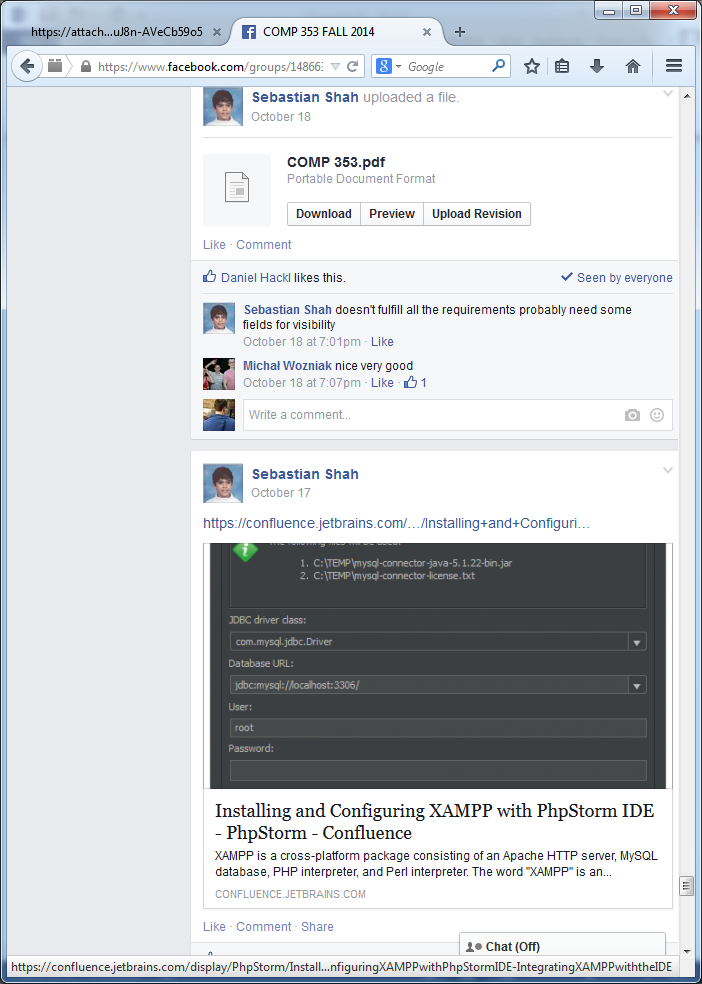
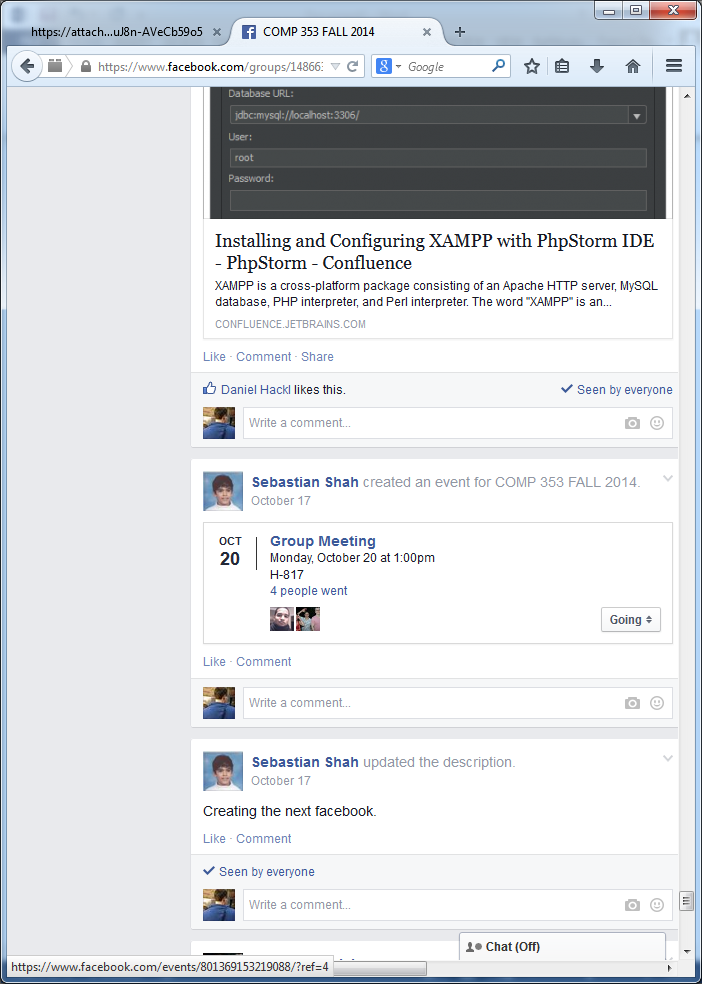
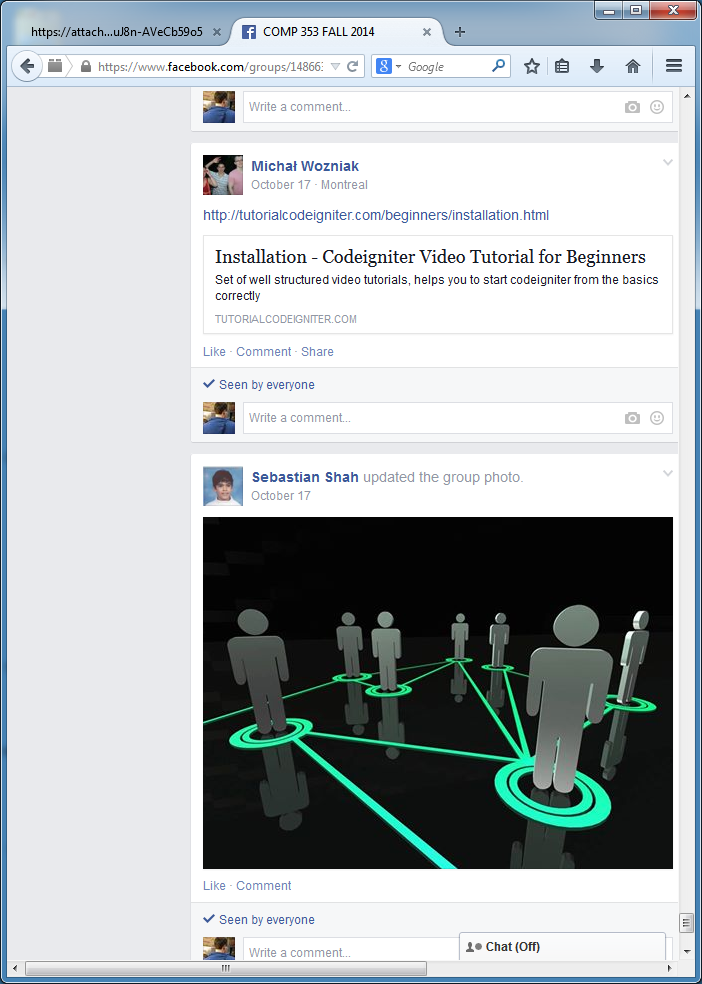


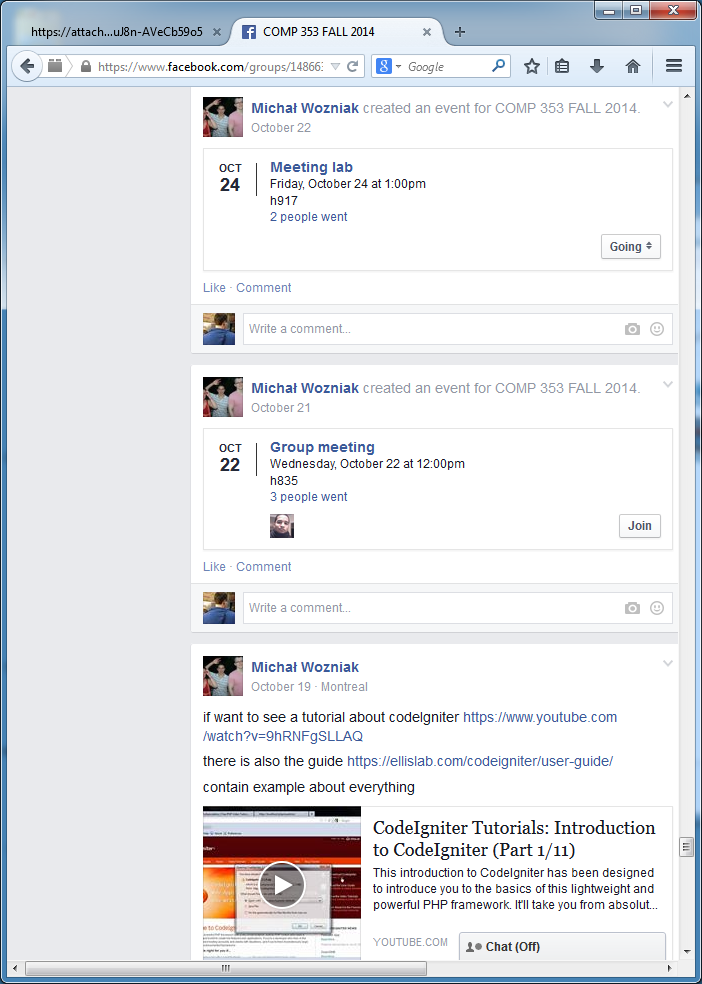
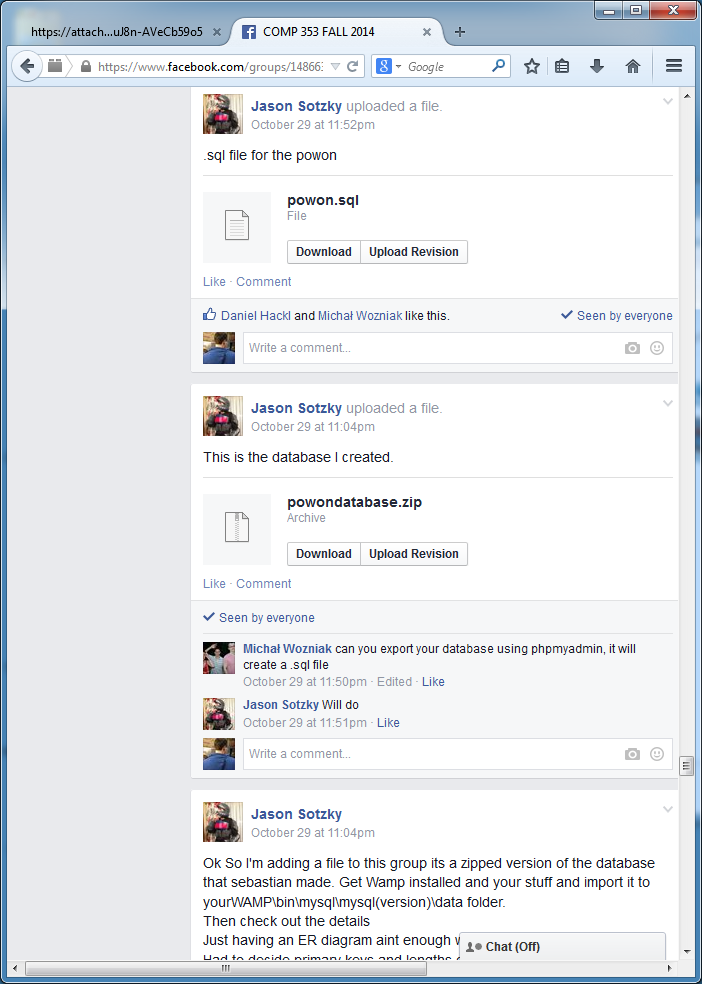


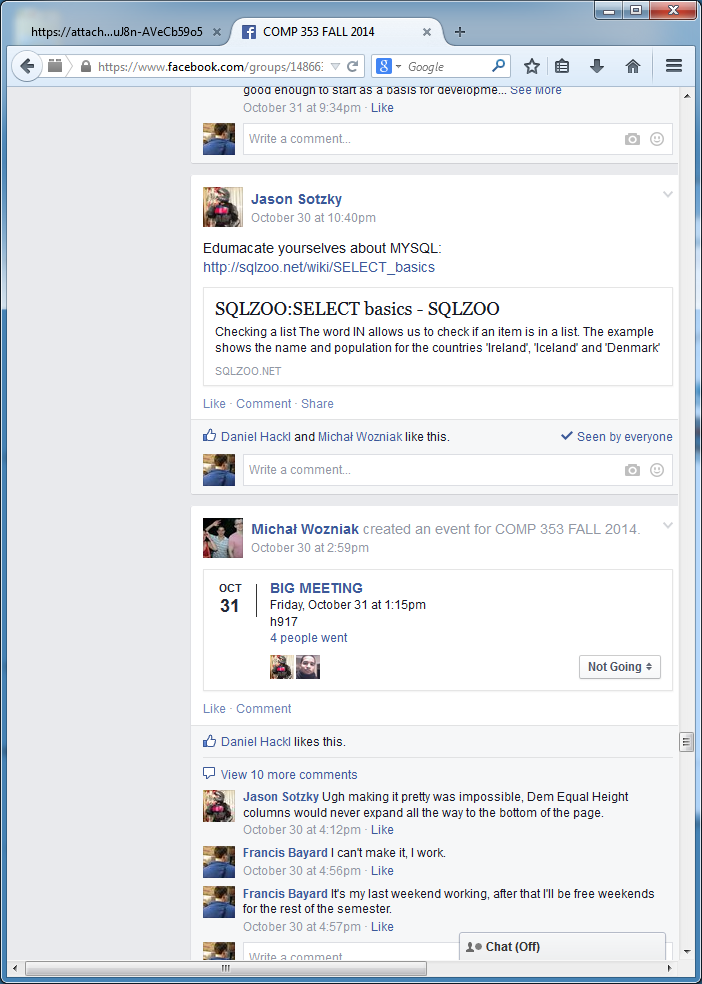
**Installation Instructions:**

1. Set up your SSH file transfer to the following.
   1. hostname: computation.encs.concordia.ca
   2. logingname: Your\_ENCS\_LOGIN
2. Put the code files of ci\_powon into the /nfs/www/groups/l/lf\_comp353\_2 directory.
3. Load up SSH Shell.
   1. Log in to SSH shell by using
   2. hostname: computation.encs.concordia.ca
   3. logingname: Your\_ENCS\_LOGIN
   4. Then get yourself to the /nfs/www/groups/l/lf\_comp353\_2 directory.
   5. type chmod -R 755 ci\_powon (This changes necessary file permissions)
4. In SSH Shell we load the database use SSH file transfer to bring the database into the /nfs/www/groups/l/lf\_comp353\_2 directory.
5. In SSH Shell we will type the following command to load the final.sql database into the lf\_comp353\_2 database.
   1. mysql -ulfc353\_2 -ptD3K6k -hclipper.encs.concordia.ca lfc353\_2 < final.sql
   2. This will load the lfc353\_2.sql database into lfc353\_2 database on clippers.encs.
6. Connect the site from:
   1. https://clipper.encs.concordia.ca/cgi-bin/cgiwrap/lfc353\_2/ci\_powon/index.php

**TEAM participation using Facebook group**

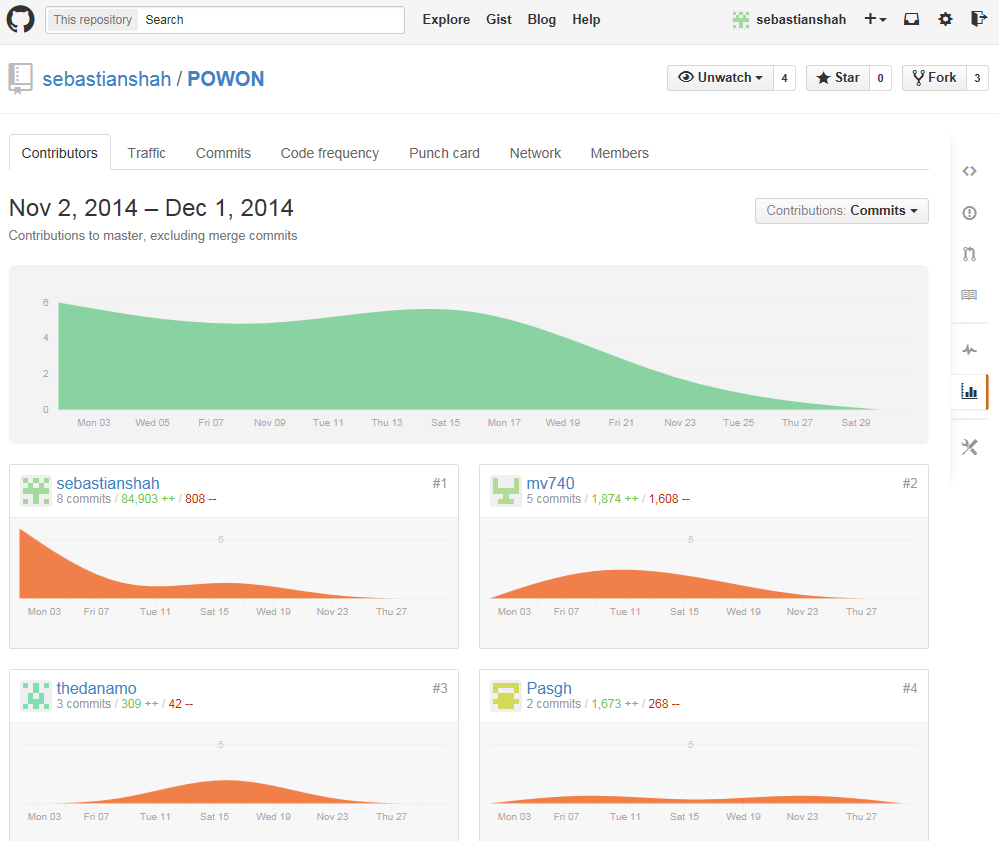






More screenshot in *group353\_screenshots.docx*

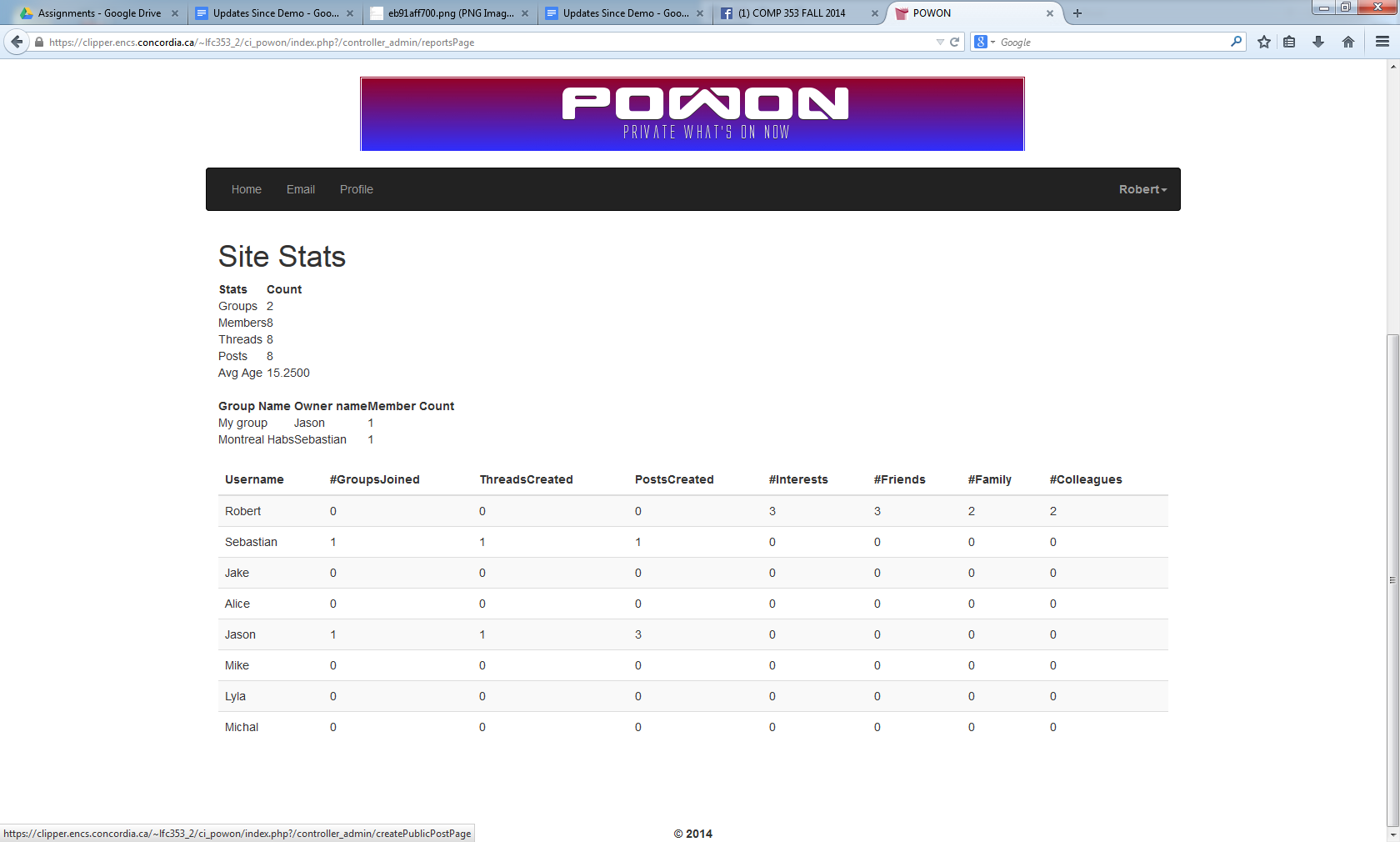
**GIT Log**

****

**Improvements since Demo**

* Updated Reports Page with more SQL queries that are actually relevant.Changes to the files:view\_reports.php, admin\_controller.php, model\_admin.php
* Update Public Posts to allow Admins to edit them at their descretion. The file added were: view\_edit\_public\_post.php, Files Changes were: controller\_admin.php, model\_admin.php, view\_home.php
* Updated the Email system so Admins can email everyone. Although members can only email their Friends, Relatives and Colleagues, or Members of groups that they are in. Files changed: view\_createEmail.php, Controller\_email.php, model\_email.php.
* Created and made a system to create Events for groups and Then Added Pools to these events for the place and time. The members can add suggestions to vote on and can only vote on one Suggestions. The Owner of the Group can close the poll and it will display the Play and Time of the Most voted on Event.
* Added checks to the database so users can’t create a member with an age before 1900-01-01 and added checks for event creation so user has to create suggestion after current date and time.
* The ability to edit public posts was added.
* The ability to edit and delete thread posts was added.

**Improvements since Demo (screenshots)**

**Added Statistics to Report page:**

**Newly implemented events page with voting:**