**Final Project Part 3: Ji-Rex**

**Alex Pan, Jihea Khil, Jiho Chang, Ji Young Lim**

**Website: http://info230.cs.cornell.edu/users/jhkhil/www/Jirex/index.php**

**Website Layout**

The actual layout of our website is: http://info230.cs.cornell.edu/users/jhkhil/www/Jirex/index.php

We are still in the process of changing our overall look of the website, so the color and design may change for the final submission.

The following is the description of our website layout:

The entire website will have the same header and footer that will include: Organization Logo, Login, Members (if member logged in), Search, and icons that link to Social Media Websites. Moreover, the website will have six main categories that will appear in a “menu bar” all across the different pages: SilkRoad, Projects, Gallery, Donations, Contact, and Blog. All of these categories will have subcategories that will appear as a drop down menu when the user hovers over each category. A line will appear under the category that the user clicks or hovers over to indicate where they are on the website. The user will not have an option to click on the main categories, and the user can only be able to click on the subcategories. For example, if the user wants to see the “Projects” category, the user will have to click on “past,” “current,” or “future” to see the actual contents. When the user clicks on “past” and reaches the bottom on the content, there will be links that will guide the user to the “current” page or the “future” page (Details on the subcategories will be in the paragraph below). We could have had all of the subcategories of the main category in one page. For example, for the “Projects” category, we could have had the “past,” “current,” and “future” all on one page. However, after careful consideration, we decided to separate all the subcategories into separate pages. It would be easier for the webmaster (admin) to edit each category separately, since he or she will be editing only one at a time. Moreover, if there is a problem after editing the page, only one section will have a problem instead of the entire main category. It will also make it easier for the users to navigate through each page, without having to scroll up and down too much. Overall, for the design purposes and the reasons above, we will have each subcategory into separate pages, and the users cannot click on the main categories, but use it to hover over to click on the subcategories.

By clicking the "Members" top-level link, the menu bar will change to different categories: Profile, Files, Admin. Each category will have different subcategories. Profile will have View Profile and Edit Profile. Files will have Upload Files and Download Files. Admin will have Manage Members, Pending Requests, and Change Contents. The overall layout will be the same as the main website, but Members (and admins) will have more pages that they can access. Since these pages are irrelevant to ordinary users, we decided to display them only to the members when they are logged in. By having a separate menu bar, it also distinguishes which pages are allowed to members-only.

**Search**

There will be a search bar on the top all throughout the website. This will allow the users to find the information that they need in a fast and easy way. Once the user inputs a word, we will create a search functionality that searches through tags, captions, and contents on the website. The results will be displayed on a different page, and the users will be directed to where they want to be when they click on the link of their choice.

**Login**

There will be a login system for the users. When the user clicks on “Login” there popup on the screen using Lightbox via Javascript. Once the user and password validates, the Login on the top will change to Log Out. This will also create a new button and give more access to those members with authorities. For those without login information, they can register by clicking on the link below the user/password box. If they forgot their username they click to send an e-mail with their username or if they lost their password they can chose to send an email to reset on this page.

**Member**

When the member logs in, they would be able to access the Members page located on the top of the page next to the login. The members’ page is divided up into three main categories: Profile, File, and Admin.

**Profile**

When the members hover over Profile, they can view their information on their name, user ID and password, when they joined the organization, and membership status. They can also see the documents that they uploaded. Another subcategory of Profile is “edit profile.” The user can edit all their information on this page.

**File**

File has two subcategories: view file and upload file. All the members with access to this page will can view the files. Those who have are granted access by the admin to upload files can upload the files.

Both the pages will look like a forum style. The most recent posts will be on the top, and the users can download the information from the database from “view the files” page. For the “upload file” page, the member will be asked to provide all the information including appropriate names before successfully uploading the document.

**Admin**

Admin category will have three subcategories: Change Membership, Approve Requests, and Change Contents. These are the three main functionalities that will be performed on Admin page. Change membership subcategory will display a table of all the members and their information. There will be a checkbox at the end of each row, so that the admin can select multiple members and change their level/permission or delete their info (expel membership). Approve requests will display a table of all the members who requested a change in their level. It will show their current level and the level that they desire. There will be a checkbox and an “Approve” button next to each of them so that the admin can approve multiple of them or one by one. Change contents will have sub-menus of Donations, Projects, Gallery, and Blog. By clicking different pages, the admin will be directly to another page with different forms so that the admin can easily change information of those pages. Different pages will display different forms depending on what kind of information it contains.

All six categories in the “menu bar” will have subcategories.

**SilkRoad**

SilkRoad category will have four categories: Mission&Purpose, History, Executive Board, Chapter Presidents. Users who visit this website for the first time will want to know more about the organization. This main category serves this purpose. The users will be able to click on the “Mission&Purpose” and know what ideology or belief the organization is based on. This will state the mission statements and purpose of the organization. The users will also click on “History” to find more about the organization’s history. There will be a timeline on when the organization was first founded, establishment of new chapters, and the highlights for each project. On the Executive Board page, there will be pictures and descriptions of each executive board member. Similarly, the Chapter Presidents will also have pictures and descriptions of each individual. The reason for these subcategories is for the users to know the leaders who are leading this organization. All of these subcategories explain what the organization is “about.” We believe that it will be easy for the users to find these four subcategories in the “SilkRoad” section.

These four pages will consist primarily of text and pictures. The users can access all the other sites by clicking on the dropdown menu for “SilkRoad” on the “Menu bar” All of the contents on this website will be similar to what is on the [www.silkroadinc.org](http://www.silkroadinc.org) website.

**Projects**

Projects category will divide up into Past, Current, and Future. We placed these three into the “Projects” category since each of them will explain in detail about the “Past” projects, “Current” projects, and “Future” Projects. Since SilkRoad is a non-profit organization that has annual projects that represent their organization, we believed that this should be one of the main categories. For the online users who are interested in knowing more about what the organization does and have done, they can find out specific information about the projects through the “Projects” category. We divided the subcategories into three parts because there are more than six past projects so far, and with proper descriptions and pictures the page will get too long if we combine the current and future projects into one page. It is easier and clearer for the users to see the specific projects in at one sight by clicking on “past,” “current” or “future” projects. We want the users to easily navigate through the website and get the feeling that our organization is professional.

For the specific categories of the website, past project page will contain pictures and descriptions of the past school building projects and shelter building for countries that faced natural disasters. Since there is only one big project every year, it would not be too much for users to scroll down to read the information. For “Current” project subcategory, the users will also be able to see pictures of the new site and location as a moving picture using javascript. Underneath the picture will be a short description on the specific details about the project. For more details, such as specific finances, the users can download files for how much money the organization needs to build the finish the project. For the “Future” project subcategory, we will have a list of all the upcoming projects.

**Gallery**

The gallery will have region, chapter, and activities as the subcategories. By clicking on “Region”, the users can choose East Asia, South Asia, South East Asian, and America. They will be able to see the gallery by the region of their choice. By using AJAX, further options will appear next to the last dropdown menu bar, until there are no more options to choose from. Since SilkRoad is an organization that has chapters in different regions and projects across the world, it would be easier for the users to navigate based on region. As for the pictures for each chapter, the users can click on the “Chapters” subcategory and choose the chapter of their choice to see the entire gallery. Users who want to participate in certain chapters could see pictures from those at their school or close to where they live. For those who are interested in SilkRoad activities, they can click on “Activities.” The activities will divide into fundraisers and projects.

There will be a map on the “Region” page that the users can click on. If they click on America, they will see pictures that were taken in America. If they click on China, they will see pictures that were taken in China. There will be a sort menu above the pictures, and the users can choose which continent, country, and date the pictures were taken (Users can choose a time frame: December 2010 to January 2011). For the “Chapter” gallery page, there will be a picture of the chapter (like an album in Project 3) that will link to the specific gallery for each chapter. There will be a sort menu that the users can use. They can use the drop down menu for the different chapters, and choose a time frame. Similarly, for “Activities,” there will be two pictures that the user can click: fundraisers and projects. Depending on what the user clicked, they can see all the pictures from fundraisers and projects. Using the drop down sort menu and time frame (giving a range for the dates), the user can limit the pictures they want to see.

**Donate**

The Donate category will divide into two subcategories: “Help whom?” and “Donation.” These two categories fit into these two topics. The “Help whom?” is a game designed for those who don’t know what project they want to fund. Although they can donate to the organization, and let the organization to decide on where they need to use the money, this game is to help the users to decide and actively participate on who to help and actually see who they are helping and how they are being helped. This game will eventually lead the user to a specific donate page. For users who have a sense of where to donate, they can click on the “Donation” page. The user can use to donate to the organization, fund an entrepreneur, and fund school materials.

The client doesn’t want to make the “Donate” category too complicated. They said the more complicated it is, the less people are likely to donate. SilkRoad wants three different categories. For those who don’t know who to donate to, we will create a short ‘donation game’ to see who the user is more willing to donate to. There will be short AJAX/PHP generated questions that the users will answer until a specific clause is reached. At the end of the game, there will be a donation that the user may be interested in. The user can use donate an amount of his or her choice from that page. “Donation” page will have three pictures that the user can click on. One will be for the organization, second one will be for a microfinance project to fund an entrepreneur, last will be to fund for school materials. There will be pictures that depict each of these categories. Once the user clicks on this, the user can put in the amount of his or her choice. The user will be able to see how much people have donated so far on the website.

**Contact**

The contact page will have subcategories of Facebook, Twitter, and Contact. For those who miss the little icon on the top left corner, they can click on “Contact” and reach the organization through Facebook or Twitter. There will also be a Contact form that will ask for a proper e-mail, name, and content for questions and comments. All of these three categories are relevant in that it allows the users to contact the organization through social media and e-mail.

The Facebook and Twitter subsections will actually link to the SilkRoad Facebook and Twitter page. Hopefully this would allow the users to like or follow the organization’s page. The contact form will have a simple form with e-mail, name, and content. We will use AJAX to valid the forms and return error messages if needed.

**Blog**

Blog category will have subcategories of each chapter. Since only Chapter Presidents (Permission = 2 🡪 we have to add new permission level in the database section) can write blog posts, it will be most reasonable to divide the blog page by chapter presidents. If Chapter Presidents are logged in, there will be a “write” button on the very top right corner of every blog page, so that they can easily write new posts. Even if they are on different subcategories, the “write” button will be still on the top so that they can automatically write on that specific subcategory (chapter).

On each page of Blog category, all the blog posts written in that subcategory will be displayed, from the most recent to the oldest. There will be different page numbers both on the top and bottom of the page so that users can easily navigate through the blog posts (Example: 1 . 2 . 3 . … >> last). Default display will be 10 blog posts per page, but we will also have a dropdown bar to allow the user to select how many blog posts they want to view per page (1, 5, 10, 25, 50). Blog page will also have a search input box on the top to let the users search through blog posts only. By clicking advanced search button, they will be able to search by Author, Title, Contents, Date, or other attributes of the blog post.

Each blog post will have “Go to Top” link at the end so that the user can go to the top of the page easily (to search or navigate through different pages). There will be comment section on the bottom of each page also. It will be only visible when the user click “View Comment” link, because it will be too cluttered if there are a lot of comments for each blog. Writing comment section will be only visible to logged in members. If not logged in, it will say “Please log in (with a link) to write a comment.” If logged in, it will automatically show their name and first 4 letters of their username, followed by leftover number of asterisks. (Example: Firstname Lastname (abcd\*\*\*)). There will be a text area to write their comments and submit button to submit. After the comment is submitted, it will be displayed on the bottom of the blog post. If the logged-in user writes a comment, it will have “edit” and “delete” buttons next to that comment. Admin and Chapter President will have these buttons for any comment.

On the right side of every blog page, there will be a separate box for Tags. When writing a blog post, people can add tags or choose from the list of tags already used before. Example: Piazza. Tag box will show tags in different font sizes depending on the popularity of the tags. (Top 1-5 will be large, top 6-10 will be medium, leftover will be small). The tag box will be always on the right side, even if the user scrolls up and down the blog page, so that the user can easily navigate to different pages.

**ER Diagrams**

The ER diagrams follows the standards described in Database Management Systems by Professor Johannes Gehrke. The book version has bolding instead of double lines and bolded arrows instead of rounded arrowheads. Here is also a summary of the book ER diagram model - http://www.usna.edu/Users/cs/adina/teaching/si440/fall2007/slides/SI440\_Set2\_ERModel.pdf

Photos

Activities\_IN

Chapters

Chapters\_IN

Countries\_IN

Countries

Regions

Regions\_IN

Activities

Receives

Makes

LikedPost

Donations

Posted

IsComment

Posts

Comments

Users

Tagged

Causes

Tags

Files

Projects

**Database Design (Relational Database Schema):**

The design contains 20 tables: Activities, Photos, Activities\_IN, Chapters, Chapters\_IN, Countries\_IN, Countries, Regions, Regions\_IN, Users, Post, LikePost, IsComment, Comment, Tagged, Tags, Cause, Donations, Files, Projects

A well-designed database system gives us the tools to add more functionality to the SilkRoad website. First of all, by having an internal database backend system, The SilkRoad will behave as a dynamic website that is tailored to individual users. It will also allow us to implement search features that can query across any data content we have already stored in the database. Furthermore, the users would be able to actively interact by changing and uploading documents and files into the database. Only one admin will have access to deleting the files and granting certain members access to secure data. Our database will contain information regarding users, photos and albums, blog posts and blog comments, donations made and by whom. Below is a table translation of the ER diagrams. For each table, a description of the attributes, keys, and constraints are given.

**Activities:**

1. Activity\_ID - Int (Primary Key and Autoincremented)

2. Name - Varchar(256)

The Activity \_ID is the unique id that will serve as the primary key to this table. The Name field is the name of the activity. This table represents different events, such as Asian Night and Pocky Fundraiser event.

**Photos:**

1. Photo\_ID - Int (Primary Key and Autoincremented)

2. Name - varchar

2. Caption - Varchar

3. Url - Varchar

4. Date\_Taken - Date

The Photo\_ID is the unique id that will serve as the primary key to this table. The Caption field is the caption associated with the current picture, the Url field is the url of where the picture is, and the Date\_Taken field is the date of when the photo was taken.

**Activities\_IN:**

1. Activity\_ID - Int (Foreign Key from Activities Table)

2. Photo\_ID - Int (Foreign Key from Photos Table)

Entries in this table show the relationship between Photos and Activities. If there is a entry that links activity #1 to photo #3 for example, then this shows that photo #3 is in activity #1. The Activity\_ID will match a single entry in the Activities table and the Photos\_ID will match a single entry in the Photos table.

**Chapters:**

1. Chapter\_ID - Int (Primary Key and Autoincremented)

2. Name - Varchar(256)

The Chapter\_ID is the unique id that will serve as the primary key to this table. The Name field is the name of the chapter. This table represents different chapters, such as the Cornell Chapter for Silkroad. It will associate photos with chapters, so that photos can be filtered when the user is looking through them.

**Chapters\_IN:**

1. Chapter\_ID - Int (Foreign Key from Activities Table)

2. Photo\_ID - Int (Foreign Key from Photos Table)

Entries in this table show the relationship between Photos and Chapters. Very similar to Activities\_IN table, but just for chapters.

**Countries\_IN:**

1. Country\_ID – Int (Foreign Key from SubRegions Table)

2. Photo\_ID - Int (Foreign Key from Photos Table)

Entries in this table show the relationship between a photo and which country the photo is in. There is a constraint here that says that a Photo must belong to at least one subregion.

**Countries:**

1. Country\_ID – Int (primary key)

2. Name – VarChar

Entries in this table represent a Country. Each photo is associated with a region. Regions are further classified into countries. The name field just names the countries, like Korea subregion inside of the Asia region.

**Regions\_IN:**

1. SubRegion\_ID – Int (Foreign Key from SubRegions Table)

2. Region\_ID – Int (Foreign Key from Regions Table)

Entries in this table represent relationships between SubRegions and Regions. There is a special constraint here that says that each SubRegion belongs to exactly one Region.

**Regions:**

**1.** Region\_ID – Int (primary key)

**2.** Name – VarChar

Entries in this table represent a Region. Each photo is associated with a region. Regions are further classified into subregions. The name field just names the region, like Korea subregion inside of the Asia region.

**Users:**

1.User\_ID - int (primary key, auto increment)

2. Username – Varchar(20)

3. Hashpwd - Char(64)

4. Name – Varchar(30)

5. Email – Varchar(40)

6. Permission - Int

Entries in this table represent users who can log into the website. The Username field will be the primary key, the Hashpwd field is the hashed version of the user's password (hashing will be done as discussed in class), the Name field is the name of the user, and the Permission field will be a integer that signifies how much permission the user has over the whole website (Member = 1, ChapterPresident = 2, Admin = 3).

**Posts:**

1. Post\_ID - Int (Primary Key and Autoincremented)

2. User\_ID - Int (Foreign Key from Users)

3. Post – Varchar

4. Url – Varchar

5. Date – DateTime

6. NumLiked - Int

Entries in this table represent a blog post. A blog post will keep a record of when and who posted the entry. Furthermore, it will have the text content stored in the Post column and the user could also decide to add a picture, which will be stored in the url. The Date field is crucial to the blog because we will need to order by this field to keep the comments and blog posts in the correct order. NumLiked field will keep track of number of likes. Every time a logged in member likes/unlikes a blog posts, it will automatically update this field. Aggregation function could have been used, but since it will require the database to calculate the total number of likes in LikePost table, it will slow down the page too much. Instead, we will keep a separate variable.

**LikePost:**

1.       Post\_ID - Int (Foreign Key from Posts Table)

2.       User\_ID - Int (Foreign Key from Users Table)

Entries in this table create a relationship between posts and users who likes it. A post is the blog post and every time a user likes it, it will add their User\_ID to this table. The post will look through this table to determine if there will be a “like” button or “unlike” button, depending on whether or not this person has already liked this post.

**Comments:**

1. Post\_ID – Int (Foreign Key from Posts Table)

2.       User\_ID – Int (Foreign Key from Users)

3.       Post – Varchar

4.       Date – DateTime (Partial Key)

Entries in this table represent a comment. A comment is just any text that other user may post onto an initial post or respond to another comment. A comment post is just text that says something about the post in question. The Date field is crucial to the blog because we will need to order by this field to keep the comments and blog posts in the correct order. The Date field is also a partial key part of a weak entity set.

**Tagged:**

1. Post\_ID - Int (Foreign Key from Posts Table)

2. Tag\_ID - Int (Foreign Key from Tags Table)

Entries in this table create a relationship between posts and its tags. A post is the blog post and a tag is a tag associated with that blog post. There can be multiple tags to one blog post. Multiple blog posts can have same tag as well.

**Tags:**

1. Tag\_ID - Int (Primary Key)

2. Tag\_name – Varchar

Entries in this table represent a tag. A tag is just any text that the author may have created or used when they wrote each blog post.

**Causes:**

1. Cause\_ID - Int (Primary Key and Autoincremented)

2. Name – Varchar(100)

3. Description - Text

4. GoalAmount - Double

5. CurrentAmount - Double

6. StartDate - Date

7. EndDate - Date

8. Cover\_URL – Varchar(50)

Entries in this table will represent different causes of the donation that are asking for donations. Each Cause will have a unique ID number to represent it. It will also have Name and Description field to describe itself. GoalAmount and CurrentAmount fields have double-type value to indicate the goal and current amount of donations this cause has collected. StartDate and EndDate are also important to indicate the time range of this cause. Cover\_URL field will have url/filename of the cover photo it will use.

**Donations:**

1.    Donation\_ID – Int (Primary, AutoIncrement)

2. User\_ID - Int (Foreign Key from Users)

3.       Cause\_ID - Int (Foreign Key from Causes Table)

4.       Amount - Double

5.       Date - Date

Entries in this table will represent a donation. A donation will consist of an amount donated from a specific donor. Donor will be indicated by using User\_ID (from users) or -1 if not logged-in). The donation to a specific cause will be indicated using the cause\_id. Amount field will be used to indicate how much money was donated. Date field will be used for the date donation was made.

**Files:**

1. File\_ID - Int (Primary Key and Autoincremented)

2. Name - Varchar(50)

3. UploadDate - Date

4. Caption - Text

5. User\_ID - int (foreign key from Users)

The File\_ID is the unique id that will serve as the primary key to this table. The Name field is the name of the file, the Uploaded field is the date the file was uploaded. User\_ID is the foreign key from the Users table, so that only the user who uploaded this file AND the admins can edit/delete this file.

**Projects:**

1. Project\_ID - Int (Primary Key and Autoincremented)

2. Name: Varchar (50)

3. Caption – Text

4. Cover\_Url – Varchar (50)

5. PCF – int

6. Url – Varchar (50)

The Project\_ID is the unique id that will serve as the primary key to this table. The Name field is the name of the project. The Caption field is the caption associated with the current project, the Cover\_Url field is the url of the cover image of this project. PCF field describes if this project is past (1), current (2), or future (3) project.

**PHP INTERACTIVITY**

**Database Query Functions**

// pass in one or more sql statement as parameter and the result is returned

function multi\_query($sql) {

$mysqli = new mysqli('localhost', 'Jirex', 'xek5hsh7vhk', 'info230\_SP13FP\_Jirex');

if($mysqli->errno) {

print($mysqli->error);

exit();

}

$result = $mysqli->multi\_query($sql);

$mysqli->close();

return $result;

}

// pass in exactly one sql statement as parameter and the result is returned

function query($sql) {

$mysqli = new mysqli('localhost', 'Jirex', 'xek5hsh7vhk', 'info230\_SP13FP\_Jirex');

if($mysqli->errno) {

print($mysqli->error);

exit();

}

$result = $mysqli->query($sql);

$mysqli->close();

return $result;

}

**Photo Gallery Functions**

// php function that gets html to show all photos in the database

// have same function for specific filters, like Country\_IN and Region\_IN, and even search results

// the sql variable will just change for these other filters to have the filter in the Where clause

function displayAllPhotos() {

$sql = "SELECT \* FROM Photos";

$result = query($sql);

generatePhotoTable($result);

}

// sql to delete an album with aid

function deleteAlbum($aid) {

$sql = "DELETE FROM Albums WHERE Album\_ID = $aid";

query($sql);

}

// sql to delete a photo with aid and pid

function deletePhoto($aid, $pid) {

$sql = "DELETE FROM Photos WHERE Photo\_ID = $pid;";

$sql .= "DELETE FROM Photos\_IN WHERE Photo\_ID = $pid AND Album\_ID = $aid";

multi\_query($sql);

}

// sql to change album title

function changeAlbumTitle($aid, $title) {

$sql = "UPDATE Albums SET Title = \"$title\" WHERE Album\_ID = $aid";

query($sql);

}

// sql to change photo caption

function changePhotoCaption($pid, $caption) {

$sql = "UPDATE Photos SET Caption = \"$caption\" WHERE Photo\_ID = $pid";

query($sql);

}

//creates an html table with all the photos present in the result parameter

//this will be used for basically any time we need to show a collection of photos

//the result parameter will just be all the photos returned from a specific db query

function generatePhotoTable($result) {

echo "<table class=\"table\">";

$counter = 0;

while ($array = $result->fetch\_assoc()) {

$pid = $array['Photo\_ID'];

$name = $array['Name'];

$caption = $array['Caption'];

$url = $array['URL'];

$date = $array['Date\_Taken'];

$thumbnail = $array['Thumbnail'];

if($counter % 3 == 0) {

echo "<tr class=\"\">";

}

echo "<td class=\"span3\">";

echo "<a class='span3' href='$url' rel='lightbox' title='$caption'>";

echo "<img class='span3' src='$url' alt='$caption' />";

echo "</a>";

echo "</td>";

if($counter != 1 && $counter-1 % 3 == 0) {

echo "</tr>";

}

$counter++;

}

echo "</table>";

}

//handling upload of photos

//this will handle all the form php interaction with addPhoto.php

//add regions, countries, chapters, and activities will follow the same upload procedures

function handlePhotosUpload() {

$target\_path = "uploads/";

if(!isset($\_FILES['uploadedfile'])) {

echo "<div class='alert alert-error'>There was an error uploading the file, please try again!</div>";

} else {

$image\_name = basename( $\_FILES['uploadedfile']['name']);

// uploads/filename.extension

$target\_path = $target\_path . $date->getTimestamp() . "\_" . $image\_name;

$allowedExts = array("jpeg", "jpg");

$extension = end(explode(".", $\_FILES["uploadedfile"]["name"]));

$name = mysql\_real\_escape\_string($\_POST['name']);

$caption = mysql\_real\_escape\_string($\_POST['caption']);

$date\_taken = mysql\_real\_escape\_string($\_POST['date']);

if(trim($caption) == '' || !preg\_match('/^[0-9a-zA-Z\s]+$/', $caption)) {

echo "<div class='alert alert-error'>Please provide a valid caption!</div>";

} else if(trim($name) == '' || !preg\_match('/^[0-9a-zA-Z\s]+$/', $name)) {

echo "<div class='alert alert-error'>Please provide a valid name!</div>";

} else if (!((($\_FILES["uploadedfile"]["type"] == "image/jpeg") || ($\_FILES["uploadedfile"]["type"] == "image/jpg"))

&& in\_array(strtolower($extension), $allowedExts))) {

echo "<div class='alert alert-error'>Please upload only .jpg files!</div>";

} else if ($caption != '' && move\_uploaded\_file($\_FILES['uploadedfile']['tmp\_name'], $target\_path)) {

$thumb\_path = "thumbs/" . $date->getTimestamp() . "\_" . $image\_name;

make\_thumb($target\_path, $thumb\_path, 300, 200);

$activityid = mysql\_real\_escape\_string($\_POST['activity']);

$countryid = mysql\_real\_escape\_string($\_POST['activity']);

$chapterid = mysql\_real\_escape\_string($\_POST['chapter']);

// database sql statements

$query = "INSERT INTO Photos VALUES(NULL, '$name', '$caption', '$target\_path', '$thumb\_path', '$date\_taken');";

$query .= "INSERT INTO Countries\_IN VALUES('$countryid', (SELECT Photo\_ID FROM Photos WHERE Url = '$target\_path'));";

if ($activityid != -1) {

$query .= "INSERT INTO Activities\_IN VALUES('$activityid', (SELECT Photo\_ID FROM Photos WHERE Url = '$target\_path'));";

}

if ($chapterid != -1) {

$query .= "INSERT INTO Chapters\_IN VALUES('$chapterid', (SELECT Photo\_ID FROM Photos WHERE Url = '$target\_path'));";

}

multi\_query($query);

//header("location: album.php?aid=$albumid");

header("location: addPhoto.php");

} else{

echo "<div class='alert alert-error'>There was an error uploading the file, please try again!</div>";

}}}

**Instant Search Functions**

// generates a table of all photos that fit the search caption criteria

function generateSearchTable($search\_caption) {

echo "<h1>Search Results: \"$search\_caption\"</h1></br>";

echo "<table class=\"table\">";

//change this sql statement depending on what kind of search we want to do

$sql = "SELECT Photo\_ID, Url, Caption, Thumbnail FROM Photos\_IN NATURAL JOIN Photos WHERE Photos.Caption LIKE \"%$search\_caption%\"";

….call the function to print html table of photo results from above

// does a dynamic ajax request ever time user types anything

if(isset($\_GET['caption'])) {

$search = mysql\_real\_escape\_string($\_GET['caption']);

if(!preg\_match('/^[0-9a-zA-Z\s]+$/', $search)) {

echo "<div class='alert alert-error'>Please provide a valid caption with letters and numbers only!</div>";

} else {

generateSearchTable($search);

}

}

//Javascript and Ajax for search

$("input[name='caption\_search']").keyup(function(){

$.ajax({

url : "searchAjax.php?caption=" + $(this).val(),

success : function(response){

$("#searchResults").html(response);

}

});

});

**Blogs**

//requires the post index number, and outputs the result in an associative array.

function getSortedBlog($sortMethod){

$blogList=array();

$mysqli = new mysqli("localhost", "Jirex","xek5hsh7vhk", "info230\_SP13FP\_Jirex");

if (mysqli\_connect\_errno($mysqli))

{

echo "Failed to connect to MySQL: " . mysqli\_connect\_error();

}

//depending on what sorting order selected, it will return a list sorted by that order

if ($sortMethod=="index"){

$query=sprintf("SELECT \* FROM Posts ORDER BY Post\_ID");

$result=$mysqli->query($query);

if (!$result) {

$message = 'Invalid query: ' . mysql\_error() . "\n";

$message .= 'Whole query: ' . $query;

die($message);

}

}

if ($sortMethod=="date"){

$query=sprintf("SELECT \* FROM Posts ORDER BY Date DESC");

$result=$mysqli->query($query);

if (!$result) {

$message = 'Invalid query: ' . mysql\_error() . "\n";

$message .= 'Whole query: ' . $query;

die($message);

}

}

//fetches row in the array and puts them in the associative arrayList

while($row = mysqli\_fetch\_array($result)){

array\_push($blogList,$row);

}

mysqli\_close($mysqli);

return $blogList;

}

//returns a associative list for a specific user based on their ID

function getUser($userID){

$userList=array();

$mysqli = new mysqli("localhost", "Jirex","xek5hsh7vhk", "info230\_SP13FP\_Jirex");

if (mysqli\_connect\_errno($mysqli))

{

echo "Failed to connect to MySQL: " . mysqli\_connect\_error();

}

$query=sprintf("SELECT \* FROM Users WHERE User\_ID=".$userID);

$result=$mysqli->query($query);

if (!$result) {

$message = 'Invalid query: ' . mysql\_error() . "\n";

$message .= 'Whole query: ' . $query;

die($message);

}

while($row = mysqli\_fetch\_array($result)){

$userList=$row;

}

mysqli\_close($mysqli);

return $userList;

}

function getTags($postID){

$tagList=array();

$mysqli = new mysqli("localhost", "Jirex","xek5hsh7vhk", "info230\_SP13FP\_Jirex");

if (mysqli\_connect\_errno($mysqli))

{

echo "Failed to connect to MySQL: " . mysqli\_connect\_error();

}

$query=sprintf("SELECT \* FROM Tagged NATURAL JOIN Tags WHERE Post\_ID=".$postID);

$result=$mysqli->query($query);

if (!$result) {

$message = 'Invalid query: ' . mysql\_error() . "\n";

$message .= 'Whole query: ' . $query;

die($message);

}

while($row = mysqli\_fetch\_array($result)){

array\_push($tagList,$row["Tag\_Name"]);

}

mysqli\_close($mysqli);

return $tagList;

}

//given PostID, fetches all the comments for that post ordered by dateTime

function getComments($postID){

$commentsList=array();

$mysqli = new mysqli("localhost", "Jirex","xek5hsh7vhk", "info230\_SP13FP\_Jirex");

if (mysqli\_connect\_errno($mysqli))

{

echo "Failed to connect to MySQL: " . mysqli\_connect\_error();

}

$query=sprintf("SELECT \* FROM Comments WHERE Post\_ID=".$postID." ORDER BY Date DESC");

$result=$mysqli->query($query);

if (!$result) {

$message = 'Invalid query: ' . mysql\_error() . "\n";

$message .= 'Whole query: ' . $query;

die($message);

}

while($row = mysqli\_fetch\_array($result)){

array\_push($commentsList,$row);

}

mysqli\_close($mysqli);

return $commentsList;

}

**Blog summary functions**

//outputs the blog summary

function displayBlogSummary($beginningIndex,$endingIndex,$sortedList){

//makes sure it doesnt not go out of bound

if ($endingIndex>(sizeof($sortedList)-1)){

$endingIndex=(sizeof($sortedList));

}

For ($count=$beginningIndex;$count<=$endingIndex-1;$count++){

DisplaySummaryTitle($sortedList[$count]["Title"]);

DisplaySummaryAuthor($sortedList[$count]["User\_ID"]);

DisplaySummaryDate($sortedList[$count]["Date"]);

DisplaySummaryTags($sortedList[$count]["Post\_ID"]);

DisplaySummaryPost($sortedList[$count]["Post"]);

//DisplaySummaryCoverPic($sortedList[$count]["Url"]);

DisplaySummaryNumLikes($sortedList[$count]["NumLiked"]);

DisplaySummaryUrl($sortedList[$count]["Post\_ID"]);

}

}

//displays the formated title of the summary

function DisplaySummaryTitle($title){

print("<div class=\"blogSummeryTitle\">".$title."</div>");

}

//displays the formated author of the summary

function DisplaySummaryAuthor($authorID){

$author=getUser($authorID);

print("<div class=\"blogSummeryAuthor\">Written by: ".$author["Name"]."</div>");

}

//implode all the tags, seperated by comma

function DisplaySummaryTags($postID){

$tagList=implode(",",getTags($postID));

print("<div class=\"blogSummeryTags\">".$tagList."</div>");

}

//displays the formated paragraph of the blog summary limit to 100 ch1aracters)

function DisplaySummaryPost($post){

$post=substr($post,0,100);

print("<div class=\"blogSummeryPost\">".$post."...</div>");

}

//displays the formated cover photo of the summary

function DisplaySummaryCoverPic($url){

print("<div class=\"blogSummeryCoverPic\">".$url."</div>");

}

//displays the formated date of the summary

function DisplaySummaryDate($date){

print("<div class=\"blogSummeryDate\">&nbsp on ".$date."</div>");

}

//displays the formated number of likes of the summary

function DisplaySummaryNumLikes($numLikes){

print("<div class=\"blogSummeryNumLikes\">Total Likes: ".$numLikes."</div>");

}

//displays the link for the summary that will pass info via URL for the blog page to process (using GET)

function DisplaySummaryUrl($postID){

print("<div class=\"blogSummeryUrl\"><a href=\"blog.php?type=blog&blogID=".$postID."\">See the full blog<a></div>");

}

**Section for displaying the full blog given arguments that was passed through the url (GET)**

//given blog id, get the content of the blog

function getSingleBlog($postID){

$post=array();

$mysqli = new mysqli("localhost", "Jirex","xek5hsh7vhk", "info230\_SP13FP\_Jirex");

if (mysqli\_connect\_errno($mysqli))

{

echo "Failed to connect to MySQL: " . mysqli\_connect\_error();

}

$query=sprintf("SELECT \* FROM Posts WHERE Post\_ID=".$postID);

$result=$mysqli->query($query);

if (!$result) {

$message = 'Invalid query: ' . mysql\_error() . "\n";

$message .= 'Whole query: ' . $query;

die($message);

}

while($row = mysqli\_fetch\_array($result)){

$post=$row;

}

mysqli\_close($mysqli);

return $post;

}

//dipsplays the blog along with it's comments

function displayBlog($postID){

$post=getSingleBlog($postID);

$tagList=getTags($postID);

$commentsList=getComments($postID);

$authorID=$post["User\_ID"];

DisplayBlogCoverPic($post["Url"]);

DisplayBlogTitle($post["Title"]);

DisplayBlogAuthor($authorID);

DisplayBlogDate($post["Date"]);

DisplayBlogTags($postID);

DisplayBlogPost($post["Post"]);

DisplayBlogNumLikes($post["NumLiked"]);

displayComments($commentsList);

}

//displays the comments

function displayComments($commentsList){

print("Comments Made By Members</br>");

foreach ($commentsList as $row){

$commenterName=getUser($row["User\_ID"]);

$date=$row["Date"];

$post=$row["Post"];

print("<div class=\"DisplayCommentsUserName\">".$commenterName["Name"]."</div>");

print("<div class=\"DisplayCommentsDate\">".$date."</div>");

print("<div class=\"DisplayCommentsPost\">".$post."</div>");

}

}

//displays the formated title of the blog

function DisplayBlogTitle($title){

print("<div class=\"blogTitle\">".$title."</div>");

}

//displays the formated author of the blog

function DisplayBlogAuthor($authorID){

$author=getUser($authorID);

print("<div class=\"blogAuthor\">Written by: ".$author["Name"]."</div>");

}

//implode all the tags, seperated by comma

function DisplayBlogTags($postID){

$tagList=implode(",",getTags($postID));

print("<div class=\"blogTags\">".$tagList."</div>");

}

//displays the formated paragraph of the blog

function DisplayBlogPost($post){

print("<div class=\"blogPost\">".$post."</div>");

}

//displays the formated cover photo

function DisplayBlogCoverPic($url){

print("<div class=\"blogCoverPic\">".$url."</div>");

}

//displays the formated date

function DisplayBlogDate($date){

print("<div class=\"blogDate\">&nbsp on ".$date."</div>");

}

//displays the formated number of likes

function DisplayBlogNumLikes($numLikes){

print("<div class=\"blogNumLikes\">Total Likes: ".$numLikes."</div>");

}

## pagination functions of blog

//display the appropriate # of links

function DisplayBlogSummaryPagination($numPerPage){

if (!is\_int($numPerPage)){

$numPerPage=5;

}

$totalEntries=getTotalEntries();

$final=$totalEntries[0]/$numPerPage;

$pageList=Array();

for ($count=0;$count<=$final;$count++){

array\_push($pageList,"<a href=\"blog?type=blogsummary&selectedpage=".($count+1)."&numperpage=".$numPerPage."\">".($count+1)."</a>");

}

implode("&nbsp",$pageList);

print("Pages:");

print(implode("&nbsp",$pageList));

}

//opens the database and counts how many blog posts exists

function getTotalEntries(){

$mysqli = new mysqli("localhost", "Jirex","xek5hsh7vhk", "info230\_SP13FP\_Jirex");

if (mysqli\_connect\_errno($mysqli))

{

echo "Failed to connect to MySQL: " . mysqli\_connect\_error();

}

$query=sprintf("SELECT COUNT(Post\_ID) FROM Posts");

$result=$mysqli->query($query);

if (!$result) {

$message = 'Invalid query: ' . mysql\_error() . "\n";

$message .= 'Whole query: ' . $query;

die($message);

}

while($row = mysqli\_fetch\_array($result)){

return $row[0];

mysqli\_close($mysqli);

}

}

**Top tags function**

//gets the top tags (5 or 10)

function getTopTags(){

$tagList=Array();

$mysqli = new mysqli("localhost", "Jirex","xek5hsh7vhk", "info230\_SP13FP\_Jirex");

if (mysqli\_connect\_errno($mysqli))

{

echo "Failed to connect to MySQL: " . mysqli\_connect\_error();

}

$query=sprintf("SELECT Tag\_Name, Count(Tag\_ID),Tag\_ID FROM Tags NATURAL JOIN Tagged GROUP BY Tag\_ID ORDER BY Count(Tag\_ID) DESC LIMIT 10");

$result=$mysqli->query($query);

if (!$result) {

$message = 'Invalid query: ' . mysql\_error() . "\n";

$message .= 'Whole query: ' . $query;

die($message);

}

while($row = mysqli\_fetch\_array($result)){

array\_push($tagList,$row );

}

mysqli\_close($mysqli);

return $tagList;

}

//get the list of top 10 tags and displays different tags in different sizes corresponding to their popularity. ALso, it displays the tags in random order (visually).

function displayTags($tagList){

print("<div class=\"topTags\">");

$finalList=Array();

$lastTag=end($tagList);

$baseWeight=$lastTag[1];

//randomly reorients the list

shuffle($tagList);

foreach ($tagList as $row){

$fontSize=floatval($row[1])/floatval($baseWeight)\*15;

if ($fontSize>50){

$fontSize=50;

}

print("<a id=\"tagLink\" href=blog.php?type=blogsbytag&tagID=".$row[2]." style=\"font-size:".$fontSize."px\">".$row[0]."</a>");

}

print("</div>");

}

//given tag ID, the function gets the blog posts associated with that tag ordered by date.

function getBlogsByTag($tagID){

$blogList=Array();

$mysqli = new mysqli("localhost", "Jirex","xek5hsh7vhk", "info230\_SP13FP\_Jirex");

if (mysqli\_connect\_errno($mysqli))

{

echo "Failed to connect to MySQL: " . mysqli\_connect\_error();

}

$query=sprintf("SELECT \* FROM Tags NATURAL JOIN Tagged NATURAL JOIN Posts WHERE Tag\_ID=".$tagID." ORDER BY Date DESC");

$result=$mysqli->query($query);

if (!$result) {

$message = 'Invalid query: ' . mysql\_error() . "\n";

$message .= 'Whole query: ' . $query;

die($message);

}

while($row = mysqli\_fetch\_array($result)){

array\_push($blogList,$row);

}

mysqli\_close($mysqli);

return $blogList;

}

**Top Authors functions**

//gets the top Authors 10

function getTopAuthors(){

$authorList=Array();

$mysqli = new mysqli("localhost", "Jirex","xek5hsh7vhk", "info230\_SP13FP\_Jirex");

if (mysqli\_connect\_errno($mysqli))

{

echo "Failed to connect to MySQL: " . mysqli\_connect\_error();

}

$query=sprintf("SELECT Name,User\_ID FROM Users NATURAL JOIN Posts GROUP BY User\_ID ORDER BY Count(User\_ID) DESC LIMIT 10");

$result=$mysqli->query($query);

if (!$result) {

$message = 'Invalid query: ' . mysql\_error() . "\n";

$message .= 'Whole query: ' . $query;

die($message);

}

while($row = mysqli\_fetch\_array($result)){

array\_push($authorList,$row );

}

mysqli\_close($mysqli);

return $authorList;

}

//displays the list of top 10 Authors

function displayAuthors($authorList){

print("<div class=\"topAuthors\">");

print("Most Popular Authors</br>");

$finalList=Array();

for ($count=0;$count<sizeof($authorList);$count++){

print("<a id=\"authorLink\" href=blog.php?type=blogsbyauthor&authorID=".$authorList[$count]["User\_ID"]." >".($count+1).". ".$authorList[$count]["Name"]."</a></br>");

}

print("</div>");

}

//given author ID, the function gets the blog posts associated with that author ordered by date.

function getBlogsByAuthor($authorID){

$authorList=Array();

$mysqli = new mysqli("localhost", "Jirex","xek5hsh7vhk", "info230\_SP13FP\_Jirex");

if (mysqli\_connect\_errno($mysqli))

{

echo "Failed to connect to MySQL: " . mysqli\_connect\_error();

}

$query=sprintf("SELECT \* FROM Users NATURAL JOIN Posts WHERE User\_ID=".$authorID." ORDER BY Date DESC");

$result=$mysqli->query($query);

if (!$result) {

$message = 'Invalid query: ' . mysql\_error() . "\n";

$message .= 'Whole query: ' . $query;

die($message);

}

while($row = mysqli\_fetch\_array($result)){

array\_push($authorList,$row);

}

mysqli\_close($mysqli);

return $authorList;

}

**User Testing**

In order to create a user friendly website, we decided to conduct a “user testing” session by reaching out to two different people with different tasks. Our goal is to see if the users can easily access the necessary information. We will gather the user’s comments to implement appropriate changes. There will be two users that we will test. One will be a visitor who is not familiar with the club. Another user will be a registered member/admin who has the authority to manage the website. We created a user testing protocol to see how easily the users can navigate through the website.

Round 1: Protocol

* Visitor/New Member: This user has no familiarity with SilkRoad

a) Try to register as a member

b) Try to request to change their membership status to “Chapter President”

c) Find out how much Silkroad wants to fundraise for School Supplies

d) Find out how much SilkRoad raised for their 2nd green ribbon project

* Member/Admin: This user will have some familiarity with SilkRoad, and will be managing information

a) Login using the admin id (jirex\_admin) and password (xek5hsh7vhk).

b) Find how many members are registered on the website.

c) Try to add a new project called “3rd Green Ribbon Project”

d) Try to add a new photo of their choice

Final Phase Protocols

For the final phase protocols, we want to test the users on the overall functionalities and navigation of the website. We will divide the users into three categories: visitor, member, and admin. We hope to have more than six users so that we can have at least two users to test for each category.

* Visitor: No familiarity with the club
  + Find out SilkRoad’s history and mission
  + How to make a donation to SilkRoad
  + Find out how to participate, and ask what method that they want to use to participate
  + Find “Duct-tape man”
  + Search for blog posts by “Jirex\_Admin”
  + Change the number of blog summary displayed per page to “3,” and go to page “3”
  + Visit the first blog created
  + Visit the earliest blog made
  + View the earliest blog entry
  + Display the blogs correlating to the most popular blog tag, and view their comments
  + Display the blogs correlating to the most popular author, and view their comments

* Member: Club member who want to register and look at the members page
  + Register with the website
  + Change personal information including the password
  + Upload a word document on the members page
  + Upload a picture into the ‘chapters’ album in ‘America’
  + Find the amount of donations we received so far
  + Write a comment on two different blogpost. Then, edit one of the comment, and delete the other comment
  + Request to be the “Chapter President”

* Admin: Club admins who have authority to manage and edit files and contents
  + All of the questions for the members can apply for admin
  + Go to a previous blog post made by the admin.
  + Add a new tag to a previous blog post, change the title and the paragraph for the blogpost.
  + Add files to the website, and delete the file that the user uploaded
  + Approve the request of the person who asked to be the “Chapter President”
  + Grant access to “jylim03” as admin
  + Make a donation to one of SilkRoad’s microfinance clients and review the total amount of donations this member made to this project

**Round 1: User Testing: Background of Users**

* Visitor

Name: Ming Xing

Major: Psychology

Age: 19

Familiarity with web/computers: Blogged for 9 years, Enjoys editing photos with photoshop

Websites most visited: Facebook, Fashion and Travel blogs, Online shopping

Screen size/resolution: Macbook Pro 13inch

Browser: Google Chrome

* Admin

Name: Nancy Ha

Age:20

Major: Biology

Familiarity with web/computers: Basic web search

Websites most visited: YouTube, Facebook, BlackBoard

Screen size/resolution: Windows Desktop at Uris Library

Browser: Firefox

Round 1: User Testing Steps

1. Explain the goal of the testing to the testers

“Hello. Thank you for volunteering to user test the new SilkRoad website. Let me first introduce myself. My name is \_\_\_\_, I’m a junior in majoring in \_\_\_\_. What is your name, year, and major?

We are not testing “you” but we need you to test our “website” on how efficient our website is in terms of navigation and overall look of it. You should think aloud so that I can follow your thought process. Keep in mind that this is a test of the site and not you. Any difficulties you have in completing a task are a direct reflection on the website and not of you. We need your honest evaluation, so that we can fix any errors and improve our layout with your suggestions. We will ask you to perform some tasks. Please think out loud during the test and tell us anything that comes into your mind when you are looking through our website. Feel free to comment on any aspect of the website. If you have any questions, you should ask them now. Thank you.”

2. Guide the testers through the protocol asking them to “think out loud”

**For Ming Xing:**

a)  Can you register as a member?

b) Can you request to change your membership status to “Chapter President”?

c) How much does Silkroad want to raise for School Supplies?

d) Can you find out how much SilkRoad raised for their 2nd green ribbon project?

**For Nancy:**

a) Can you login using the admin id (jirex\_admin) and password     (xek5hsh7vhk)?

b) Can you find how many members are registered on the website?

c) Please add a new project called “3rd Green Ribbon Project.” You can go to http://www.silkroadinc.org/projects.html to get the information.

d) Please add a new photo of your choice to the gallery

3. Take notes for future reference and submission (what worked and what didn’t)

4.  Ask users for their general thought of the website.

“Thank you for participating in this user test. What are your general thoughts on the site? Are there specific things that you did not like? What did you like about the site? How does it measure up to other student organizations or non-profit organizations web pages you have visited? Your feedback on the usability of the site is incredibly valuable to us so we can make improvements. Thanks.”

**Round 1: User Testing Notes**

* Visitor/New Member (Ming Xing)

a) Try to register as a member

User tried to look for “register” button. Then quickly found “Login.”

- Would be better if there’s a separate button for Register

- Menu bar doesn’t change right away (e.g. when logged in, the menu bar still shows login, because menu bar is displayed before php functions)

b) Try to request to become a Chapter President

- She wasn’t sure where to go, and couldn’t complete the task.

c) Find out how much Silkroad wants to raise for School Supplies

- After logging in as a member, clicking on member functions or admin functions makes the original navigation bar disappear so wasn’t sure how to go back.

d) Find out how much SilkRoad raised for their 2nd green ribbon project

- User thought for 5 seconds and clicked on logo. Then she clicked on “project” and “past” and read through the information to find the answer.

General Comments:

- Texts are too plain. Use Bold/Italics and different styling’s to emphasize certain words

- Contact page’s emailing links are not very clearly that it can send emails to the Presidents by clicking their names

- Don’t like dropdown menu. Prefer the menu to have its subcategories in different color under the main categories.

-  Would prefer a cleaner modern look using black and white with Green fonts.

* Member/Admin (Nancy Ha)

a) Login using the admin id (jirex\_admin) and password (xek5hsh7vhk)

User easily navigated to the top of the page to find “login”

- Font is too small on the top menu.

- Login is too far right on a big screen, but likes how it is scales to fit the size of the screen when it is small.

b) Find how many members are registered on the website

Hovered over all the categories before clicking on “manage members”

- Subcategories, have to hover-over, which is hard

- Good to have hover-over highlight effect for the table

c) Try to add a new project called “3rd Green Ribbon Project”

- Went to projects navigation tab in order to look for relevant option, but failed to find option.

She was confused and looked at all the tabs (from left to right) and finally clicked on members. She hovered over the three new navigation items and clicked on projects then added a new project. She liked the warning on the form. Finally filled out and submitted the project.

- She thought that there were too many links to change the contents.

* + Admin>Change Contents> Edit/Delete Existing Project>

- Hard to go back to previous subcategories perhaps have to go through too many pages/links.

- Good form checking. Show errors on the right away

d) Try to add a new random photo

Tried to click on gallery in order to look for an add photo option. She was thinking the layout would be similar to that of Facebook. Then remembered that there were additional options after clicking on members. Played around with the drop down menu clicked on “Change Contents”>”Gallery”.

General Comments:

- Would be better if we can email members directly from the website

- Wasn’t sure what Rice Magazine was about. Directed to a new webpage, how to go back?

- Facebook and Twitter pop-ups would be better than changing the page

- The members/admin page was nice, but a clear button to go back to the main page would be nice.

Implementations: Changes We Will Make

* Move PHP functions to the very beginning, even before reading the htmls, so that the changes to the membership will show appropriate menu bar.
* Instead of a drop down menu, we will have the subcategories appear on the bottom of the main category when users hover over.
* We will allow html when adding projects/donations, to give more effects to descriptions/captions.
* Show the subcategories of the main menu bar constantly if they click one of the main menus.
* Show all the paths it took for pages with deep depth. Something like “Change Contents > Gallery > add Photo”
* Link emailing system to all the members in the “Manage Members” page, email to multiple members
* Make external links to new page.
* Better to have separate “register page”
* Have another page for FAQ. The questions will come from these User Testings.