IAT 352 A4 Final Report

April 2, 2014

Danny Blackstock #301162355

Description of Project

This website project is meant to be used to provide better information about the SIAT program at SFU to visitors, especially prospective students. It aims to do this by connecting current SIAT students and alumni with high school and other prospective students.

Prospective high school students may appreciate seeing someone from their high school doing well, so this website enables prospective high school students find, "follow," and (optionally) contact current SIAT students and alumni (members) from their school. Posts from members appear in the visitor's "News Feed."

Alumni and current students in SIAT can post updates on their work or lives, add their contact information, and can also display their Flickr and Twitter updates.

List of features

Anyone on the site can:

- browse for members via their names or high schools attended
- search through the database for members and posts
- see members' details and posts
- toggle showing tweets or posts on a user's page

Member accounts are able to:

- register with the website, by providing their name, email, phone, high school they attended, high school graduation year, bio
- optionally provide their phone number, if they prefer to be contacted by phone, upload a profile photo, and enter Twitter and Flickr usernames
- post short (text) blog-style messages about their experiences in SIAT

- display tweets from their Twitter account on their page (tweets are refreshed automatically with AJAX and a custom PHP generated XML response, then added to the user's page, every 7 seconds)
- display images from their Flickr account on their page

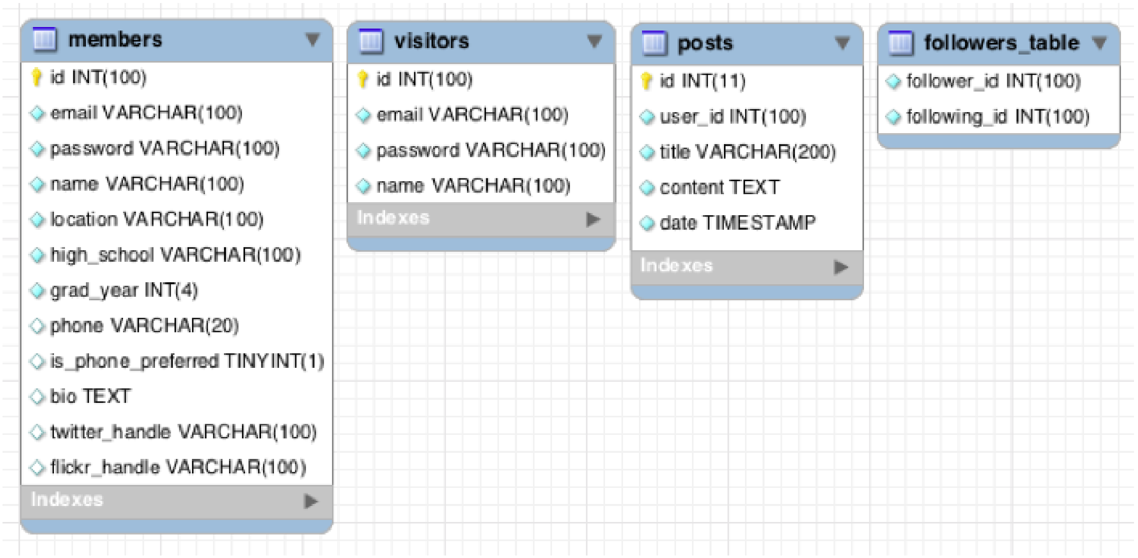
Visitor accounts are able to:

- register with the website, by providing their name, email, and password
- select members to follow
- view activity from members they follow on the site through a "News Feed" page

---

Database

I decided to have two types of users for my SIAT Outreach project, “visitors” and “members.” They each have their own table, because visitors have much less attributes than members (fig.1). For example, members have optional fields that they can fill out, including their Twitter and Flickr handles, which are used to show their content from those sites on their page.



```
members
id INT(100)
email VARCHAR(100)
password VARCHAR(100)
name VARCHAR(100)
location VARCHAR(100)
high_school VARCHAR(100)
grad_year INT(4)
phone VARCHAR(20)
is_phone_preferred TINYINT(1)
bio TEXT
twitter_handle VARCHAR(100)
flickr_handle VARCHAR(100)
Indexes

visitors
id INT(100)
email VARCHAR(100)
password VARCHAR(100)
name VARCHAR(100)
Indexes

posts
id INT(11)
user_id INT(100)
title VARCHAR(200)
content TEXT
date TIMESTAMP
Indexes

followers_table
follower_id INT(100)
following_id INT(100)
```

fig. 1 - A diagram of the tables and attributes they contain in the database. Members have a lot more information stored than visitors. (Screenshot taken by Danny Blackstock, 2014)

In order to model the relationship where visitors follow members (fig 1), I created a "follower"s table that contains rows of follower IDs (visitors) and IDs of members being followed. This was to allow me to query the database to create a "News Feed" page for visitors to see posts from all the members they follow.

Members' IDs are used heavily throughout the site structure. Posts are connected to a member's ID, and have a date attribute that is used to sort them when mixed with a user's tweets. Members' profile pictures are also linked to their IDs, so when a user uploads a new profile picture, it is renamed to their ID number. Each user's page is also linked to their ID.

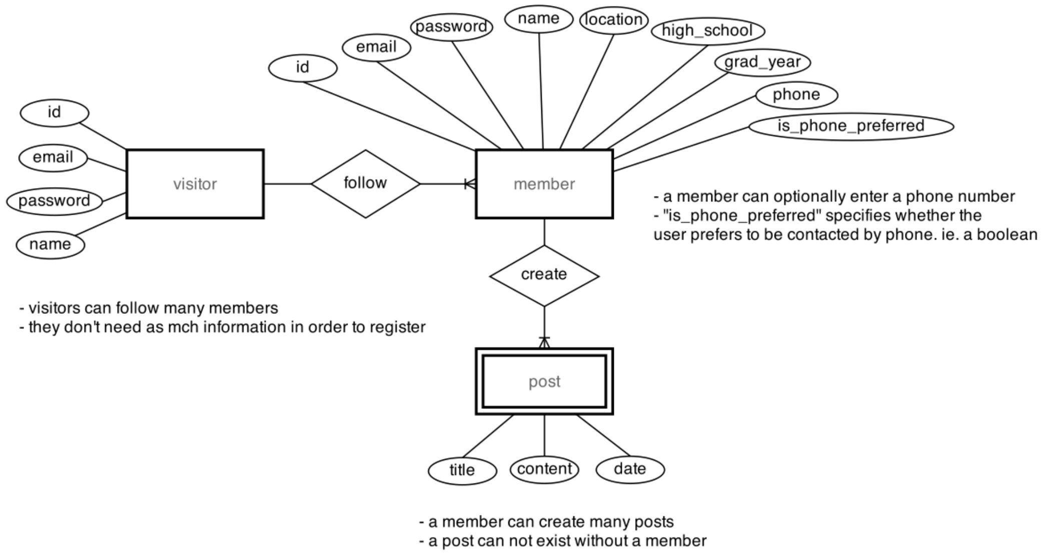

id
email
password
name
visitor
follow
member
id
email
password
name
location
high_school
grad_year
phone
is_phone_preferred
- a member can optionally enter a phone number
- "is_phone_preferred" specifies whether the
user prefers to be contacted by phone. ie. a boolean
create
- visitors can follow many members
- they don't need as mch information in order to register
post
title
content
date
- a member can create many posts
- a post can not exist without a member

fig. 2 - A ER diagram of an early version of the database. (Screenshot taken by Danny Blackstock, 2014)

PHP

My website's homepage gives users a list of the members on the site, so anyone can start exploring the content of the site quickly. They are also given the option to view members sorted by high school instead of alphabetically, so visitors can easily find people who went to their own school.

Each user’s page displays their information and posts, as well as their recent tweets and Flickr images if they have supplied their account names. The user’s most recent 10 photos from Flickr are displayed using Flickr’s REST API and XML processing. The user’s most recent 5 tweets are retrieved using the Codebird library to get a JSON file from Twitter’s API, then processed into XML through a custom PHP script. They are then parsed by Javascript and displayed intermixed with the member’s blog posts, sorted by date.

One feature I added for this assignment is the ability for members to upload their own profile photos. Members can do this by logging in, and navigating to the “edit_user_info.php” page, via the dropdown or from their own page. They can then choose a file to upload as their new profile image. Once they submit the form, if they have uploaded an image less than 500kb, it is renamed to their user ID (eg “1.png”) in PHP and stored in a special profile pictures folder (“img/profile_pics”). Whenever their profile picture is displayed, it retrieved using their user ID and either .png, .jpeg, .jpg, or .gif extension.

One of the most important parts of my page is the navigation, or top menu bar. A lot of code in the main menu bar files (ie. “main_menu_bar.php” and “main_menu_bar_https.php”) is for checking if the user is logged in via a session variable, in order to display different navigational options. For example, if the user is logged in as a member, a drop down menu appears allowing them to logout, view their page, or edit their account details. The reason their are two different menu bar PHP files is that “main_menu_bar_https.php” is to be displayed in SSL secured pages, like the sign up pages, login pages, and account editing pages, while the other is not. Furthermore, “main_menu_bar_https.php” does not reset the user’s callback_URL session variable, but “main_menu_bar.php” does. This makes is so that the user will be forwarded back to the page they were looking at before they logged in, signed up, or edited their account info.

The search bar, also part of the navigation bar, is available everywhere on the site. It allows users to search posts, and member details to find what they are looking for. It displays the results in a list, and also specifies what type of result each is.

Front-end and AJAX

The front end was designed to be responsive, that is, adjust its appearance depending on the size of the viewport, in order to support usability while maintaining functionality (fig. 3). I used the Bourbon Neat library to supply some basic CSS grid functionality, so certain elements like the user buttons on the home page could scale according to a grid. One thing I found when developing the front end this way was that in order for images to scale the way I wanted them, they had to be displayed using CSS background-images instead of <img> tags.

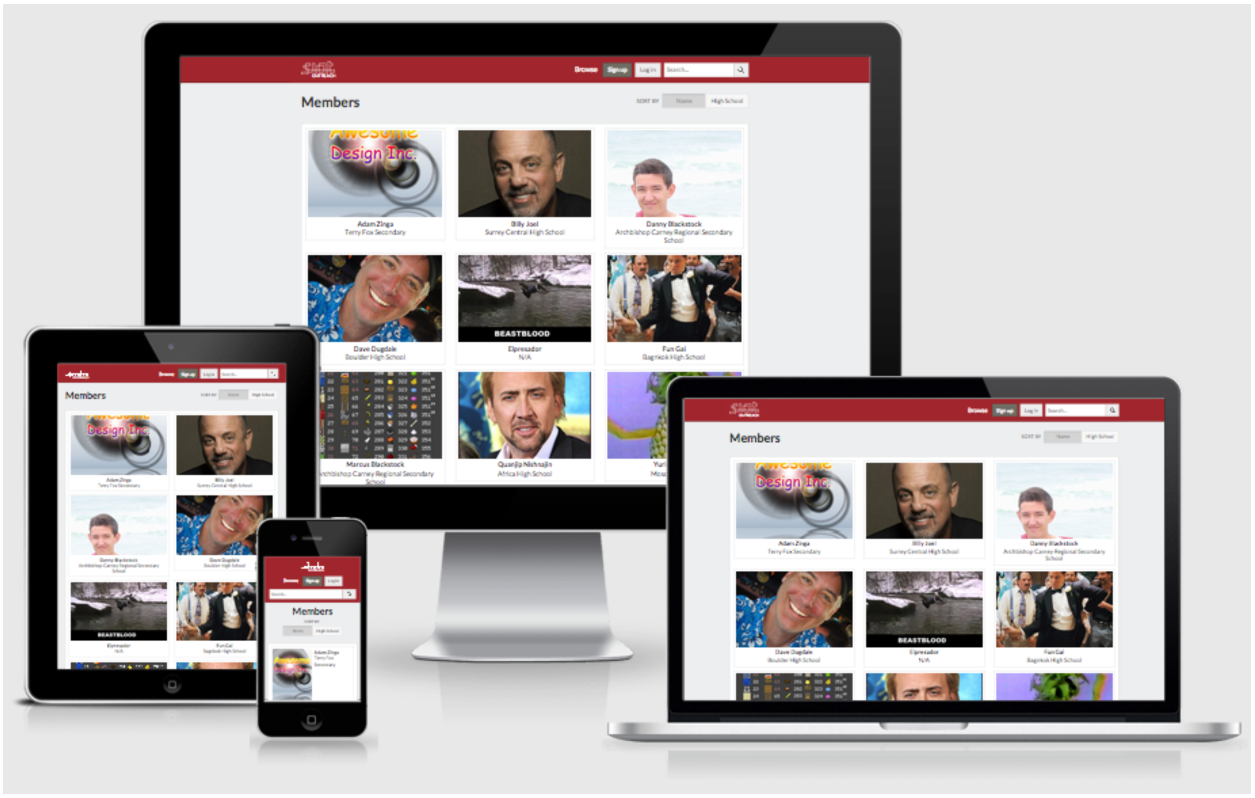


fig. 3 -The website design showing how it adjusts to different viewport sizes. (Screenshot taken by Danny Blackstock, 2014)

I used the JQuery library to support toggling tweets and posts on members' page. Any user looking at a member's page has the ability to toggle showing/hiding either tweets or posts from the feed by clicking two large toggle buttons on the top of the feed, which show/hide the posts using JQuery (see dblackst/js/main.js).

Flickr thumbnails are also displayed on the user's page in their own content container box. I chose not to mix them in with tweets and posts because Flickr is not used as often as Twitter, and they would also take up a large amount of space in the news feed on bigger monitors, compared to Twitter and posts, making the page very long, and other content seem less significant.

I use Javascript to call a custom PHP script that gets the most recent 3 tweets from Twitter every 7 seconds. I turn the JSON response from Twitter into a custom XML format, and return it to the Javascript using AJAX. The Javascript then parses my custom XML response, checks if any of the tweets are newer than the ones currently displayed on the page, and if so, adds them to the page. There are some limitations with my technique though. Firstly, if many users are visiting the site, my API key may be blocked for too many requests, since my requests always get 3 tweets and are timer based. Second, if a user tweets more than 3 times in 7 seconds, my code will only get the most recent 3. Finally, my script only adds tweets to the page if they were created at least a minute after the most recent tweet on the page, because I only display minutes, not seconds, on the page, which is what the Javascript use to determine if a tweet is new. I would aim to solve these in the future problems by caching tweets, storing more accurate time data and metadata about them, and seeing if there is a way to only retrieve tweets newer than a certain time.

Addressing previous feedback

Assignment 1 feedback:

"Project is well designed and code is structured in a very good way. However, there are some minor issues that should be addressed. There is no navigational menu, which makes navigation much harder and less intuitive. Once we insert new member, there is no navigation to “members” page. Field “Bio” allows inserting values over multiple lines, however, that is not handled appropriately. If user enters new line in this field, it causes wrong inserts in the file. It is not required to implement login form for the first assignment, but since it is provided, errors should be handled properly. In overall, very good assignment.”

In order to address this feedback, I added a dropdown so that logged in users can navigate to their own page quickly and easily. I also added support for new lines in the user’s bio section, as well as in blog posts. Finally, I implemented error handling in the back end for login forms, and basic HTML5 front end handling for user sign up pages. In the future, I would like to add another field for password confirmation when signing up.

Assignment 2 feedback:
“Good code, excellent design. All functionalities are implemented. Report should be better structured. While further improvements are well described, existing features should be better documented. It is not necessary to describe every page, however documentation should provide description for most important use cases for visitors and members. Tables and figures (in your case only the figure) should be labeled properly in a report, and referenced in text. Just as a note for further development: database design is good and covers all necessary functionalities, however, you did not provide an option for storing user images. Interface provides such a functionality, but database not.”

In order to address this feedback I tried to label and use my figures in my report more clearly this time, while giving the report a clearer structure. I also implemented the ability for members to upload their own profile photos. Members can do this by logging in, and navigating to the “edit_user_info.php” page, via the dropdown or from their own page. They can then choose a file to upload as their new profile image. Once they submit the form, if they have uploaded an image less than 500kb, it is renamed to their user ID (eg “1.png”) in PHP and stored in a special profile pictures folder (“img/profile_pics”). Whenever their profile picture is displayed, it retrieved using their user ID and either .png, .jpeg, .jpg, or .gif extension.

Assignment 3 feedback:
"New features are handled properly, and everything seems to work as expected. I had only one issue with mysql_real_escape_string, since it was giving a warning on most of the pages. Since you are using mysqli API, you should change this function to mysqli_real_escape_string"

This issue has been fixed for this assignment. I now use the object-oriented "$db->real_escape_string" function instead of mysql_real_escape_string.