University Gaming Association Event System and CMS Documentation

Travis Nickles

December 6, 2010

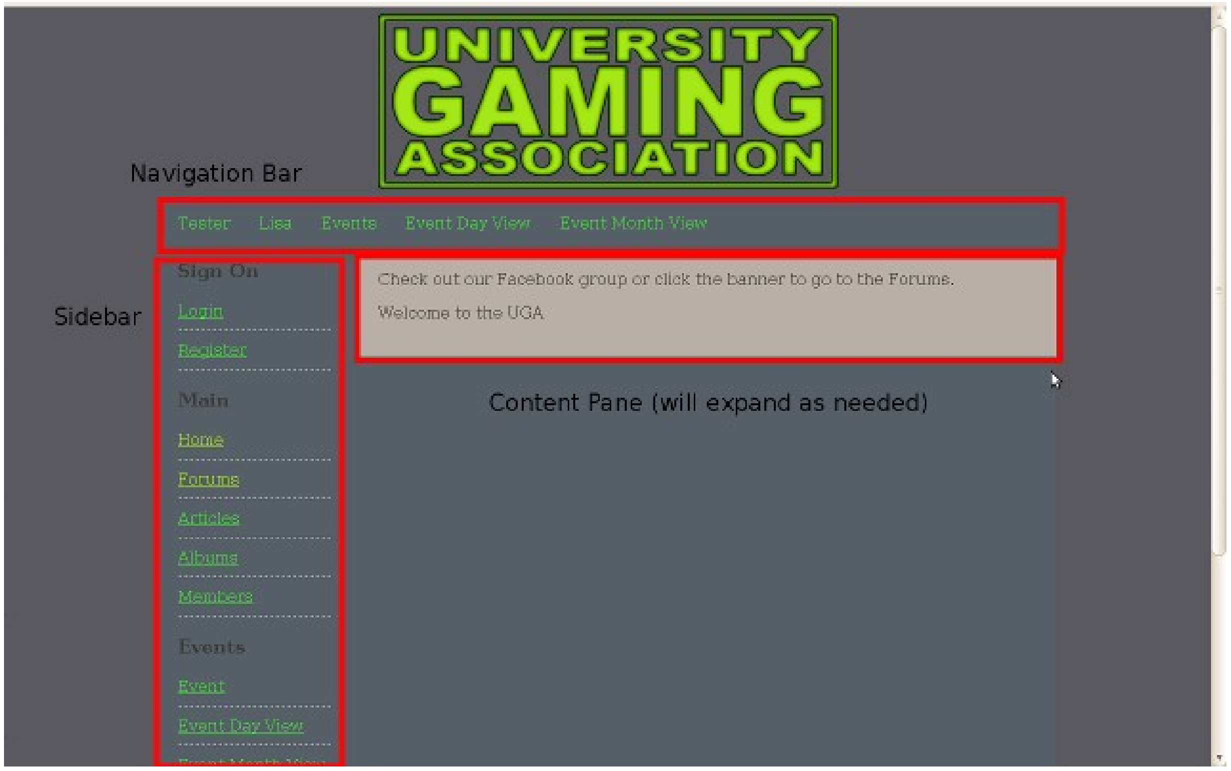
Table of Contents

1. Introduction to Web Site

Welcome to the University Gaming Association. The web site for our organization offers content to you in order to show the many activities that the group has to offer. To start using the web site, you will now become acquainted with the main screen.

1.1 Main Screen

The main screen will depict the primary layout of the web site. Each page will typically be split into multiple sections. The picture shows that the main portions of the main screen that are used to access the content of the site are the sidebar, the content pane to the right and the quick navigation panel on top. The top of the sidebar will change when you login to the web site. Also, when navigating the events portion of the web site, the sidebar changes above the main site navigation link section to show you specific content related to events. New links show up for the two kinds of feeds (RSS and .ics) for event details and also a “Filter Platform” section which can be used to filter page results by a choice of game type.


UNIVERSITY GAMING ASSOCIATION
Navigation Bar
Tester Lisa Events Event Day View Event Month View
Sidebar
Sign On
Login
Register
Main
Home
Forums
Articles
Albums
Members
Events
Event
Event Day View
Content Pane (will expand as needed)

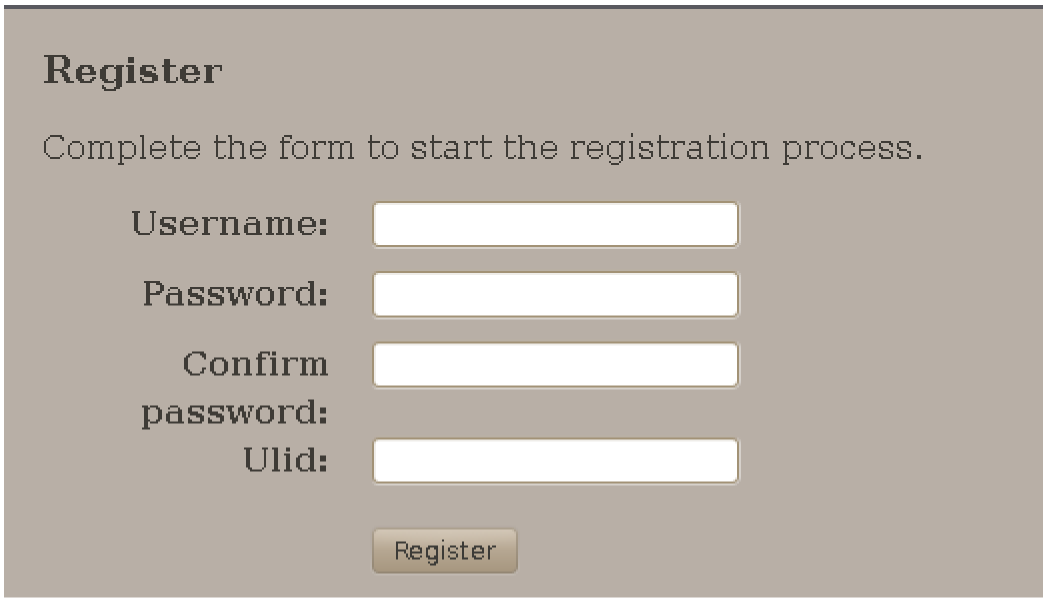
Check out our Facebook group or click the banner to go to the Forums.

Welcome to the UGA

1.2 Registration Process

By registering to the web site, you will be able to look at the details of other members of the web site including online gaming identities (Steam, Xbox Live, PlayStation Network, Wii), post an event to the web site and also mark yourself as attending an event. Registration takes a few steps but we hope that you find that registering to the web site is worth your while. The process involves you choosing a user name to identify yourself, confirming your status as an Illinois State University student via an email link, verifying the credentials you previously set and then awaiting approval from the administrator. Besides confirming your attendance at Illinois State University, we also hope that such a process will help curb spammer attacks.

To start with, click the "Register" link in the sidebar. The following form will appear:


Register

Complete the form to start the registration process.

Username:

Password:

Confirm password:

Ulid:

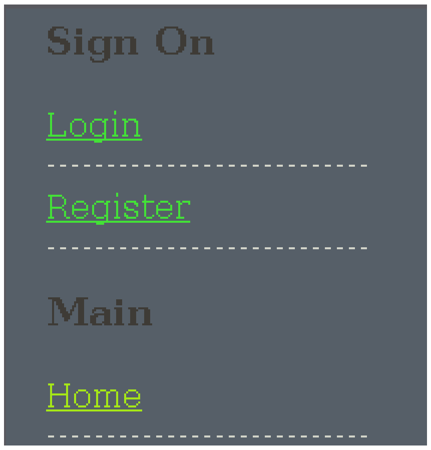
Registration Form

Fill in the form and submit to start the registration process. Firstly, if the user name you entered is already taken by a current user, you will have to choose a different user name; if somehow your ULID is already found, discuss the matter with the administrator of the web site. Next, you will receive an email with a message from the University Gaming Association and a link that you must go to within three days in order to continue the registration process; failing to do so will result in your user being deleted and you will have to start the registration process again. Once on the verify page, enter the previously specified user credentials in the form and submit it. If all goes well, your user will now be listed as needing admin approval and the administrator of the web site will have to approve of your user before you can login to the web site.

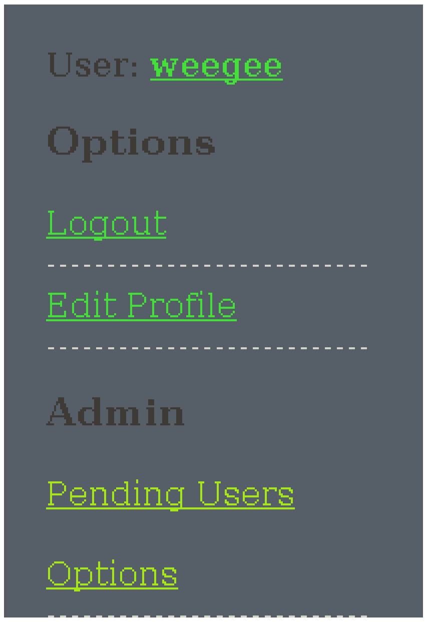
1.3 Login

Once your user is approved by the administrator, you can login to the web site and start using some of the exclusive functionality only available to members. The login form of the web site is not much different than the login for other web sites. However, we will offer you a brief description of it.

To access the login page, click the "Login" link at the top of the sidebar. From the login page, you can enter your user name and your password in order to confirm your identity. The sidebar of the web site will change to show that you are currently logged in. You can logout of the web site or edit your profile from the links at the top of the sidebar.



Top of sidebar when not logged in



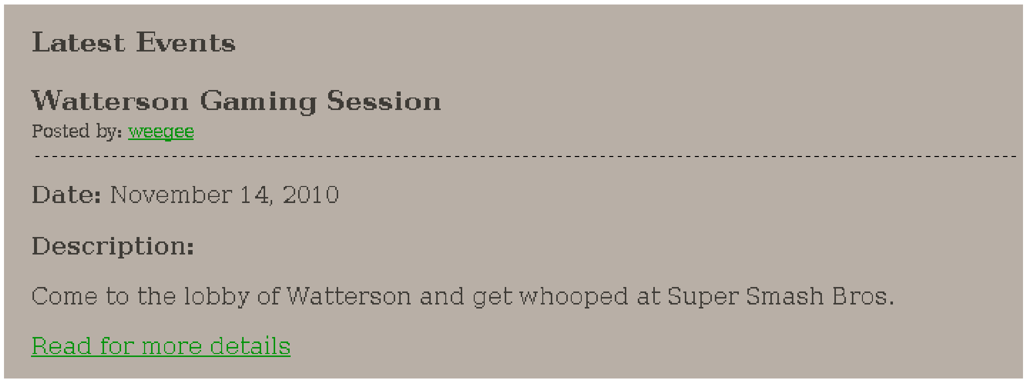
Top of sidebar when logged in as admin

2. Web Site Functionality

In this section of the documentation, we will introduce you to the many portions of the web site and the functionality of the pages. The heart of the web site is the event system so that is where this section will begin. Other pieces of functionality that will be discussed will be the member listings, article listing (news/blog) and the albums section.

2.1 Event Listing

The base Events page is a listing of all the latests events that will take place. Only approved events will appear in the listings. Click on the **Read for more details** link or on the event title to get more details about an event. If you are a member, you can mark yourself as attending an event by clicking the **Mark as Attending** button. To remove yourself as attending, go back to the event detail page and click the button that says **Remove from Attendance**.


Latest Events

Watterson Gaming Session

Posted by: weegee

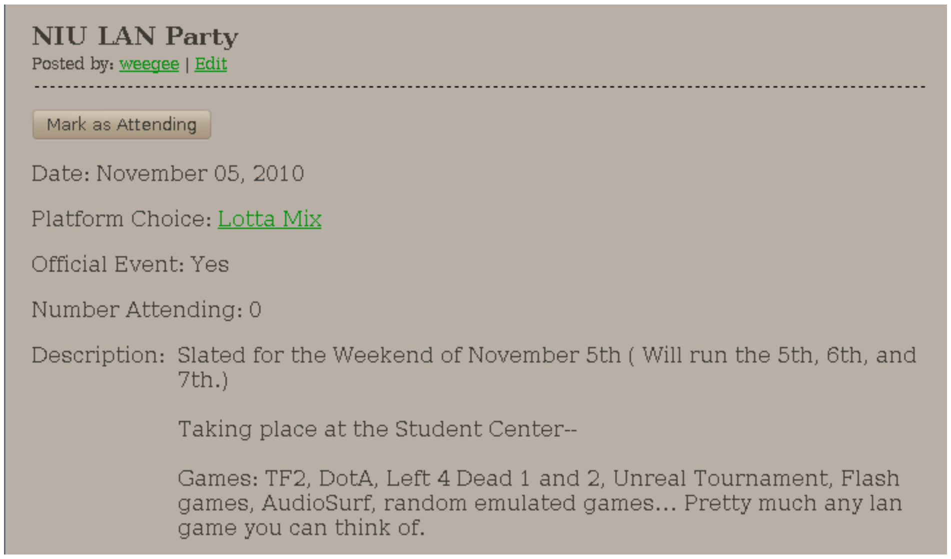
Date: November 14, 2010

Description:

Come to the lobby of Watterson and get whooped at Super Smash Bros.

Read for more details

Event Listing


NIU LAN Party
Posted by: weegee | Edit

Mark as Attending

Date: November 05, 2010

Platform Choice: Lotta Mix

Official Event: Yes

Number Attending: 0

Description: Slated for the Weekend of November 5th (Will run the 5th, 6th, and 7th.)

Taking place at the Student Center--

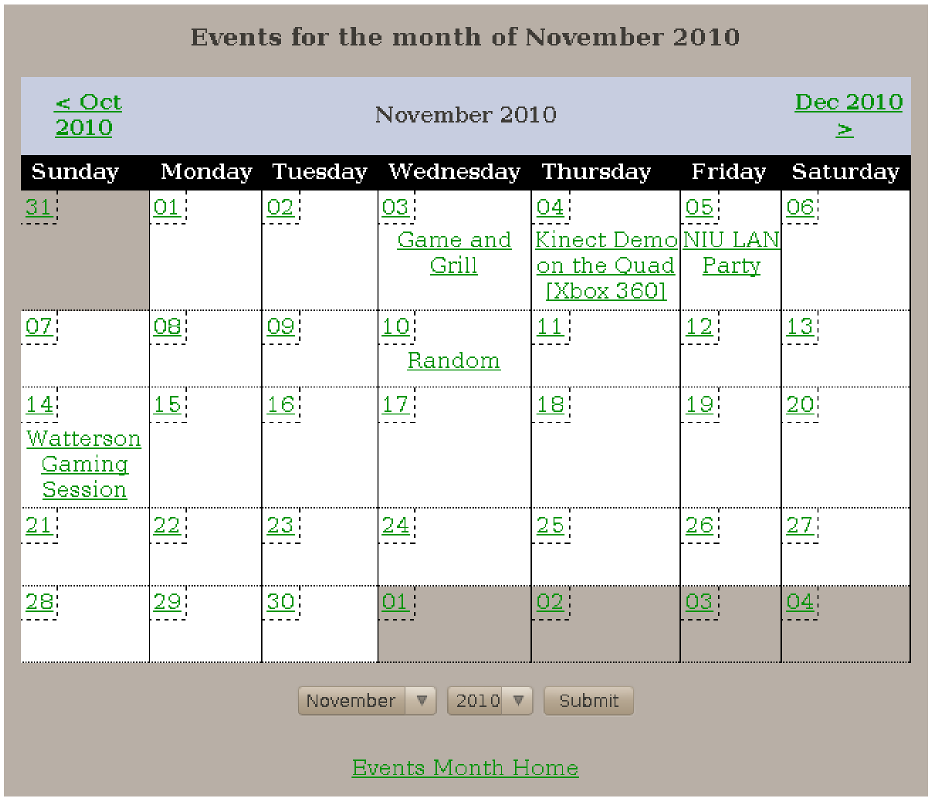
Games: TF2, DotA, Left 4 Dead 1 and 2, Unreal Tournament, Flash games, AudioSurf, random emulated games... Pretty much any lan game you can think of.

Event Detail Screen

2.2 Event Calendar

The “Events Month” section of the web site will show a calendar of all the events happening in the current month by default. The events will be displayed on the appropriate day in the calendar with multiple events being separated. Clicking on the title of an event will take you to the details page for the event.

Other aspects of the calendar are that clicking on the day number in the calendar will show a different view of the events occurring on a day. Previous month and next month links are found at the sides of the calendar title-bar. A navigation form on the bottom can be used to quickly check out the calendar view for a different month and/or a different year. Event on any calendar view can also be filtered by platform type by using the “Filter Platform” section in the sidebar.



Events for the month of November 2010

< Oct 2010 | November 2010 | Dec 2010 >

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
31	01	02	03 Game and Grill	04 Kinect Demo on the Quad [Xbox 360]	05 NIU LAN Party	06
07	08	09	10 Random	11	12	13
14 Watterson Gaming Session	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	01	02	03	04

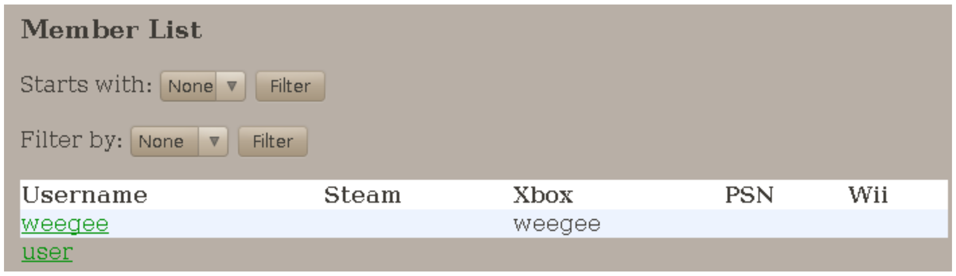
Month Calendar View

2.3. Event Feeds

Besides the listings show on the pages of the web site, you can also get information about the latest UGA events occurring by using an RSS feed reader or a Calendar application to subscribe to an events feed. When navigating any of the events page, the sidebar will have a section titled “Event Extras”. The available options are RSS and iCalendar. Both are read-only feeds.

2.4 Members

Members of the UGA like to be connected to each other with the many different online services available. The member listing on the web site is mainly used as a hub to share online identities so that people can friend each other and take further advantage of those services. Also, users can use the list and filter the results by which online identity that they are interested in; an example would be someone who only has a PlayStation Network identity can filter the results by PlayStation Network id in order to find other members playing a PlayStation 3 online.



Member List

Starts with: None ▼ Filter

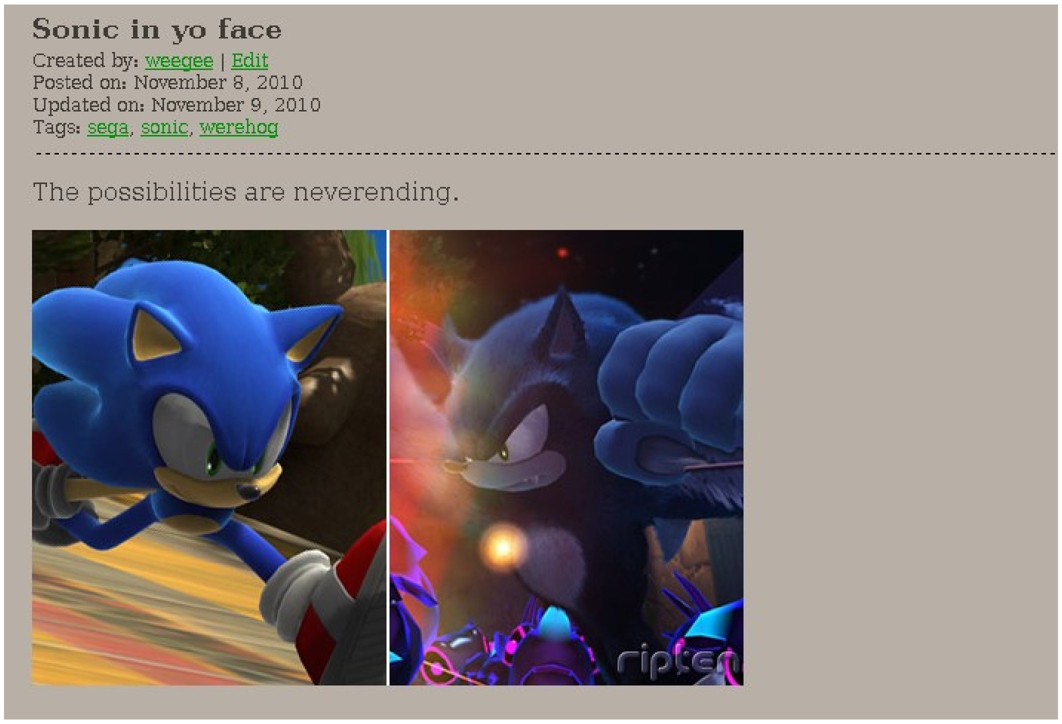
Filter by: None ▼ Filter

Username	Steam	Xbox	PSN	Wii
weegee		weegee		
user				

Online identifer listing

2.5 Articles

The Articles section will act as a news section or blog for the University Gaming Association. News will be posted about upcoming UGA meetings and other interesting gaming news. The articles can be tagged and clicking on an tag will allow you to see only articles that are posted with that tag. This is one way to pay attention exclusively to the news that interests you.



Sonic in yo face

Created by: weegee | Edit

Posted on: November 8, 2010

Updated on: November 9, 2010

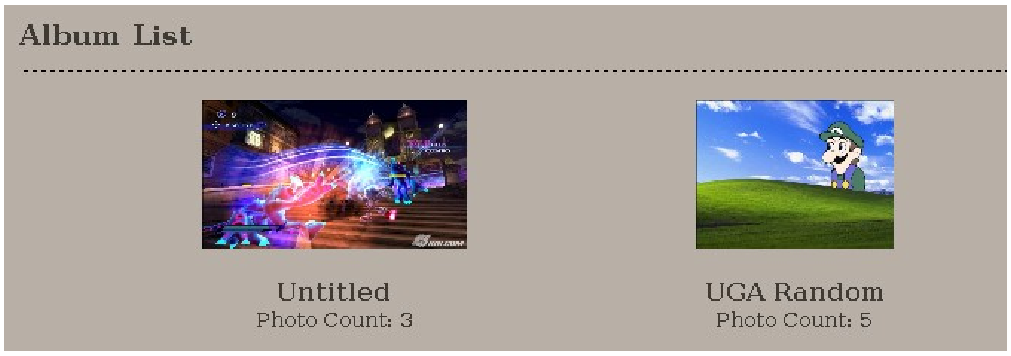
Tags: sega, sonic, werehog

The possibilities are neverending.

Showing an article

2.6 Albums

The Album section of the web site is useful for members to look at photos from previous University Gaming Association events. Each album has a collection of photos that can be browsed. The last uploaded photo of an album will serve as the album's cover.



Album Listing

3. Administer the Web Site

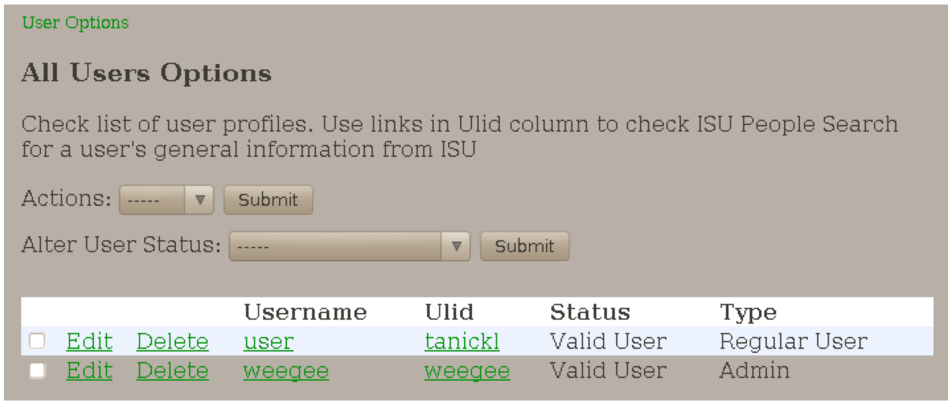
Now we are at the relevant section for the person running the web site. This section will be dedicated to the portions of the web site that are only accessible by a user with administrative privileges.

3.1. Introduction and Layout of Admin Pages

To start administering the web site, you have to log in to the default admin account that will be created when the application is first installed. The username is admin and the password for the account is admin. It is recommended that you change the password for the admin account when you first log in.

The basic layout for the main pages of the admin section are the same so it would be best to use the Users section as the example. Each options page of the admin section will display the current content in the database in a table with data to help identify each individual entry. Within the table, there are **Edit** and **Delete** links to the left side as well as a checkbox. Clicking on the **Edit** link will show the full details of an entity in a form with the option to edit the data. Clicking the **Delete** link will show a confirmation for deleting an entity with a button to click to confirm the deletion.

As for the checkbox, that is for the **Actions** portion of the page. The **Actions** portion of the page can be used to select many entities in the table and perform an action to all selected elements. The main action available currently is a **Delete** action that will delete all selected elements and any currently associated elements. The administration index pages will also include a link titled **Create** which will take you to the form to create a desired entity; the Users admin index does not have an explicit create form page as creating a user should be done through the registration process.


User Options

All Users Options

Check list of user profiles. Use links in Ulid column to check ISU People Search for a user's general information from ISU

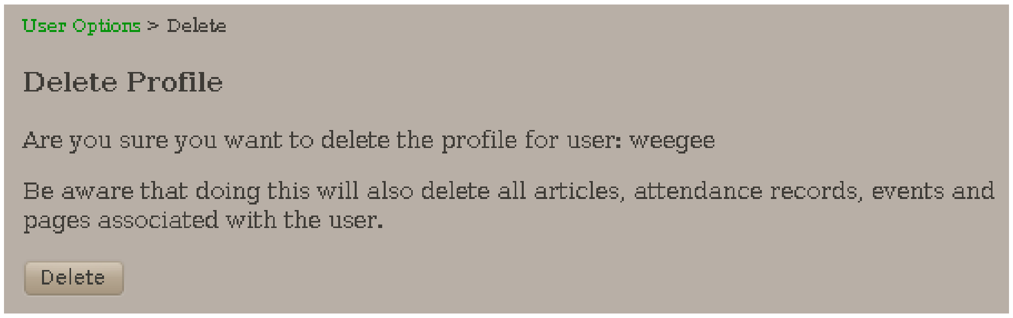
Actions: ----- Submit

Alter User Status: ----- Submit

			Username	Ulid	Status	Type
☐	Edit	Delete	user	tanickl	Valid User	Regular User
☐	Edit	Delete	weegee	weegee	Valid User	Admin

Example of administration index interface (using Users admin index)

Each section of each admin section will include a breadcrumb trail leading back to the index page of the current admin section. Also to note, the Edit and Create pages are very similar for each admin section. The delete section of each admin page will be very similar with a prompt asking if you want to delete an entity.


User Options > Delete

Delete Profile

Are you sure you want to delete the profile for user: weegee

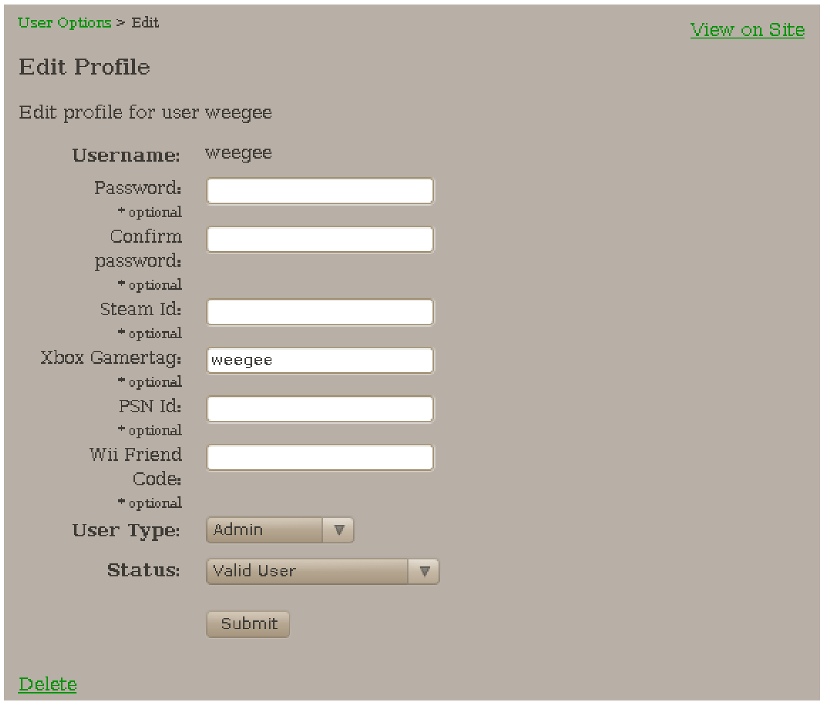
Be aware that doing this will also delete all articles, attendance records, events and pages associated with the user.

Delete

Delete Example (using User delete)

3.2. Users

The Users admin section is used to administer user accounts on the web site. All identifiers are optional. Also, in order for a user to be considered a valid user on the web site, the user must not be of type **Anonymous** and must have a status of **Valid User**.



Edit Profile

Edit profile for user weegee

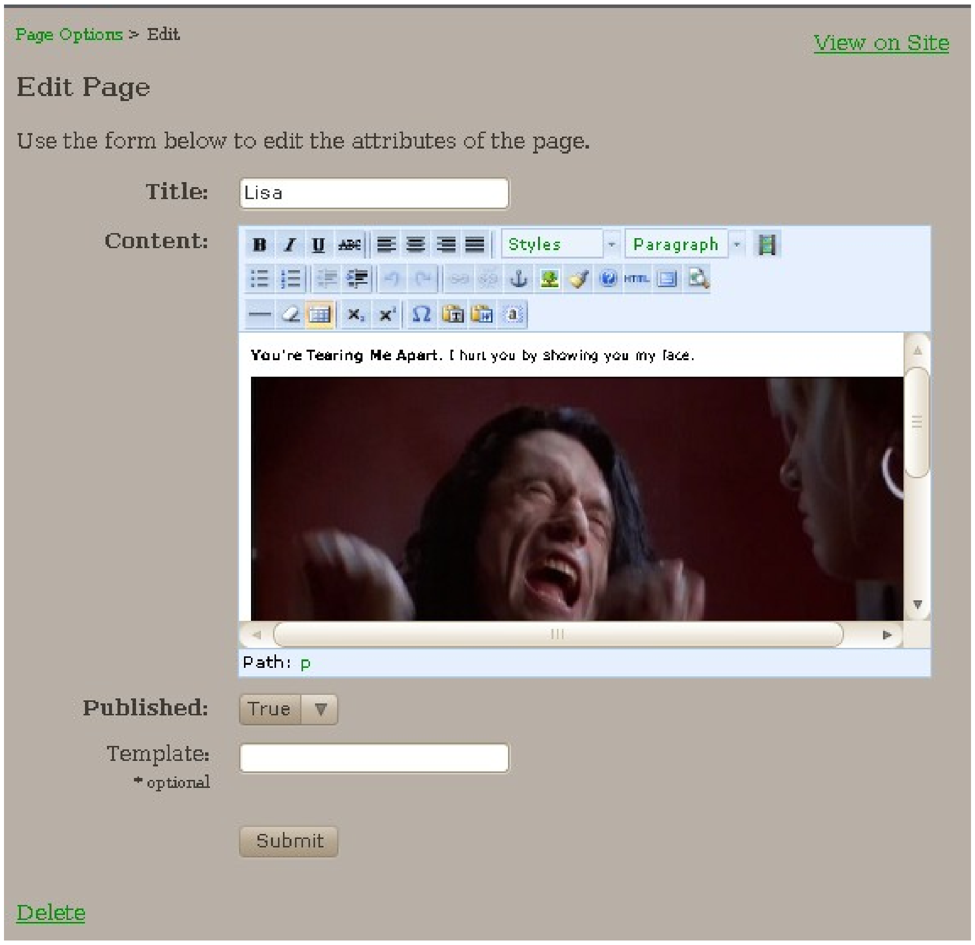
Username:	weegee
Password: * optional	
Confirm password: * optional	
Steam Id: * optional	
Xbox Gamertag: * optional	weegee
PSN Id: * optional	
Wii Friend Code: * optional	
User Type:	Admin
Status:	Valid User

Users Edit Form

3.3. Pages

Pages represent static pages that can have a different layout than the rest of the web site using the template property; if not specified, the default template file view_page_tpl.php will be used. The template file specified will be relative to the main template directory. The TinyMCE is used for editing the content of the page so a user does not need to have knowledge of XHTML.

In order for a page to be visible to all users, the published status has to be set to **True**. However, an administrator can also view unpublished pages in order to see and edit the final appearance of the page before it is visible to regular visitors.


Page Options > Edit

View on Site

Edit Page

Use the form below to edit the attributes of the page.

Title: Lisa

Content:

You're Tearing Me Apart. I hurt you by showing you my face.

Path: p

Published: True

Template:
* optional

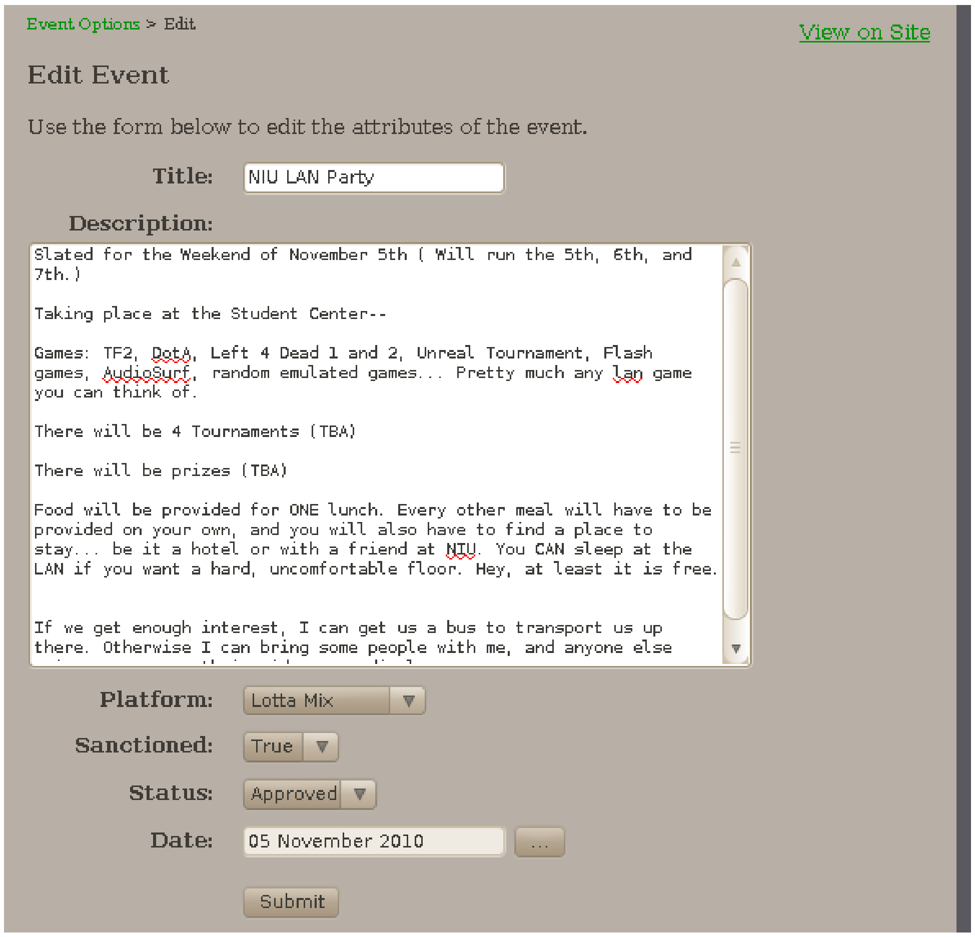
Submit

Delete

Page Edit Form

3.4. Events

The Events admin section is used to regulate the events being entered into the system. Each platform must be categorized by a platform from the 6 specified in the application. In order for any event to appear on the web site, the status of the event has to be **Approved**. The date is specified by clicking the button to the far right of the **Date** row which will bring up a calendar to use to specify the date of the event.


Event Options > Edit

View on Site

Edit Event

Use the form below to edit the attributes of the event.

Title:	NIU LAN Party

Description:

```
Slated for the Weekend of November 5th ( Will run the 5th, 6th, and
7th.)

Taking place at the Student Center--

Games: TF2, DotA, Left 4 Dead 1 and 2, Unreal Tournament, Flash
games, AudioSurf, random emulated games... Pretty much any lan game
you can think of.

There will be 4 Tournaments (TBA)

There will be prizes (TBA)

Food will be provided for ONE lunch. Every other meal will have to be
provided on your own, and you will also have to find a place to
stay... be it a hotel or with a friend at NIU. You CAN sleep at the
LAN if you want a hard, uncomfortable floor. Hey, at least it is free.


If we get enough interest, I can get us a bus to transport us up
there. Otherwise I can bring some people with me, and anyone else
```

Platform:	Lotta Mix
Sanctioned:	True
Status:	Approved
Date:	05 November 2010

Submit

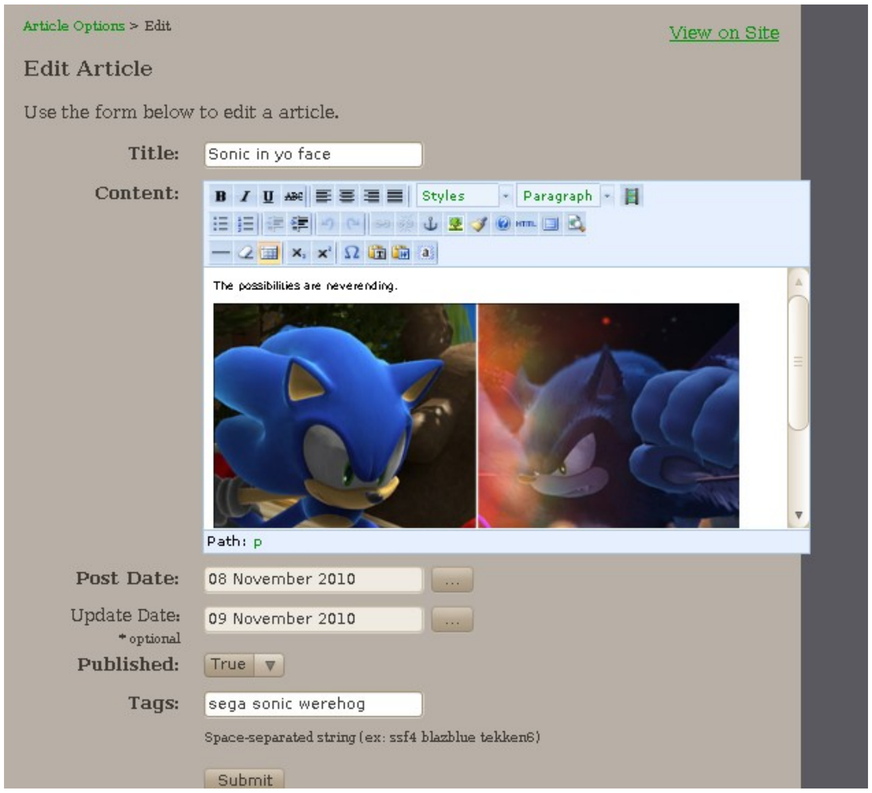
Event Edit Form

3.5. Articles

The Articles admin section is used to administer articles being posted on the web site. Like the Pages forms, the Articles forms use TinyMCE for editing the content of the article to easy the input of XHTML. An article has two dates: post date and update date. The **post date** must be defined but the **update date** field is optional and represents the most recent date that an article was updated.

Regular visitors of the web site will only be able to view the article if the published status is set to **True**. Administrators will be able to view unpublished articles in order to look and edit the final output of the article before publishing it and having it visible by the masses. Articles will be sorted in the listings first by **descending post date** and then by **descending id**.

A special note should be made about the tags field. The form is expecting a space-separated string of tags. Within the application, the string will be split and the tag names will be sorted. The application will look for each desired tag in the database and create a new **ArticleTag** if it does not exist. In the end, **TaggedArticle** entries will exist that will link each **ArticleTag** with the **Article** entity. **TaggedArticle** entities will be synced when an article is edited. Articles can be linked together using the same tag(s). In the articles table, the sorted tag string is sorted in the article row.


Article Options > Edit

View on Site

Edit Article

Use the form below to edit a article.

Title: Sonic in yo face

Content:

The possibilities are neverending.

Path: p

Post Date: 08 November 2010

Update Date: 09 November 2010
* optional

Published: True

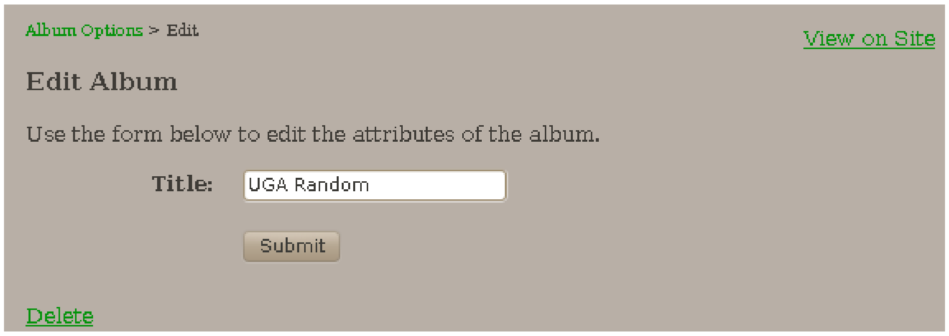
Tags: sega sonic werehog

Space-separated string (ex: ssf4 blazblue tekken6)

Article Edit Form

3.6. Albums

Albums are placeholders that will be used to contain photos. Because of this, the administration of albums is done in two sections. The first is for the albums themselves. An Album entity only consists of its id and a title. The title is the only editable property of the album.



Album Options > Edit

View on Site

Edit Album

Use the form below to edit the attributes of the album.

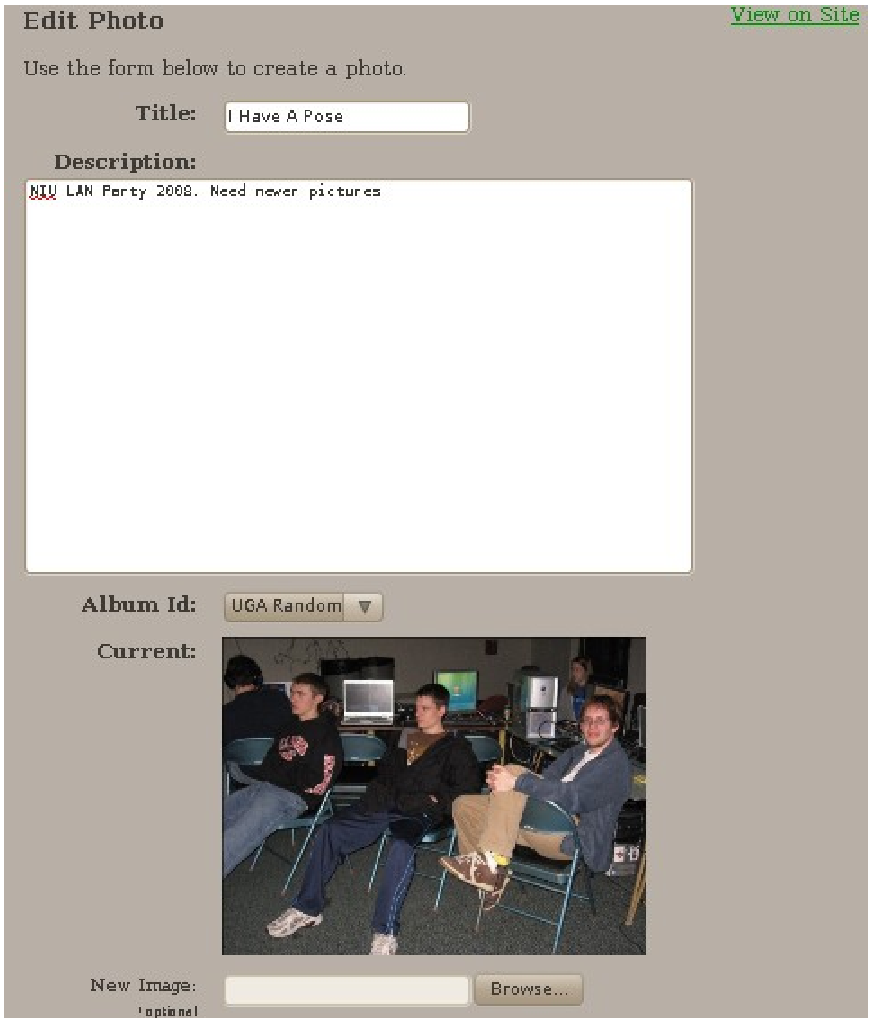
Title: UGA Random

Submit

Delete

Album Edit Form

In order to administer the photos within, an administrator has to go to the regular album section while logged in. While in the album pages, a link for the **Create Photo** page will be visible. While viewing a photo in an album, the **Edit** link will appear next to the album that a photo is filed under.



Edit Photo

View on Site

Use the form below to create a photo.

Title: I Have A Pose

Description:

NIU LAN Party 2008. Need newer pictures

Album Id: UGA Random

Current:

New Image:
optional

Browse...

Photo Edit Form

Within the photo edit form, you can look at the current image associated with a photo and upload a new image, if desired, to associate with a photo. You can also edit the album that a photo belongs to.

When creating a new photo entity, a thumbnail is created for the upload image if the width of the image is greater than **640px**. The width of the thumbnail will be **640px** and the height will be resized keep the thumbnail at the same aspect ratio of the original. The thumbnail will be stored at /media/uploads/thumbnails/$Year_$Month/$filename_thumb.$ext while the actual image will be stored at /media/uploads/thumbs/$Year_$Month/$filename.$ext.

3.7. Extra Notes

It should be noted that there are some alternative ways to access the portions of the administration pages without going through the administration index. When logged in as an admin, the heading for an object will include an **Edit** link that can be used to access the edit form of an object; this can be seen in the article image posted in section 2.5. Also, each edit form page in the administration section has a **Delete** link underneath the form as well as a **View on Site** link to view the page for an object directly.

4. Install and Configure Application

4.1. Dependencies

There are some dependencies that must be met in order to run this application on a web server. This section will outline the dependencies necessary to run the application.

- Apache web server
- MySQL version 5.0 or greater. **Required for use of prepared statements**

- PHP version 5.1 or greater. **Required due to use of PHP Data Objects interface**
- JsCal2. **Included** (http://www.dynarch.com/projects/calendar/)
- PEAR Mail. (http://pear.php.net/package/Mail/redirected)
- phpThumb. **Included** (http://phpthumb.sourceforge.net/)
- TinyMCE. **Included** (http://tinymce.moxiecode.com/)

4.2. Directory Structure

This application consists of many files which are grouped together in different directories. In order to get you better acquainted with the application, here is a list of the file structure of the application and the types of files included.

- / - Main directory holds Controller files and sub-directories
- /config/ - Contains default configuration file
- /feeds/ - Consists of two sub-directories for the RSS and iCal feeds
- /includes/ - Consists of files defining classes that are used in the application
- /includes/models/ - Contains files which specify the Model and DAO classes of the application
- /media/ - Placeholder for images, css, Javascript, etc. files to be accessed in a page
- /media/uploads/ - Holds the directory for uploaded content from the photo gallery.
- /media/uploads/images – Contains all full size images uploaded to photo gallery. **Folder must be writable by the web server**
- /media/uploads/thumbnails – Contains all generated thumbnail images from the photo gallery. **Folder must be writable by the web server**
- /shared/ - Contains bootstrap file and a file to define extra global functions
- /templates/ - Contains XHTML/PHP template files that will be used by the Controller classes to render and populate data

4.3. Configuration

This application comes with a /config/config.default.php file. This file is a placeholder configuration file that predefines many global constants that will be available within the application. There are many constants that the application will expect to be defined although others can be defined as needed. In order to use the configuration file, the values within the config.default.php file might need to be redefined. This section will outline the constants that are expected. **The final configuration file must be located at /config/config.php**.

- **DOMAIN_ADDR** – Defines the domain that the server resides on

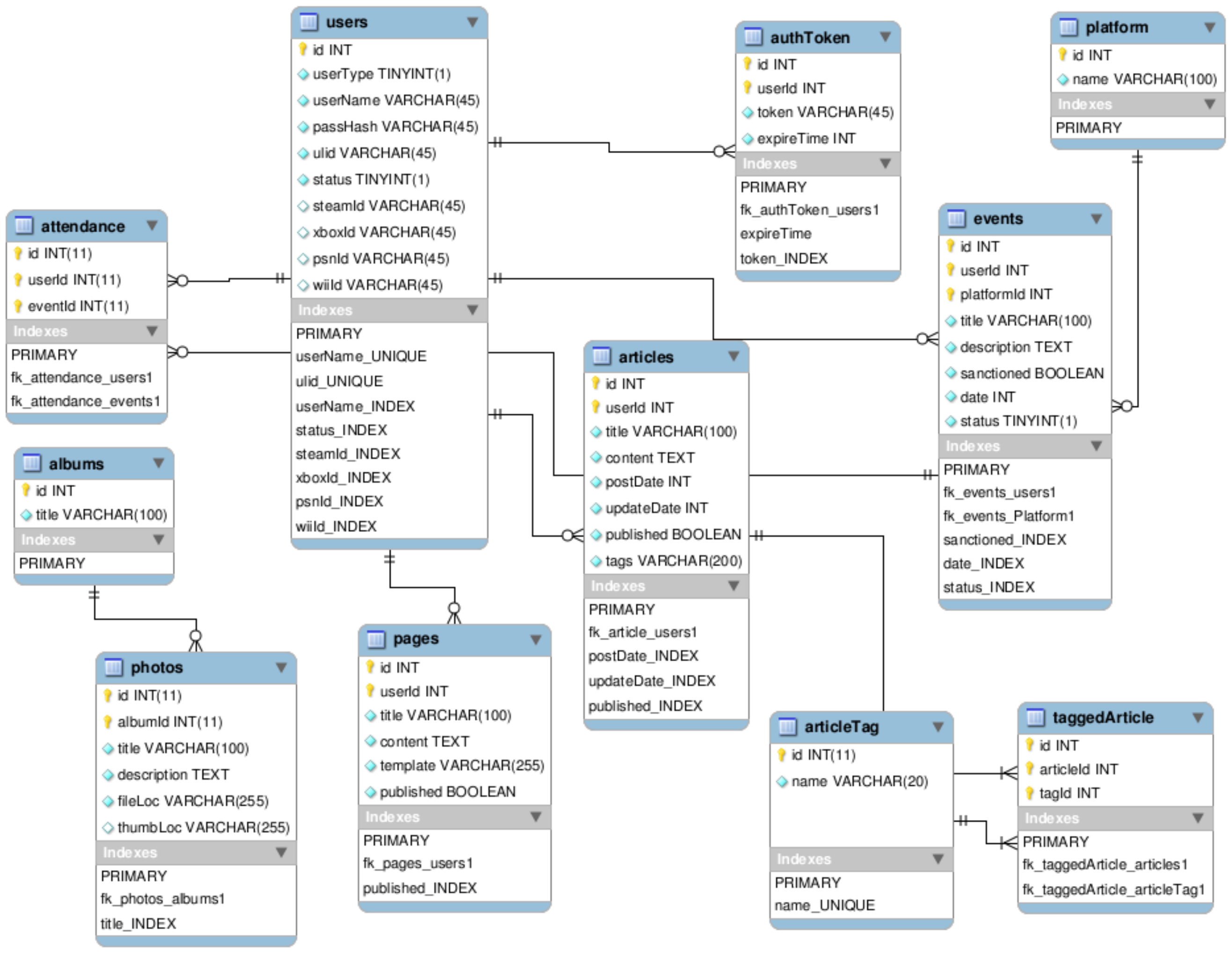
- **BASE_URL** – Defines the URL that is used to access the main directory
- **MEDIA_PATH** – Defines the location of the media directory in the application
- **MEDIA_URL** – Defines the URL that is used to access the media directory
- **SITE_NAME** – Defines the name of the web site which will be seen in web pages
- **DB_HOST** – Define the address of the database server
- **DB_USER** – Define the user of the database
- **DB_PASS** – Define the password required to access the database
- **DB_NAME** – Define the default schema to access
- **EMAIL_ADDRESS** – Define the email address of the site admin. **Required to use email**
- **SMTP_HOST** – Define the address of the SMTP server. **Required to use email**
- **SMTP_USERNAME** – Define the user name used to access the SMTP server. **Required to use email**
- **SMTP_PASSWORD** – Define the password used to access the SMTP server. **Required to use email**

Again, **the final configuration file must be located at /config/config.php** or the application will fail to load.

4.4. SQL Setup

In order to run this application, you will have to have a running MySQL server with a populated schema created by the included uga_schema.sql file. You can do this many different ways but it is recommended that you load and run the file using MySQL Workbench (http://wb.mysql.com/); you could also load the uga_model.mwb file and perform a **Forward Engineer** database operation. Other than that note, this section will define the tables used in this application.

To start this section, here is a picture of the current ERD of the database for this application with indexes specified.



ERD of Database with Indexes Shown

The next portion will give a little more detail about the structure of the tables of the database.

Table: albums

Column	Type	NULL?
id	INT (11)	N
title	VARCHAR (100)	N

Table: articles

Column	Type	NULL?
id	INT (11)	N

userId	INT (11)	N
title	VARCHAR (100)	N
content	TEXT	N
postDate	INT (11)	N
updateDate	INT (11)	N
published	TINYINT (1)	N
tags	VARCHAR (200)	N

Table: articleTag

Column	**Type**	**NULL?**
id	INT (11)	N
name	VARCHAR (20)	N

Table: attendance

Column	**Type**	**NULL?**
id	INT (11)	N
userId	INT (11)	N
eventId	INT (11)	N

Table: authToken

Column	**Type**	**NULL?**
id	INT (11)	N
userId	INT (11)	N
token	VARCHAR (45)	N
expireTime	INT (11)	N

Table: events

Column	**Type**	**NULL?**
id	INT (11)	N
userId	INT (11)	N
platformId	INT (11)	N

title	VARCHAR (100)	N
description	TEXT	N
sanctioned	TINYINT (1)	N
date	INT (11)	N
status	TINYINT (1)	N

Table: pages

Column	**Type**	**NULL?**
id	INT (11)	N
userId	INT (11)	N
title	VARCHAR (100)	N
content	TEXT	N
template	VARCHAR (255)	N
published	TINYINT (1)	N

Table: photos

Column	**Type**	**NULL?**
id	INT (11)	N
albumId	INT (11)	N
title	VARCHAR (100)	N
description	TEXT	N
fileLoc	VARCHAR (255)	N
thumbLoc	VARCHAR (255)	Y

Table: platform

Column	**Type**	**NULL?**
id	INT (11)	N
name	VARCHAR (100)	N

Table: taggedArticle

Column	**Type**	**NULL?**

id	INT (11)	N
articleId	INT (11)	N
tagId	INT (11)	N

Table: users

Column	**Type**	**NULL?**
id	INT (11)	N
userType	TINYINT (1)	N
userName	VARCHAR (45)	N
passHash	VARCHAR (45)	N
ulid	VARCHAR (45)	N
status	TINYINT (1)	N
steamId	VARCHAR (45)	Y
xboxId	VARCHAR (45)	Y
psnId	VARCHAR (45)	Y
wiiId	VARCHAR (45)	Y

Within the archive for this application, there is also the MySQL Workbench file (uga_model.mwb) that can be used to modify the models used in the database and then synchronize any changes to the database.

5. Developer Documentation

This section is devoted to people who would like to expand this application and use the included classes. Since this application is written in PHP, there is nothing forcing you to use the included classes and not write pages in a typical PHP fashion.

5.1. phpDoc

This application uses phpDoc to document all aspects of the application. phpDoc can be thought of as an adaptation of Javadoc written for PHP. Although this section will include an introduction to the various components of the application, there are many details that will not be included in this version of the documentation. For more detailed documentation about the various components of the application, open the phpDoc API documentation. Open the /doc/index.php file in a web browser to view the complete API documentation for the application.

5.2. Introduction to Framework

This application has an infrastructure that utilizes the Model-View-Controller paradigm to organize application functionality. This allows for a much more cleanly structured application and a mostly clear, with a few exceptions, breakdown of where certain kinds of operations will appear which allows for much easier debugging. Although there are many other approaches that can be taken to extending this application, it is recommended that your changes fall in line with the structure of the current pages. In order to assist in this, the main components that make up a page will be explained.

5.3. Bootstrap

The /shared/bootstrap.php file loads many important classes, the /config/config.php file, and several other constants that will be useful in the application. Once loaded, all loaded components will be in scope for the duration of the rendering of the page. This section will mention the classes and other components loaded by the bootstrap file.

Loaded classes and interfaces:

Controller – Defines a controller interface with a run method

DAOBase – Defines portions of a SQL query and includes many methods for building a query

ModelBase – Defines id setter and getter as well as defining magic __get and __set methods. Magic methods are used to access model data in template by $obj->var vs $obj->getVar ().

Paginator – Used to split a result set into several pages

Template – Used to load a template php file

Session – An abstraction for PHP's session management used to load and set expected session variables

Loaded constants (not specified in /config/config.php):

IN_APP – Used in other modules to disallow direct accessible

DS – Shortcut to PHP DIRECTORY_SEPARATOR constant. Defines directory separator character

ROOT_DIRECTORY – Defines the main directory of the application

IS_AJAX – Checks request header to determine if request is from an xmlhttprequest and will set bool accordingly

5.4. PageController Structure

Each page that is meant to be accessible first loads the /shared/bootstrap.php file to load the application's base classes. Next, load any desired DAO classes. The next major step is to create a class that implements the **Controller** interface which defines a **run** method. In the __construct method of the class, define a **PageTemplate** instance which will be used to render the XHTML of the page. The template specified in the constructor of PageTemplate (defaults to index_tpl.php) should be the base template file to use for rendering; the index_tpl.php can use subtemplates if defined with the main_page key in the array passed to the PageTemplate **render** method. Now within the **run** method, implement any page logic that is necessary to fulfill a use case.

To finish within the class, call the **render** method of the **PageTemplate** instance with an associative array with any key, value pairs that you want available within the scope of the XHTML template file. Finally, at the bottom of the file, create an instance of the **PageController** that you just defined and then call its **run** method. The following is a simple example of what a **PageController** file would look like. If you want to override the content pane, specify a file path using the key **main_page** in the data array parameter of the PageTemplate **render** method.

```
<?php

$current_dir = dirname (__FILE__);
require_once ($current_dir . DIRECTORY_SEPARATOR . "shared" . DIRECTORY_SEPARATOR .
"bootstrap.php");
require_once (joinPath (INCLUDES_DIR, "models", "Article.php"));

class PageController implements Controller {
   protected $template;

   public function __construct () {
      $this->template = new PageTemplate ();
   }
```

```
    public function run () {
        $session = Session::getInstance ();

        $articleDAO = ArticleDAO::getInstance ();
        $results = $articleDAO->all ();

        //$results = $user;
        $this->template->render (array (
                                "title" => "Index",
                                "results" => $results,
                                "session" => $session,
                            ));
    }
}

$controller = new PageController ();
$controller->run ();
?>
```

5.5. Models and DAO Classes

The **Model** and **DAO** classes go hand in hand. Both classes will typically be defined in the same file. The **Model** class will be used to hold and manipulate the data from the database. The **DAO** class will define how to manipulate the database and the methods will retrieve **Model** instances on database reads and save data from **Model** instances on database writes. The following section will explain in more detail about how the **Model** and **DAO** classes are structured.

5.5.1. Model

A **Model** class will extend the abstract **ModelBase** class. A **Model** class will define properties that will correspond with columns of a database. A Model class will also define a **setter** and **getter** method for each column defined in the database. Getter methods must be named as **get** + uppercase property name (ex. getArticleId). Setter methods are defined in the same way except with a **set** prefix instead of get (ex. setArticleId). That is the basic structure of a Model class. You can define any additional methods or properties as needed. The following is an example of a small **Model** class.

```
require_once ("Article.php");
require_once ("ArticleTag.php");
```

```
class TaggedArticle extends ModelBase {
   protected $articleId;
   protected $tagId;

   public function setArticleId ($articleId) {
      $this->articleId = $articleId;
   }

   public function getArticleId () {
      return $this->articleId;
   }

   public function setTagId ($tagId) {
      $this->tagId = $tagId;
   }

   public function getTagId () {
      return $this->tagId;
   }

}
```

5.5.2. DAO

A **DAO** class will extend the abstract **DAOBase** class. Each DAO class should correspond with a table in the database. A DAO class will define some properties of a database table, such as table name and table columns, along with many methods for manipulating the database. Also, a DAO class will utilize the Singleton pattern so only one instance will be available.

The property $tableName is a string that is the name of the database table to manipulate. The property $columns is an array of strings. Each element corresponds to the name of a column in the table. The property $instance holds the actual instance of the class to be retrieved or created by the **getInstance** method.

Only the **getInstance** method is required to be implemented due to the **DAOBase** class. However, all other included DAO classes utilize a similar interface that consists of load, save, insert, delete, all, and count methods. The following example is an outline of what a DAO class consists of; for more detailed information, look at the included **phpDoc** package.

```
class TaggedArticleDAO extends DAOBase {
    protected static $instance;
    protected $tableName = "taggedArticle";
    protected $columns = array ("id", "articleId", "tagId");

    public static function getInstance () {
        if (!isset (self::$instance)) {
            self::$instance = new self ();
        }

        return self::$instance;
    }

    public function load ($id, $options=null);
    public function save (TaggedArticle $tagged);
    public function delete (TaggedArticle $tagged);
    public function insert (TaggedArticle $tagged);
    public function all ($options=null);
    public function count ($options=null);
```

Another thing to note is that the queries used in this application make use of prepared statements in order to prevent SQL injection from occurring. It is advised that you use prepared statements in queries to escape input from the user to prevent SQL injection from occurring.

One last thing to note, this portion of the documentation does not include any details about the DAOBase class. If you want more details about the functionality provided by the DAOBase class, please read the **phpDoc** API documentation for the DAOBase class.

5.6. PageTemplate Class

The **PageTemplate** class is used to load an XHTML + PHP template file. Within the __construct method, the template file and base directory are set along with the SITE_NAME and MEDIA_URL constants if not set in the configuration file.

```
public function __construct ($tpl_file = "index_tpl.php", $tpl_dir_name = "templates")
```

The other major method of the PageTemplate class is the **render** method. The method takes an optional associative array and the key, value pairs from the array will be extracted and then available in the scope of the template file. The PageTemplate will also load the extra template files specified in

/includes/template_functions.php file. Finally, the template file is loaded and the output from the file will be rendered.

```
public function render ($data_array=null)
```

The actual template file will typically be named *_tpl.php and contains the XHTML needed to render a page along with some PHP to make the page dynamic. The template file will have access to any variables passed to the render method via the $data_array along with other global constants. The following is a small example of what a template file would look like.

```
<html><head><title>Example</title></head>
<body><p><?php echo “Hello World” ?></p></body></html>
```

6. Conclusion

This is the end of the introduction documentation for this application. Hopefully, reading this document has been enlightening. As noted before, this is not the end of the documentation for this application. For more detailed documentation of the components of this application, please read the included **phpDoc** API documentation located at /doc/index.php. Thank you for reading this document. You win.