COSC 360 Web Programming

Course Project

Alpacapella

Implementation Report

Carson Prokopuik 41859166

Tahmeed Hossain 62409164

2020/04/09

The University of British Columbia Okanagan

Introduction

Alpacapella was the result of multiple sessions of brainstorming between project partners Carson Prokopuik and Tahmeed Hossain. The site is intended to allow users to express their opinions on the latest music industry news, as well as get caught up with the latest info regarding the music industry. We envisioned our website to act as a hub for all things music.

Implementation and Process

Implementation of the website started with page structure development with HTML. We developed static HTML documents for each page of the site with hardcoded content to build the foundation on which our project would grow. As the COSC 360 course progressed, our web development skills increased allowing us to style our HTML with CSS, create client-side processing with JavaScript, and then backend processing via PHP and MySql.

In order to have the website start having actual data be presented, we needed to implement a database and server. For this we chose the XAMPP/MAMP Localhost MyPHPAdmin. This proved to be somewhat of a hinderance as both partners were using a different localhost tool and thus ran into permission issues when connecting to the database. We were able to somewhat overcome this issue by having different database credentials for each developer, however this meant that we would have to ignore the db_credentials from github diffs. Our database is set up to have four tables; customers, admin, blogpost, and customers. Initially we thought we could perhaps store blogposts in the customers table, however further thought led to use realizing that the best strategy was to have separate tables for each and link them by user_name. The same goes for comments and blogposts respectively.

The majority of the code written for this project was in PHP. PHP allowed us to have the website display what we wanted, and how we wanted it. For example, our blog search bar takes text input from an HTML form which then runs a PHP script on submit to query a '%LIKE%' search through the blogposts relation. By doing this, we can easily iterate and display each post via a for loop making our code more compact and robust. Additionally, all user authentication was done via PHP. For example, when a user enters the site at first, they are not logged in, the site knows this due to the $_SESSION['authenticatedUser'] variable being equal to null. To sign in, the user must navigate to the sign in page via a link in the header (or through the media column) and submit the login form. The website then executes a query to check if the user name and password entered by the user matches that of an existing user, if so the $_SESSION['authenticatedUser'] variable is set to the user's user_name rather than null, and will be used throughout their session until they logout by clicking logout and setting $_SESSION['authenticatedUser'] back to null. Customer creation was also done via PHP and MySql commands. Once the user submits the signup form, a php script executes an sql 'INSERT' statement to the customers relation with all the info from the signup form. Additionally a new directory is made in the '/MEDIA/User/' directory, as this directory will hold the users profile picture as well as directories for each blog post they make and the photos that belong to that blog post.

For admin access and features, we added an admin relation to our database to support our admin accounts. From the login page, you may select if you would like to log in as an admin or not. If you select log in as admin, the validatelogin.php file queries the admin relation instead of the customers relation when checking if the inputted user_name and password exist and match. Once an admin logs in, they will be brought back to the home screen which will for the most part look the same. The main difference being that the header now displays a link to 'Site stuff', which is where all the admin actions may be performed. At the Site stuff page, an admin may be able to display customers, admins, blogposts, and comments. They may also select a tuple in the displayed table and remove the selected tuple. The remove is executed by a DELETE FROM statement using PHP and MySql. The described functionality applies to the site deliverable of allowing admins to remove or delete users, comments, blogs, etc.

Users may upload blogs by starting a blog in the right media column. Once the user clicks get started, they are sent to a more descriptive form which is pre-populated with info from the previous form regarding blog post details. Users may upload an image for their blog if they would like however for now only .jpg photos are supported. Users may enter text which acts as the blog content. Once a blog is created, a new directory is created in '/MEDIA/User/'user_name'/'blog_title'', this directory will hold photos that are used in the blog.

Lastly our site allows users to update their profile. The extent of the update includes editing any information that they currently have about them selves except for their date_signed_up attribute which is data used by the admins. This update is implemented in PHP by executing a MySql Update query. The query is based on the users 'old_username' which is the user's username before the submission of the updates, since the database will still hold the old username until the update command finishes.

Summary

In summary, this project proved to very rigorous and demanding. Both partners found them selves in multiple positions where "all-nighter" development sessions were needed to get the website to where it is now. However for what the site is, and given the recent worldly circumstances, we are happy with the core functionality of Alpacapella, as well as its concept