Networking Website for Artists

Keenan S. Staffieri
University of Redlands

Follow this and additional works at: https://inspire.redlands.edu/cas_honors

Part of the Art and Design Commons, Databases and Information Systems Commons, and the Software Engineering Commons

Recommended Citation

Creative Commons Attribution-Noncommercial 4.0 License
This work is licensed under a Creative Commons Attribution-Noncommercial 4.0 License
This material may be protected by copyright law (Title 17 U.S. Code).
This Open Access is brought to you for free and open access by the Theses, Dissertations, and Honors Projects at InSPIRe @ Redlands. It has been accepted for inclusion in Undergraduate Honors Theses by an authorized administrator of InSPIRe @ Redlands. For more information, please contact inspire@redlands.edu.
Senior Capstone Thesis

Computer Science

"Networking Website for Artists"

Keenan S. Staffieri

Project Advisor: Pani Chakrapani
ArtSmirk.com – Networking Website for Artists – Thesis

ArtSmirk is a networking website designed for artists to promote and enhance their creative expression. This can be achieved by providing artists with a web application that lets them upload their work for others to comment, critique, or offer suggestions. Users of this web application can view other member profiles and galleries, while being able to manage their own. ArtSmirk takes advantage of Web 2.0 technologies such as AJAX for retrieving data from the webservice asynchronously. These approaches to development make the web application more responsive to user requests and the navigation more intuitive. ArtSmirk has been designed to incorporate the future implementation of the latest invention for the web: HTML5 Canvas. With HTML canvas, graphics can be rendered and manipulated via JavaScript directly on the page without the need of a plugin such as the Adobe Flash Player. Currently, ArtSmirk does not use the HTML5 doctype, because many browsers lack support at the time of this writing.

The ArtSmirk website can be accessed by anyone who has a computer with an internet connection and a modern web browser. There are essentially three types of users for the ArtSmirk network, and the features offered by the web application depend upon the type of user. The types of users are Guest-User, Member-User, and Admin-User. Guest-Users are people who have just accessed the website and either 1) have not yet registered an account, or 2) they have previously registered an account, but are currently not logged in. Member-Users are users who have registered an account on ArtSmirk, and are currently logged in because they provided a matching username/email and password. Admin-Users either refer to the administrator of the website (there should only be one), or a website moderator. The administrator has unlimited power, and is permitted to remove other users, entire content categories, comments, posts, and critiques from the database. Moderators can delete comments, posts, or critiques which they deem inappropriate.

Users who access the domain, www.artsmirk.com, are brought to the homepage as a Guest-User. The homepage shows the most recently uploaded content by members of ArtSmirk,
contests, polls, and an option to become a member. The core purpose of the homepage is to give users immediate feedback on what’s new and to quickly and easily search for other artists, uploads, or groups. Guest-Users still have access to many of ArtSmirk’s features, such as the browsing of other profiles and member galleries. However, Guest-Users cannot browse the profiles or galleries of members with “private” content or accounts. Content that is marked “private” means that only ArtSmirk members or a select group can view the content. Obviously, Guest-Users cannot create and manage their own profile, and they cannot create or join user groups. The ArtSmirk web application can “get out” of Guest-User mode after the user has joined the website and logged in with a matching username/email and password. Currently, all submissions on ArtSmirk are marked “public” and can be viewed by anyone, including Guest-Users.

When a user logs into their account and becomes a Member-User, a User CP (Control Panel) appears on the header of every page. The User CP contains links to the user’s Account Settings, the Help Center, an option to logout, and the File Uploader. In the future, the homepage will also include more modules, such as a component that displays the most recent uploads by friends on the current user’s friend’s list. Towards the right of the page, will be a live feed component that displays the most recent statuses by the user’s friends.

The File Upload page, only accessible by members, lets users submit their content, which can either be artwork or writing. Members can also select the category their piece belongs to, input a title, description, and some keywords that help for search queries. Files that are submitted are immediately visible from that user’s gallery and the homepage also displays the most recently submitted work. Others users (guests or members) can then click on a link that brings them to a page, displaying the piece with comments, critiques, and some statistics about the image or writing. The appearance and content of the viewing page is different for both visual arts and writing. If a user is viewing an image submitted by another user, links are made to download a full version of the image, or to add the image to their favorites. User-submitted
writing can be viewed in a text-viewing module that can display two pages on the screen (like a book-reader), and the user can change the size of the text, or invert the colors (i.e. white text on black background instead of the reverse). However, no matter what type of content is being viewed, users may view comments and critiques, or write one of their own. There are also filters and tabs that let users view only the critiques, or sort them from most recent to oldest comments or critiques. In ArtSmirk's current development stage, the viewing page only supports the display of submissions under the visual arts, comments, image statistics, and an artist description. Critiques and a full-fledged text reader have not yet been implemented, but users can reply to comments (this is known as having “nested comments”).

Members can also leave comments on each other’s profiles and view the latest submissions by user. In the future, users will be able to send private messages and request friendships with one another. Member friendships are useful because friends are immediately notified when friends upload new work, and they may have access to additional galleries on a friend’s profile that other members do not have access to.

In addition to submitting content for user galleries, users will be able to submit articles and tutorials for other members to use as resources, or even submit texture packs and other useful tools for image manipulation and writing. Also, members will be awarded points in rank for submitting an article or tutorial. Writing a constructive and helpful critique of another member’s submission also helps promote rank. A higher rank gives members and their work precedence while search results are being calculated.

What separates ArtSmirk from other “artistic” networking websites like deviantArt is more emphases on enhancing creativity. This is achieved by offering more robust functionality for submitting and searching for articles, tutorials, and resources like texture packs, plugins, and tools. Also, users will be able to comment on other work with more than just words. Members should be able to literally write comments directly on the submitted image, with arrows, or anything else can be drawn with a mouse or tablet. Also, users should be able to submit “in
progress" work, which creates tabs that display an art piece or writing in its different stages of development.

ArtSmirk, in its current stage of development includes the basic functionality for many of the features I had hoped to accomplish. This includes a new user registration from that is validated client-side (JavaScript) before being validated yet again on the server-side (PHP) and stored in a MySQL database. Users can upload images (JPEG or PNG) and the full image, including a thumbnail variant are stored on the webservers. After an image is uploaded, metadata about the image, such as the user ID, submission category, file size, and size in pixels are stored in the database. The homepage retrieves sorted data from the webservers in JSON format and displays the thumbnail images using HTML and CSS for positioning and style. Member profiles have two modules, one for displaying the last four submissions, and another for displaying profile comments. Members can also edit their status messages and select a mood. Submitted artwork can also be filtered by category. Users can change their personal information and the privacy information regarding what is displayed on their profile.

Websites
The following websites are related to ArtSmirk development, including my portfolio website.

ArtSmirk Development Wiki: http://smirkydevwiki.artsirk.com
My Portfolio Website: www.keenancreations.com
ArtSmirk: www.artsirk.com
Use Cases – ArtSmirk

When developing ArtSmirk, I thought about the types of users who would be using my website, and with that, I formulated a collection of Use-cases. Use-cases are a description of steps or actions that can occur between a user and a software system. These sequences of steps end up leading towards something useful.

There are essentially three types of users that are associated with the ArtSmirk social network: Guest-User, Member-User, and Admin-User. These types of users can be viewed as Actors that want to use the web application. These Actors have different goals and ambitions in mind.

Types of Actors

Guest-User

Guest users are typically users who have just reached the homepage of the website. At first, this user may or may not be an actual member of ArtSmirk. These users are typically people who just want to browse the public profiles and creative works that were uploaded by members of the website. Therefore, user control panel content will not be displayed (otherwise known as the User CP). Instead, guest users will see input fields asking for a username and password. Guest users will not be able to access the private information of members. Also, other website content will be omitted for guest users because it will not be necessary. For example, status updates and news feeds and User CP modules will be omitted from the website for guest users.

Member-User

Members of ArtSmirk have access to the User CP so they have more access to the features on the website such as the ability to change account settings and check their private messages (PMs). Members can also view the status updates and news feeds of other members who are on their friend’s list. Members also have access to the private profiles of other members who are on their friend’s list. Members also have additional interface functionality that lets them upload their creative works.

Admin-User

Administrators have special accounts that give them access to pages that are hidden and inaccessible for the standard user. These pages provide admin users with administrative powers; however, these powers are limited to their “admin privileges,” which is dictated by the founder of the website (me). Administrators also have additional interface functionality on the main website. For example, admin users can delete inappropriate comments, critiques, or uploaded work. They can also ban users temporary and these privileges are preserved on the forums.
Actor Goals

The following is a list of the three actor types, and their possible actions.

**Guest-User**

**Possible Actions:**
- Create a new account
- Browse creative works uploaded by public members
- Browse articles and tutorials uploaded by members (note: tutorials and articles are always public)
- Browse the profiles of public members
- View the status of community contests
- Read public forum posts by the community
- Search for ArtSmirk members with different search parameters
- Browse public member-created groups

**Member-User**

**Possible Actions:**
- Members maintain all functionality of guest users, in addition to:
  - Change account settings
  - Browse creative works uploaded by public members in addition to private works by those on their friend’s list
  - Browse the profiles of public AND private members on their friend’s list
  - Check for private messages
  - Send private messages to other members
  - Participate in the community forums (read and post)
  - Change privacy settings on the account
  - Upload their own creative works for other members to critique or comment
  - Comment or critique work uploaded by other members
  - Upload a new article or tutorials (must be written by the user)
  - Browse public AND private member-created groups
  - Create new user groups
  - Invite members to user groups
  - Edit friends list

**Admin-User**

**Possible Actions:**
- Admins maintain all functionality of members, in addition to:
Use-Cases

Use-Cases provide developers with a clearer vision, or larger picture, of what is to be accomplished. Each use-case represents a specific scenario, or goal, the user is attempting to accomplish. Every use-case has a name, the type of use its applicable for, the end goal, description, and expanded detailed information.

UC-001

Name: Create A New Account
Applicable for: Guest-User
Goal(s): Guest users should be able to register a new account on the website. In other words, database information must be stored for this new user.
Description: This should only apply to guest users. A guest user can register an account on ArtSmirk by clicking on the appropriate link. Account registration should be fast, simple, and easy. Account registration requires an email, password, real name, date of birth, gender, and whether or not the user wants to receive newsletters in their email.
Expanded Detail: The manner in which account registration is handled should be very modern. As the user types in their desired username, requests should be sent to the server asking for whether or not that username has already been taken by another member. Also password fields should be checked for consistency and "password strength." This can all be achieved via AJAX functionality. After the form is checked for validity, the new data will be sent to the server and added to the database.

UC-002

Name: Browse Creative Works Uploaded by ArtSmirk Members
Applicable for: Guest-User, Member-User, Admin-User
Goal(s): Users should be able to browse content uploaded by ArtSmirk members. The browsing experience should be intuitive and interactive for all users. Generally, content is separated by category, but can be sorted based on the user's preferences. The goal of this use case is
to provide users with the functionality required, while presenting this content clearly and logically.

Description: Guest users should only be able to browse content uploaded by members who want their work to be displayed publically. When a user (of any type) enters the homepage, there should be a list of content that has been most recently uploaded. This new content will be represented via thumbnails. To the user's left is a navigation pane that lists off all categories on the website. When a user clicks on a category, a new query is sent off to the server and then relevant content is returned to the user for viewing. Each thumbnail contains a descriptive tag associated with it. The tag contains the title of the piece, the author's name, and the category it is in. However, the category will not be displayed if the user is already browsing a specific category (that would just be repetitive information!). Clicking on a thumbnail will load a new page that displays an expanded view of the image, along with critiques, comments, statistics, and other interface elements that are associated with the creative work.

Different categories may display user-uploaded content differently. For example, the writing category will render previews of uploaded work more appropriately. However, categories with mixed media will display content in a more universal manner.

Expanded Detail: The user browsing experience of media is definitely one of the most important components of the website. Also, “browsing” content will most likely be one of the most common activities for users. Thus, it is very important to consider performance when scripting this portion of the website. Server-side scripting should be avoided as much as possible. Requiring javascript to run on the browser is a big plus in terms of reducing server load. When a user is on a page, portions of the website should be updated during different time intervals. These time intervals vary depending on the current server load and whether or not the user has most likely abandoned their computer for extended periods of time.

**UC-003**

Name: Browse Articles and Tutorials Uploaded by ArtSmirk Members

Applicable for: Guest-User, Member-User, Admin-User

Goal(s): All users should be able to view the articles and tutorials that have been uploaded by ArtSmirk members. Articles and tutorials can never contain 'private' data, because these resources are meant to be distributed to the public.

Description: Tutorials and articles should be listed in an organized fashion, and should be categorized just like the uploaded work by other members. If the user is a member, then comments can be made, in addition to a rating out of five “smirks” (1=lowest, 5=highest). Positive reviews reward the member who wrote the article or tutorial more “points.” Tutorials and articles can also be sorted depending on the user's request.

Expanded Detail: Positioning the mouse over various elements on the articles/tutorial list will display additional details about the article/tutorial.
**UC-004**

**Name:** Browse Member Profiles

**Applicable for:** Guest-User, Member-User, Admin-User

**Goal(s):** Users should be able to view at least basic information about other ArtSmirk users. Users without appropriate privileges should not be able to view the profiles of private members.

**Description:** User profiles are usually accessed by clicking on another user's name, or they can be found using the websites searching feature. Not every member wants to share every bit of information to the public. It would be wise to let the user specify which bits of information are public and which aren't. Also, profile information should be displayed logically; relevant information should be categorized into various "modules," which could be collapsed or expanded. If the current user is a member and is viewing their profile page, an 'edit' button or icon should appear at the corner of each icon. When this button is clicked, the user should be able to edit the module on the spot without having to be directed to a new page.

**Expanded Detail:** Different sections of the profiles will be separated by tabs. Clicking on a tab should not direct the user to another page, but rather update a portion of the page to display different data.

**UC-005**

**Name:** View Status of Community Contests

**Applicable for:** Guest-User, Member-User, Admin-User

**Goal(s):** Members and guests want to check the status of community contests.

**Description:** Various community contests will be held every week (or month depending on the contest). Guests will be able to view the status and content of these contests but will not be able to submit their own work or comment on other submissions.

**Expanded Detail:** Members will be able to vote for the piece they think should win. The current number of votes for each piece will be displayed. Contest winners will receive special trophies.

**UC-006**

**Name:** Participate in the Community Forums

**Applicable for:** Guest-User, Member-User, Admin-User
Goal(s): Members should be able to post and read forums. Guests should only be able to read forum posts. User accounts for the forums should be obtained from the database.

Description: Various community contests will be held every week (or month depending on the contest). Guests will be able to view the status and content of these contests but will not be able to submit their own work or comment on other submissions.

Expanded Detail: Forums will be extended from the phpBoard implementation. Users can navigate to the forums by clicking on the appropriate link or by typing in the appropriate url: www.forums.artsmirk.com.

UC-007

Name: Searching for ArtSmirk Members

Applicable for: Guest-User, Member-User, Admin-User

Goal(s): All ArtSmirk members can be searched for. Members with private accounts will be displayed but information will be very limited (i.e. like Facebook). An account will no longer seem private if the current user is a friend with the other user.

Description: The 'search for artist' box can be easily found on the home page (or any page for the matter). As the user types into the box, AJAX functionality will be implemented and other members will appear that closely match the query. Therefore, users do not need to type out entire names. A small list will be dynamically generated and the user can click on any of the suggested users or use their arrow keys to cycle through the choices. When the user hits "Enter," a new page will be displayed that lists all the users who match the query that was sent off to the database. Users can sort the results in several different ways, for example: alphabetical, by location, date joined, etc.

Expanded Detail: Javascript will be used to request information from the server, and queries will be sent to the database requesting for users who closely match what the user types into the search box. Then, this information will be outputted in HTML and styled with CSS.

UC-008

Name: Browsing Member-Created Groups

Applicable for: Guest-User, Member-User, Admin-User

Goal(s): Users want to view user-created groups and potentially participate in them. Some information should be hidden to the public, but only if that is desired by the group leader.

Description: Users should be able to create new groups, invite users, and possible kick members out of a group if they happen to misbehave. Groups should also support different titles (or roles) that members can serve. For example, there may be a group leader and several officers,
and each officer may have rights to certain privileges and others won't.

Expanded Detail:
Group leaders can send messages to each and every member in their group. Group members can post comments or images on the group's main page.

UC-009

Name: Change Account Settings

Applicable for: Member-User, Admin-User

Goal(s):
Users may want to change the settings associated with their account. For example, the user may have submitted incorrect information during account registration and they may want to change their date of birth, or gender. Users may also want to change their privacy settings.

Description:
Users must be logged into their account before they can access the page that lets them change these settings. The link to the account settings page can be found on the User CP, and the User CP is only visible to users who are logged in to their account. The account settings page should let users change the following:

• Privacy settings: change profile content to either public or private.
• Date of birth
• Gender
• Location
• Time Zone
• Forum & comment signature
• Change email & password
• Option to deactivate account (this will permanently delete the account from the database).

UC-010

Name: Check for Private Messages

Applicable for: Member-User, Admin-User

Goal(s):
Members can send messages to one another via the website build-in private messaging service. Users must have an account in order to send a private message (guest users cannot send an anonymous message to a member).

Description:
A link to the private messaging page can be found on the User CP. A user is notified of a new private message when the 'mail icon' is highlighted and includes a small number that indicates how many unread messages that user has. The mail page lists out all the messages that have been unread first, but this list can be sorted, filtered, and searched. Users can reply to these messages like how they do with emails. When a new message or reply is sent, the other user will be notified promptly and that message will show up in
their inbox.

Data Flow Diagrams – ArtSmirk

Pages
- sources of destinations of data.

Processes
- take data as input, do something with it, and output it.

Data Flows
- input & output data objects.

Data Stores
- databases & XML files.

External Entities
- refers to users or objects outside of the actual app.
Test Case Definitions & Systems Testing – ArtSmirk

During the Systems Testing stage of development, I am looking for errors, bugs, or implementation problems regarding my code. Test case definitions make this procedure more organized by forcing me to recognize the desired result and the procedure required to reach that successful state.

### Case ID: 001 – Logging in a User

**Description:** A user is attempting to login to ArtSmirk with the appropriate credentials. The server should successfully log the user in if a matching username or email and password are found.

**Procedure:**
1. Guest-User is viewing a web page from the ArtSmirk website, containing the User Control Panel.
2. Guest-User positions the mouse courser above the 'Login' button.
3. Guest-User types in their username/email and password into the appropriate fields.
4. Guest-User hits the 'Enter' key or clicks on the 'Login' button.
5. Values from both input fields should be sent to the login_user.php file via POST, and a MySQL query is issued to retrieve a matching row of data from the user with the requested credentials.

**Success Criteria:**
If a user is found with those credentials, then the information selected from that MySQL row should be stored in a PHP session. If the MySQL result set returns no rows, then no users were found with those matching credentials, then an alert saying 'Supplied Username/Email and Password combination is incorrect' should be displayed.

**Result:**
This test case was successful given username and password matches, and the alert is displayed when the combination is invalid. However, input for the username is no case-sensitive, so the functionality is not perfect.

### Case ID: 002 – Logging out a User

**Description:** A user is attempting to log out of his account (or session), and the web page should be refreshed and return functionality back from Member-User to Guest-User status.

**Procedure:**
1. Member-User is viewing a web page from the ArtSmirk website, containing the User Control Panel.
2. Member-User positions the mouse courser above their username div container.
3. Member-User selects the 'Logout' option.
4. That user's PHP session should be destroyed and the web page should be reloaded and bring the user to Guest-User status.

**Success Criteria:**
The logout_user.php script should be called after the 'Logout' option is clicked by the user. If session data does indeed exist (and it should if the user
is already logged in), then the session should be destroyed and the page reloaded.

Result: The test case has been successful so far. Under all circumstances, all users have been able to successfully log out of their accounts.

<table>
<thead>
<tr>
<th>Case ID</th>
<th>Description</th>
<th>Procedure</th>
<th>Success Criteria</th>
<th>Result</th>
</tr>
</thead>
</table>
| 003          | A user is attempting to post a new comment or reply to a previous comment. Data describing this comment should be stored on the database and the comment listing should automatically be updated via AJAX after the comment has been submitted. | 1. Member-User is viewing their profile, the profile of another member, or a submitted piece of artwork.  
2. Member-User clicks on the 'Write Comment' button, or the user decides to reply to a previous comment by clicking on one of the 'Reply' buttons.  
3. Member-User types in their message into the input field and hits the 'Send' button.  
4. The data is sent to the post_comment.php file and the text data should be stored on the database. | The comment should be successfully added to the database, and the comment listing on the web page should be updated via an AJAX call and looping through the resulting JSON object and formatting the results in HTML and CSS. | This functionality works for the most part, but not everything works as intended. What does work is the following: Users can not submit blank messages, and alert will appear saying that the message is blank and refuse to send useless data to the php script. However, the input values are stored in HTML format, so users can apply HTML formatting to their posts, by making some text bold, or changing the font size or color. |
| 004          | A user is attempting to upload their artwork onto ArtSmirk. The uploaded image should be saved on the server along with a generated thumbnail of the piece. | 1. Member-User is on the Upload form and wants to upload an image of a supported type: JPEG or PNG.  
2. Member-User supplies to form with all required fields (title, category, file to upload, and description).  
3. Member-User hits the Upload button.  
4. Using PHP's File uploading and Post data, the file along with a thumbnail version of the image should be saved onto the file system of the webserver, and meta-data describing the image upload should be stored on the webserver. | | |
Success Criteria: The recently submitted piece should appear immediately on the homepage, the thumbnail displayed, and data describing the upload.

Result: Files of varying types and sizes have been tested for this input form. The PHP scripts associated with this form do check for the type of file that is being uploaded and the size of the image. If the size is >4MB, then the uploaded refuses to upload/store the image on the webserver because it is too large. Also, files of an invalid type (such as a file type that is not JPG or PNG), are also refused. (note: GIFs are not supported because the GD library used for resizing images does not support GIF manipulation).

<table>
<thead>
<tr>
<th>Case ID:</th>
<th>005 – Fetching Recently Uploaded Submissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description:</td>
<td>A user would like to view the most recently submitted artwork on ArtSmirk. Thumbnails representing these submissions should be displayed within a container, and the user should be able to easily view information about those images without much effort.</td>
</tr>
</tbody>
</table>
| Procedure: | 1. Member-User is on the home page and is viewing the 'Recently Added' module.  
2. Member-User positions his mouse over the 'Show' selector and selects 'Show 32.'  
3. Once selected, an AJAX call is issued, and JSON encoded data is requested from the server.  
4. If the request was successful, JavaScript will loop through the elements of the JSON object and append each thumbnail, formatting them with HTML and CSS. |

```javascript
function fetch_sub_json(showNum) {  
  $.getJSON("fetch_submissions.php?show_num=" + showNum,  
  function(json) {  
    $('#stream-container-recent').empty();  
    for(i=0; i<json.length; i++) {  
      $('#stream-container-recent').append(newThumb(json[i].sub_id,  
        json[i].title, json[i].cat_id, json[i].cat_name,  
        json[i].thumb_url, json[i].username,  
        json[i].date_uploaded));  
    }  
  });  
}
```

Success Criteria: If the JSON encoded data was retrieved successfully, then each thumbnail will be appropriately appended to the thumbnail container. The user will therefore be able to see the most recent submissions to ArtSmirk.

Result: So far, this has always been successful. If the user selects any of the options 'Show 16,' 'Show 32,' or 'Show 64,' the thumbnail container is cleared of the previous results and replaced with the updated HTML/CSS content from the resulting JSON data. However, the implementation of these features are not ideal. When a user issues a request to update the container, there is some delay because the request needs to be processed by the server-side PHP script, and the thumbnails do take some time to download. A more elegant way would be to show a temporary HTML div that says something like
A user is attempting to view the profile of another member, or his own. Elements on the profile page should be filled with information that is retrieved from the database on the webserver.

1. Member-User clicks on a link that brings him to a member profile. The url leading to the member profile is provided with a GET parameter called member_id, which tells the PHP script which user to grab information from.
2. An AJAX call is made, and data is returned from the PHP script, where each data element is separated by a comma. (This method is inefficient when compared to returning JSON encoded data).
3. The populate_profile() function is called upon, and various elements on the webpage are replaced with the data obtained from the database on the webserver.
4. If the current user is viewing his profile, then an option to update the status and mood should be made available to the user.

If the data is retrieved successfully from the webserver, then all fields should be appropriately populated.

This sequence of steps has not failed me in the tests I have ran thus far. However, my implementation was inefficient because I was not retrieving the data in a JSON encoded format. Because of this, my JavaScript code was much longer than it needed to be, and also, far more confusing than it had to be.
Conclusion

Working on my senior capstone project in Computer Science has certainly been an intense learning experience. Previously, I had never developed a large-scale web application like this before, so I had to endure some rather frustrating experiences. On occasion, I had consumed alcoholic beverages while programming, in order to relieve the stress. But usually, I relied on caffeine to keep my mind in a highly beta-wave induced state of consciousness.

In the end, there are some things that I would’ve done differently. As I was programming the application, I was starting to lose sight of the "big picture." I was forgetting about why ArtSmirk should exist, and what it would give to the users. I was getting lost in the details, so my vision became blurred by all the rush to get things done in a timely manner. But since then, I have learned my lesson: Never lose sight of the big-picture. In my current mental state, I feel like I could answer any question about my project relating to my code, or some small detail. But I would stumble if I was asked a very basic question like, "So... why should someone use ArtSmirk instead of deviantArt."

“Well...", I would say. The truth of the matter is that, “ArtSmirk is far too young right now to live up to its older brother, deviantArt.” However, ArtSmirk does have potential, and I hope to raise ArtSmirk as a loving father would raise his child.
Works Cited

