Project Description:

Our website, JamOn, gives its users the ability to share music with friends. The user can create groups that his friends can join. In joining a group, the user essentially inputs his or her music preferences into that group. Then anyone within the group can play music from the collection of music that the group collaborates on.

Overall Design:

Our website utilizes the Soundcloud API, and the phpMyAdmin database. So long as a user has a Soundcloud account, the Soundcloud API provides our website with its user's music preferences, by accessing all of the songs that he or she has marked as a favorite in Soundcloud. The SQL database, phpMyAdmin, allows us to create tables, which serve as our users' created groups. Within these groups (SQL tables), the music preferences of each group member are stored in the form of permalink_urls which are essentially links to songs on the Soundcloud website. In our code, we are able to get these permalinks from phpMyAdmin, and then request for the songs that these permalink_urls represent from Soundcloud, in order to stream the song on our site.

The majority of our code is written in php, because php allowed us to access music from Soundcloud and create a website most effectively. We also use HTML and CSS to style our webpages, and JavaScript to do minor error checking on the browser side.

Pages:

Login/Register:

In register.php, we allow someone to register to use our website by taking the user's desired username and password from the html form, register_form.php, via POST. After affirming that the inputted request is valid, we create a new row in the SQL table, users, within phpMyAdmin that contains the user's username, hashed password and user id.

In login.php, we get a user's username and password from the html form, login_form.php, via POST and after checking if the user did in fact input a username and password into the form, we check if that username and password exist in the same row of the phpMyAdmin SQL table, users. If so, the user is logged in and is redirected to connect.php.

Connect

In connect.php, we access the Soundcloud API for the first time. By creating a new Soundcloud_Services variable, we send the Soundcloud website our client ID and client secret which we received upon registering our website with the Soundcloud API. Once Soundcloud recognizes that we have registered our website, it redirects our user to the Soundcloud Sign In page, asking if he or she would like to allow the website, JamOn, to access information that he or she has provided Soundcloud, including song favorites. If the user allows us to access this info, Soundcloud redirects the user to index.php – the redirect uri that we sent to Soundcloud with our request – with an access code that will allow us to get an access token which we can use to identify that the user has given our website, JamOn, permission to get his music preferences from Soundcloud.

Index

In index.php, after verifying that an access code was properly sent to our website from Soundcloud via GET, we get a non-expiring access token from Soundcloud, using the access code. We then store information in json about the user in an array. From this array, we get the user's Soundcloud username and user Id and store them in SESSION so that we can easily access them in future pages. We check if we were able to successfully access the access token and, if so, we call the function playlist(), which returns the user's Soundcloud song favorites in a json array. The website then renders home_template.php which welcomes the user to our site, prints out the users most current Soundcloud music preferences and allows the user to play each of these songs. The user is given various directions to proceed at this point. Lets say that he or she decides to create a group. The website is thus redirected to createGroup.php on the server side.

Create Group

In createGroup.php, if the user accesses the page via GET, the server renders the page createGroup_template.php, which is a very simple template containing one input text box that asks for the user to input the name of the group he or she would like to create, and a button that serves the purpose of submitting the information within the text box to createGroup.php via POST. Once the user submits the group name he or she would like to create, a JavaScript function, check(), runs to make sure that the name does not have any characters other than letters, numbers and spaces. It also checks to make sure that the user did not submit a blank form, or a form that only contains a space. We did this in JavaScript so that these three cases of invalid input would not be sent to the server, requiring the website to apologize to the user and reload the page. If the group name is valid based on those requirements, createGroup.php accesses this desired group name from the form via

POST and checks if it does not contain a SQL key word. If it does contain a SQL keyword, it is regarded as invalid and the user is prompted to input a different name. If it does not, the group name is changed via the function, altGroup(), so that it does not contain spaces allowing us to make SQL queries for this group. We create a new table in phpMyAdmin via the function query() and, after requesting from Soundcloud the user's music preferences via the function, playlist(), we input the group creator's music preferences into that table. Then we insert into the table, groups, the original name that the user inputted, so that we can keep track of existing groups to make sure that no one can create a group that has the same group name as another group. We also insert into the phpMyAdmin table, userGroups, the group name and the username of the user who created the group so that we can keep track of what groups the user is a part of. The user is then redirected to the home page. Lets say that the user now tries to join a group.

Join

In join.php, if the user accesses the page via GET, the server renders the page joinGroup_template.php, which is a very simple template containing one input text box that asks for the user to input the name of the group he or she would like to join, and a button that serves the purpose of submitting the information within the text box to createGroup.php via POST. Once the user submits the group name he or she would like to join, a JavaScript function, check(), runs to make sure that the name does not have any characters other than letters, numbers and spaces. It also checks to make sure that the user did not submit a blank form, or a form that only contains a space. We did this in JavaScript so that these three cases of invalid input would not be sent to the server requiring the website to apologize to the user and reload the page. If the group name is valid based on those requirements, joinGroup.php accesses the desired group name from the form, joinGroup_template.php, via POST and checks if the group name does not contain a SQL key word. If it does contain a SQL keyword, it is regarded as invalid and the user is prompted to input a different name. If it does not, the group name is changed via the function, altGroup(), so that it does not contain spaces, allowing us to make sql queries for this group. After all error checking is complete, we access the user's Soundcloud song preferences, calling the function, playlist(). If the user has Soundcloud favorites, we insert the permalink URL of each of the user's favorite songs and his or her username into a row in the phpMyAdmin SQL table designated for the group that the user wants to join. We also insert the user's username and the group name into a row in the table, userGroups, so that we can determine what groups the user is in later on. The user is then redirected to home. Lets say that the user now wants to remove a group member from the group, because that group member is absent from the group. He or she can click on the button, Update Members.

Update Members

Once the user presses the Update Members button, he is redirected to updateGroupMembers.php on the server side. The program will render updateGroupMembers_template.php, sending with it an array containing all of the groups that the user is a part of, and the usernames of all of the members within these groups. On the webpage, a table will be displayed. Each row of the table will contain the name of a group the user is a part of, the members within that group, and an input text box and button. The user is directed to type into the text box whatever user he or she would like to remove from the group in the text box's row and then click the Remove User button. This submits the form via POST to updateGroupMembers.php. Then, in updateGroupMembers.php, we check that the username that the user submitted is not blank or a space. Also, we check to make sure that the specified user is part of the specified group by determining if there is a row in the phpMyAdmin SQL table designated for that group with the user's username. If so, we delete all rows from that phpMyAdmin SQL table, containing the specified user's username, and also delete from the table, userGroups, the row that contains the group name that the user is being deleted from, as well as the username of that user, so that we are kept up to date on what groups the deleted user is a part of at all times. The user has one last page that he or she can visit. That is My Groups.

My Groups

In myGroups.php, all that we do is select the group name from all of the rows in the phpMyAdmin SQL table, userGroups, that contain the username of our user. We store this data in the array, $groups, and send this value to the html template, myGroups_template.php. This html template is what makes the web page that the user sees on his or her computer screen. On the webpage, there is a table similar to the one on the page, Update Members. It displays all of the group names of the groups that the user is a part of, and then, to the right of these group names, it contains a Play Songs button, which, when pressed, redirects the website on the server side to play.php. Similar to how we displayed the group names in a table in Update Members, in My Groups the html template, myGroups_template.php, uses a php foreach loop to iterate through the array $groups and print out each group name contained within it. With each iteration of the loop, the template also inserts a Play Songs button into the row, assigning the value of that button to be equal to the name of the group that is in the button's row.

When the user presses the Play Songs button, the website redirects on the server side to the page, play.php. In play.php we get the name of the group that the user wants to play music from, from one of the forms in myGroups_template.php via POST. We then create a variable containing the name of the group the user wants to play music from, altered so that it can be used in a SQL query. After confirming that there are songs in the specified group by checking in the phpMyAdmin table designated for that group, we select a random row in that group's phpMyAdmin table and take from that row the permalink_url of a song from within the group. We

store this permalink_url in $_SESSION['lastSongPlayed'], for later use. Using code from the Soundcloud API guide, we send a request to the Soundcloud API asking for the oembed information of the song represented by the permalink_url. We then embed the Soundcloud widget of this song in the html template, play_template.php. The webpage that this template creates allows the user to listen to the initial song, and/or press the button, Next Song, which redirects the website on the server side back to the page, play.php. This page will get another permalink_url from the SQL table designated for the specified group, and make sure that the selected song is different from the last one played, by comparing it with $_SESSION['lastSongPlayed']. It will then embed the Soundcloud widget for this song by sending its oembed information to the html template, play_template.php, just as it did before.

On the My Groups page, the user also has the option of pressing the button that asks, Changed Favorites Recently? This button, once pressed, redirects the website on the server side to update.php. Also, once pressed, the user is notified via the JavaScript alert function that his or her groups have taken note of any changes he or she has made in his or her Soundcloud song favorites. In update.php, we create two arrays, one containing the names of all of the groups that the user is a part of, which is found via a SQL query to the table, userGroups, and the other containing the user's most recent Soundcloud song favorites as permalink_urls. We then loop through each element of the array that contains the names of the groups that the user is a part of. Upon each iteration of this foreach loop, we first delete every row that contains the current user's username from the phpMyAdmin SQL table designated for the group that the foreach loop is on. We then reinsert into that group's table, all of the music preferences of the user, by using another foreach loop, which operates on the array containing all of the user's Soundcloud favorites. We then redirect back to myGroups.php.

Log Out

The only option left for the user is to log out. By clicking the Log Out button, the website redirects on the server side to logout.php. In logout.php, the function logout() is called, which empties the $_SESSION and $_COOKIE arrays of the user. Then the session is destroyed using the function, session destroy(). The user is redirected to index.php, where he or she is again prompted to either log in or register.

That was a tour beneath the hood of JamOn!