Spring 2016

Artistic Connections, a Community Development Tool

Amanda Cobb
Governors State University

Patrick Douglas
Governors State University

ABSTRACT

This purpose of this project is to develop a peer to peer connection service, designed to give artists the ability to easily find, connect and collaborate with other artists. Current attempts at meeting this need revolve around forum based architecture or social media outlets. These existing structures do not provide a clean or focused means of connecting artists. We intend to provide artists with a smooth and transparent management interface designed solely around the needs of its artistic community. With our software, artists will no longer have to scour media sources or generate thread posts for collaborative needs. As this concept grows, this software will be the foundation of a new artistic community.

Table of Content

1 Project Description

This website is to develop a peer to peer connection service, designed to give artists the ability to easily find, connect and collaborate with other artists. Current attempts at meeting this need revolve around forum based architecture, such as artist forum or social media outlets, such as Facebook. The existing structures do not provide a clean or focused means of connecting artists. We intend to provide artists with a clear management interface designed solely around the needs of its Artistic Community. With our website, artists will no longer have to scour media sources or generate posts for collaborative needs. As the concept grows, this website will be the foundation of a new Artistic Community.

1.1 Competitive Information

Our competitors will be the social media and forum based sites. Our website has the potential to be the first on the market to provide a clear managed interface designing for the Artistic Community in mind.

1.2 Relationship to Other Applications/Projects

This project will gain user traction through the use of existing social media outlets. It may eventually utilize login functioning from other sources.

1.3 Assumptions and Dependencies

- This project is not dependent on outside progress.
- This project is dependent on the current structure of the internet, including standard browsers such Google Chrome, Firefox and Internet Explorer. As well, existing social media structures are essential to the growth of the user base.
- This project assumes the health and maintenance of database management system, MySQL.
- A monetization program must be developed for future development and maintenance.

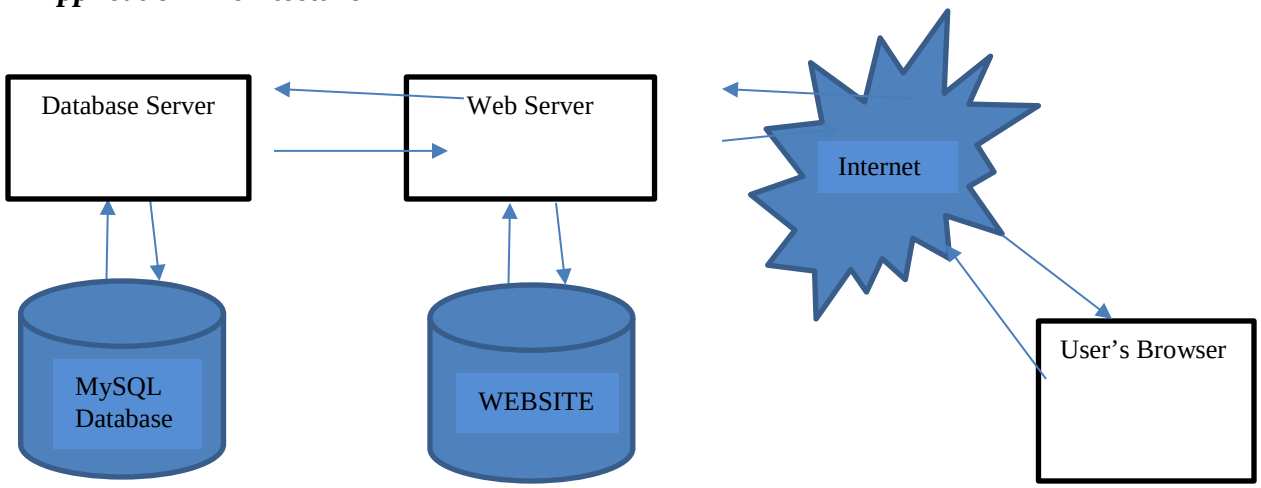
1.4 Future Enhancements

- A system of communication for the artists
- Inner-site economy based on ratings. Users receive up votes or down votes based on their projects. Should incentivize thoroughly assisting on projects.
- An interactive workspace, such as a built in text editor, like Google's, "Google Docs."

2 *Project Technical Description*

This project is an ASP.NET focused community development tool. It's structured around a standard member-based website. It will be written using Microsoft Visual Studios 2015. It should use HTML, JavaScript, ASP.NET and CSS to structure the user interface/frontend. The backend and database connection will be written in C#. Database queries will be written in standard SQL format. The database itself will be MySQL. Population of the database will be done through user registration and profile updating on the website, as well as project use.

2.1 Application Architecture



2.2 Application Information flows

The user will land on the webpage at an introduction page (See Home.aspx, and Home.aspx.cs in the attached project folder). This page is designed to introduce the user to the concept of the website while providing them with the tools to set up their own account. The home page has two major functions, the login mechanism and the registration jump. While returning users can utilize the log in function, new users will be addressed to the registration page (See registration.aspx/registration.aspx.cs). Following a login or registration, the user will then land at their profile (See artistprofile.aspx/artistprofile.aspx.cs), which is populated via the database. There are multiple directions a user can go from here, including a profile section to update their profile information, an artwork section to upload previous works, a search section to find other artists, and a project section for managing each project you have set up. The user can go from the search section to other users profiles (See requested_user.aspx/requested_user.aspx.cs). Here, they can set up projects (See projects.aspx/projects.aspx.cs) or return to their profile.

2.3 Interactions with other Applications

- Database
  - Utilizing MySQL, the project will need to maintain a connection string to manage SQL queries and data requests, updates, inserts or deletes.
- Web Browsers
  - The website will need to maintain a functioning status on multiple web browsers including Firefox, Google Chrome, Internet Explorer and Microsoft Edge.

2.4 Capabilities

- Provide a login/logout function to maintain user states
- The database must maintain user information including demographic and functional characteristics.
    - E.g. the database is responsible for identifying the different artists
    - Information must be retrievable, updatable, able to be deleted, and able to be inserted from the website.
- Must be able to create project spaces for user combinations, including file creating and database space.

2.5 Risk Assessment and Management

Due to its web based nature, this project is susceptible to all standard web based attacks. One such attack is an SQL injection attack. (See registration.aspx.cs for how this project deals with this risk). Healthy, time tested preventative measures should be taken to protect from data theft.

3 Project Requirements

3.1 Identification of Requirements

<GSU-GS_SP2016 User-Capability-000101>

The user must be able to register a new account by the application, as well, the user should be able to use that registration information to log into the website. A simple registration and log in page should be easy to implement on the websites first iteration. The end product should have be secure and use validators to maintain integrity.

Implementation: Mandatory

<GSU-GS_SP2016 User-Capability-000102>

The user must be able to manage their profile, including adding/removing skills, uploading a user profile picture and uploading previous artwork for display. Each of the previously specified should be implemented with simple ASP.NET connections at first, such as the upload implementation. As well, the adding/removing skills section should utilize the SQL data connection to update/insert/delete

Implementation: Mandatory

<GSU-GS_SP2016 User-Capability-000103>

The user must be able to find and view other users profiles based on their talents, as update in their profile. This process should begin with users being able to select a talent they are seeking. The database should take this information and supply a gridview that displays all the users with that specific talent.

Implementation: Mandatory

<GSU-GS_SP2016 User-Capability-000104>

The user must be able to create a project space with other users that allows for artwork uploads. They should also be able to access this project space from their user profile. From the gridview in Capability 000103, the user should be able to navigate via hyperlink or search function to other users profiles.
Implementation: Mandatory

<GSU-GS_SP2016 Data-Capability-000101>

The database must allow new users to be added by the application's registration page. The connection string set up should allow for this type of privilege.
Implementation: Mandatory

<GSU-GS_SP2016 Data-Capability-000102>

The database should use triggers and normalization to maintain integrity. This should be implemented across many situations. On new member inserts, the database must update each table with new user information. Primary keys and Foreign Keys should be set up with cascading options so that information can be updated/deleted across multiple tables.
Implementation: Mandatory

3.2 Operations, Administration, Maintenance and Provisioning (OAM&P)

The project backend will provide data protection including redundancy. Routine database and website maintenance will provide a constant stream of updates based on user feedback.

3.3 Security and Fraud Prevention

There are plenty of possible security issues including user identity theft and database theft. Users should have the capability to upload project materials, including sensitive materials that may be the focus of attacks. These issues need to be addressed.

3.4 Release and Transition Plan

The website will have a single-point launch

4 Project Design Description

Database Tables

The four tables are members as listed in figure 1 are artist, projects and uploads.

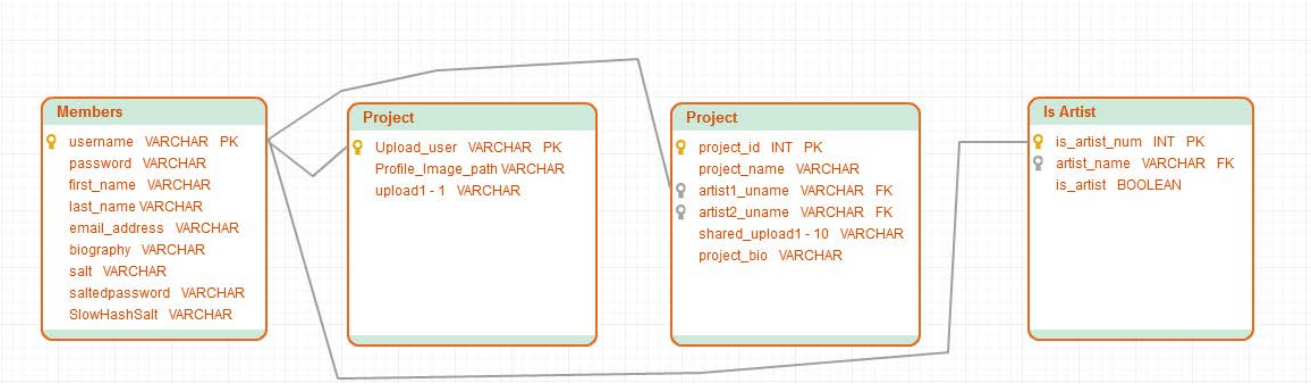


Figure 1

Members

The members page consist of 9 columns; username, password, first_name, last_name, email_address, biography, salt, saltedpassword, and SlowHashSalt. This page demonstrates the beginning of a portion of this application that has yet to be finished. The project requires that the user's password be stored in a safe manner. This means the password should be ran through a hashing algorithm before being stored. The database has the required fields, the application front end simply needs to implement the usage of the hash. This table actually has a trigger attached that submits directly after any insert operations are done. The trigger populates the Is Aritst tables initially.

Is Aritst

Is Artist is actually a placement name for the multiple tables there actually are. There are as many tables are there talent types. For example, the talent programming has its own table and is name is_programmer. This is the case for all of the different talents. It is easier to simply specify one table type rather than all of its iterations. These tables function exactly the same. There are three columns. The primary key is is_artist_num is automatically created. Artist_name is a unique object that is a foreign key to the member's table username column. This relationship makes sure there are no artist that are not already users. On creation of a new user, there is a trigger that populates each of the artist tables. The last column is a Boolean value or tinyint in MySql language. A 1 represents that user has this talent and 0 represents that they do not have this talent.

Project

This table manages the different information needed for the user's project space. It has the columns of project ID, project Name, artist1_uname, artist2_uname, shared_upload1-10 and project_bio.

Uploads

This table manages each user's uploads. They are limited to 10 uploads which are represented by the 10 upload columns. The upload_user column is a primary key as well as a foreign key to the members.username column.

Landing Page

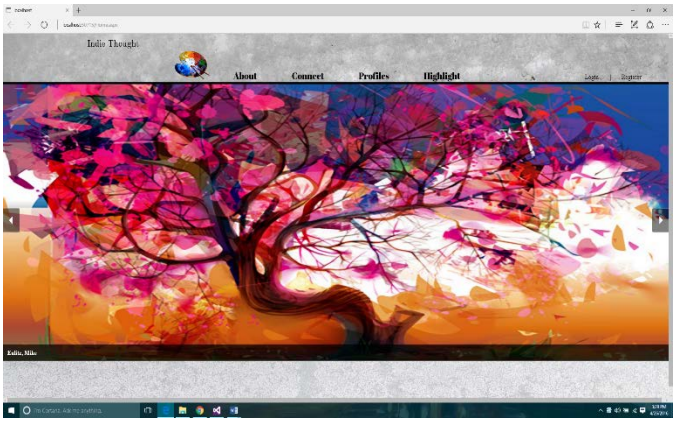


Figure 2

This is the first interaction the user will have with the web page. Its elements are designed to provide the user with a pleasant description of the services provided by the website. It will act as a host and an advertisement. The page will highlight moderator-selected artwork submitted by current users. As can be seen as the main focus of the page. This image will be rotating to demonstrate different types of art on display. Not only will the page display the artwork, it has a title and artist section to highlight the source. The landing page should have the project's logo as well as the project's name. These can be found in the upper left hand corner. Directly to the right of the information tab is the required login and registration tabs which will be further discussed later.

Landing Page, Information

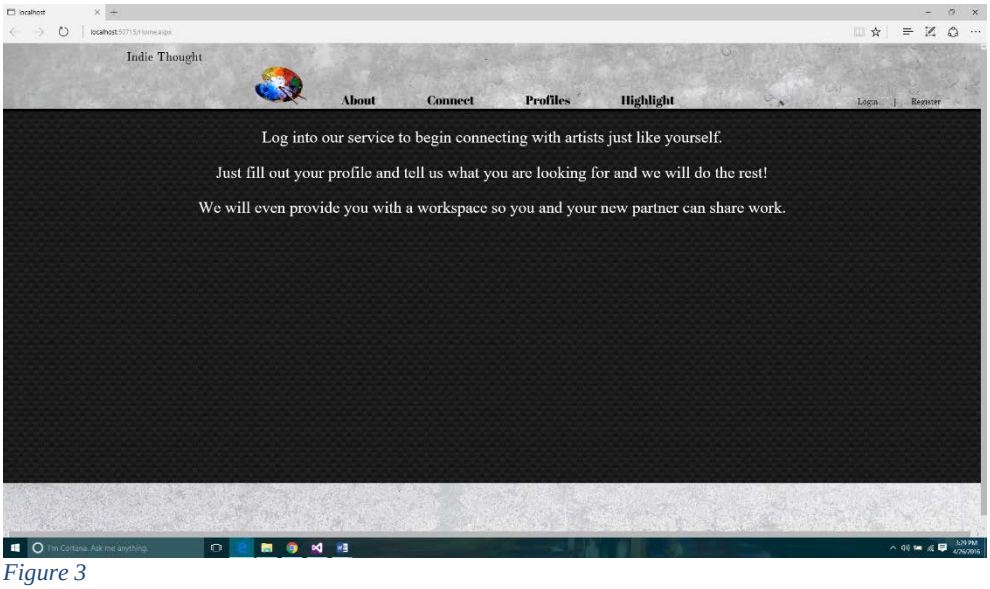


Figure 3

As seen can be seen in figure 3, the landing page has objects for the user to interact with that display information about the project. The “About” tab displays general information regarding the project. “Connect” details how the website will bring new users together and the “Profiles” tab explains how each user will manage the website. The “Highlights” tab explains what is going on with the landing page and how to submit artwork to be displayed.

Registration Page

The registration page is accessed via the registration button found on the home page and can be seen in figure 5. This is a one-time-visit page that queries the user for first time profile information. This

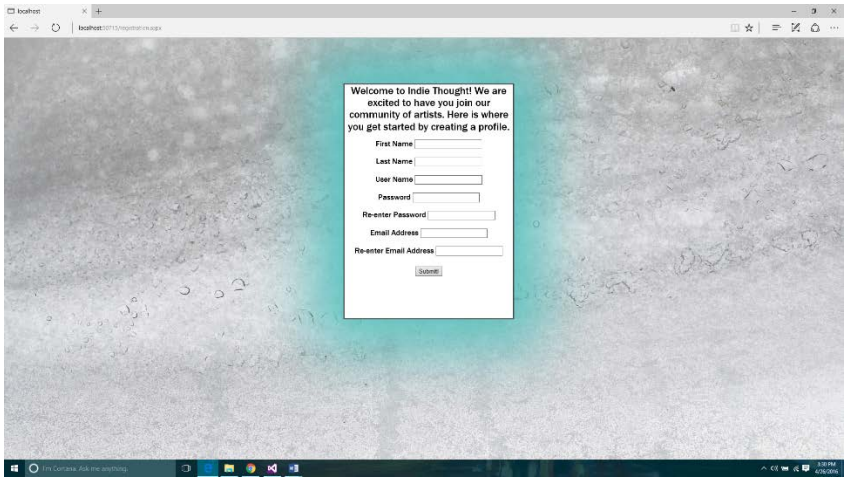


Figure 4

is a standard registration page that requires the user to enter the following items; first name, last name, desired username, desired password and a valid email address.

Registration Page, Validations

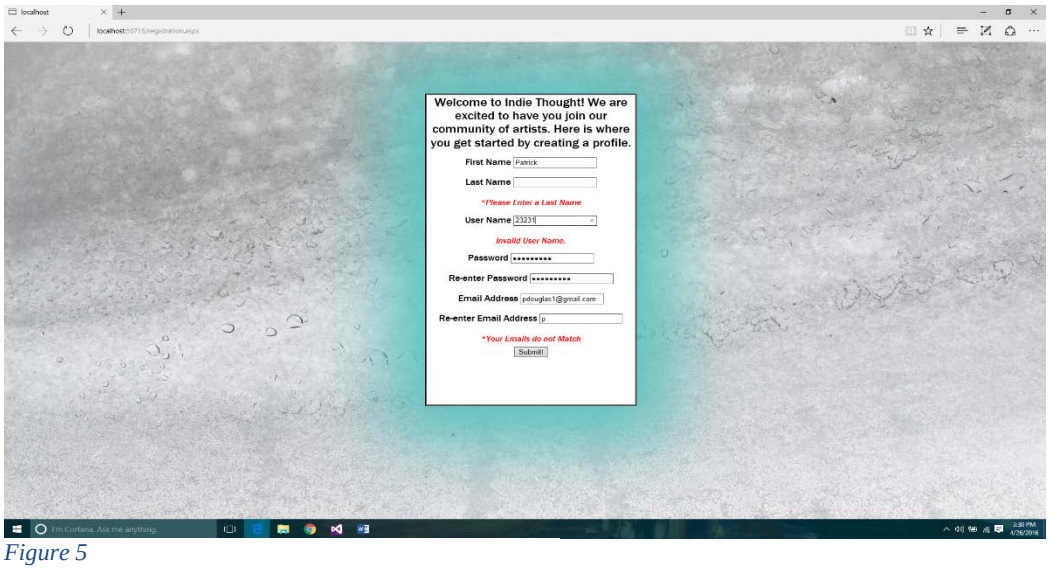


Figure 5

The registration page enters sensitive information into the database and as such many of the fields have a form of validation, the results of which can be seen figure 6. Each field has a required field validator, meaning if the user attempts to create a

profile without filling one of these pieces of information out, it will be prevented. The username uses a validator to restrict special characters and limit what users may use. The usernames are used in many functions across the website and it is much easier to have them basic alphanumeric keys. Passwords sections have two validators. It is imperative that the password is typed correctly by the user so the project supplies a second user input field to double check this information. The validator checks to see if the string in the first password box matches the string in the second password box before allowing the registration.

The first password box also has a special character validator that makes sure the user is using a password of the correct length and complexity. This validator is called a regular expression validator and it validates the existence of certain characters in the string. After clicking the submit button, two things happen. For one, an alert message, initially hidden by JavaScript is put on display and the C# function related to the OnClick method of the submit button is processed. This function is instructed that they will be returned to the landing page so they may now use the login function.

Landing Page, Login

The login section will be hidden with a JavaScript function, initially. When the user clicks the login link, a new JavaScript function will display not only the login DIV, but a dimmer to highlight the DIV and an exit button as can be seen in figure 7. The dimmer is simply a div stretched to the length of the DIV and uses a lower opacity to give the page a darkened look. The exit button is simply a div with JavaScript attached that hides the previous objects on click.

The login section has an important piece of security function in the home.aspx.cs page. Login pages that request user information from a database often susceptible to SQL injections. Code can be found in this document to prevent these types of attacks.

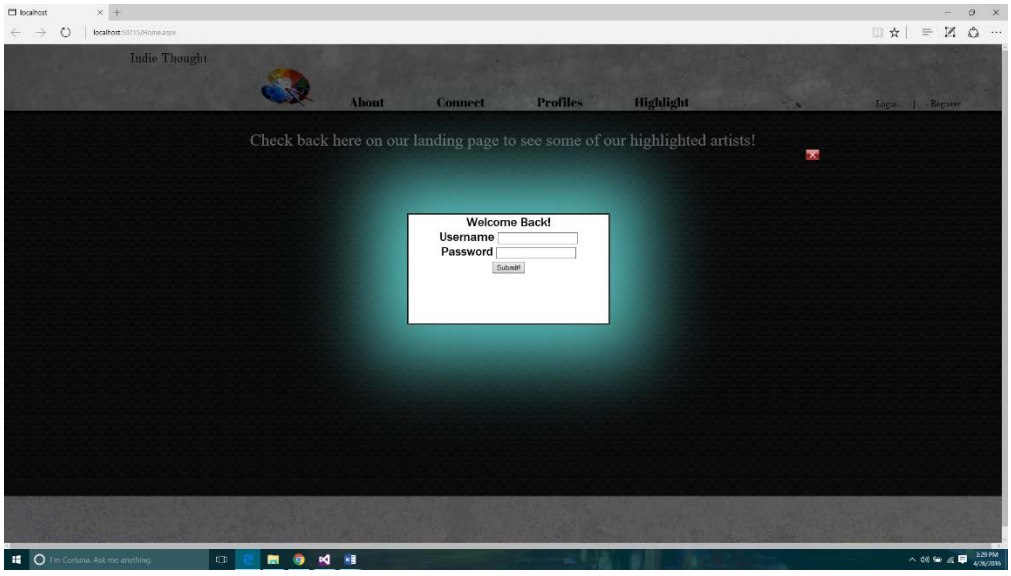


Figure 6

The login div displays a welcoming message and two sections for user input including the username and the password as well as a button that will submit the information for query against the

database. If the username and password match up to information in the database, the landing page sets a session variable for the username information and moves the user forward to the profile page.

Profile Page

Figure 8 is the user's profile page. The on-page load function of the C# file takes the session variable delivered from the landing page login button and uses it to query the database. It selects all of the user information for display. The query inner joins multiple tables to identify the user's talents, as each talent is on a different table. Found on the page is a logout button that simply redirects to the home page and more importantly, clears the session variable from the previous login function. This means, if the user logs out, their information cannot be obtained by the next person that sits down.

Displayed on the landing is the users profile image, their name, their biography and several menu options. The first script that runs on the page is a JavaScript that actually hides all of the different sections of the user profile for a cleaner experience. The user can direct themselves to view more detailed information by manipulating the menu with the following options:

- Profile
  - To find more detailed information about the user
- Artwork
  - To submit previous artwork
- Projects
  - To manage and access projects
- Find Artist
  - To find new artists

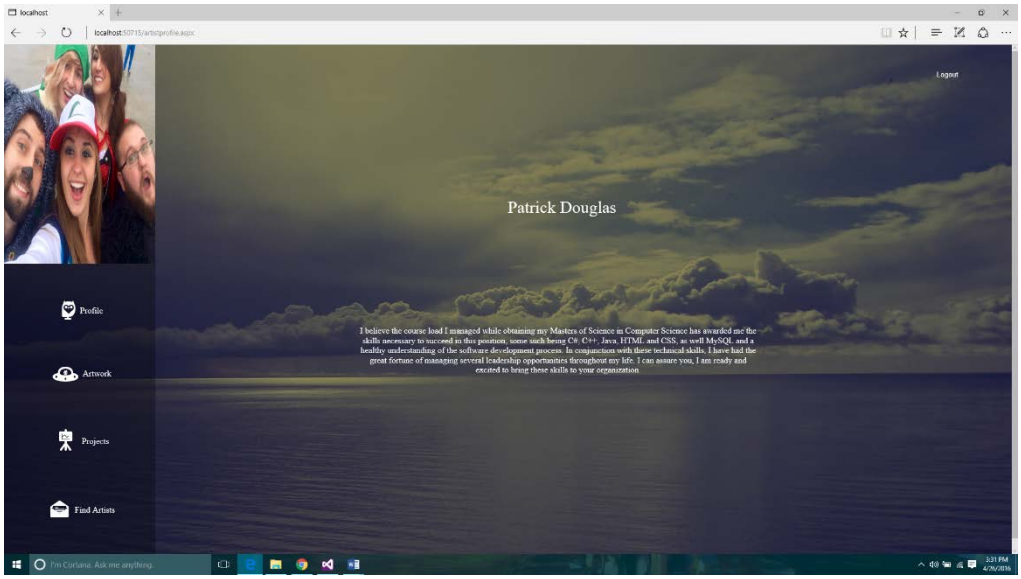


Figure 7

Profile Page, Profile Information

To launch this page (which can be seen in figure 9), the user clicks on the “profile” button/icon which triggers a JavaScript to clear the initial background to make way for the new container. This new container states the user’s name, their username and their email as well as the list of their talents in picture form. These talents can be removed by simply clicking on them which runs an OnClick function that submits an update statement to the database.

On the initial load of the profile page, the database is queried for “is_artist” values, meaning user is artist, for example: user is bassist. These are saved in the database as Boolean values. A value of 0 (the

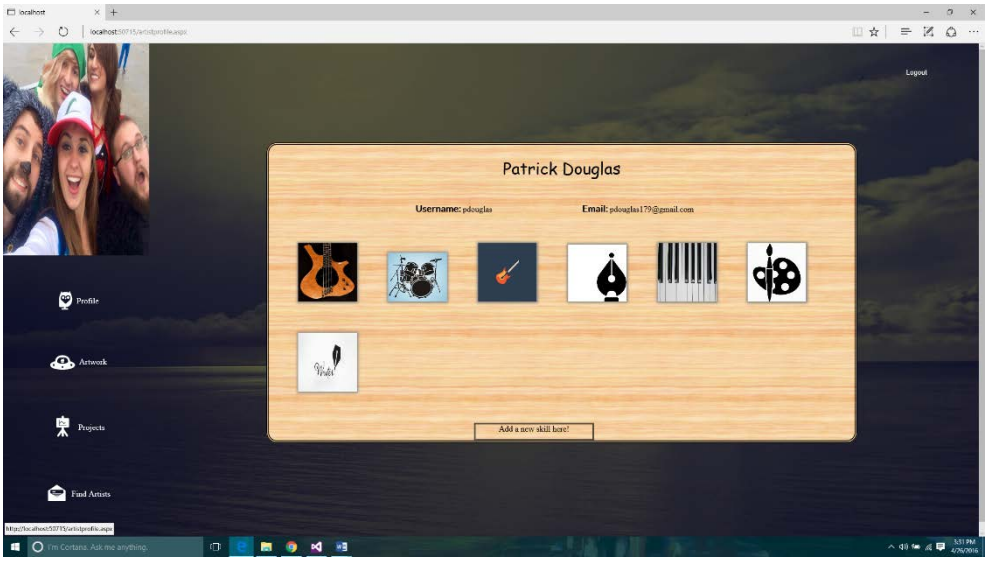


Figure 8

initial value) means the user is NOT this type of artist. These values are then used to populate this list of talents. If the user returns a value of “1” for is_actor, the actor icon will show up, symbolizing that this user has the actor talent. Instead of deleting the value in the database, when the user selects a talent to be removed, the application simply runs an update statement changing the value from a “1” to a “0” and refreshes the page, effectively removing the talent.

At the bottom of the container, is a button that when used, brings up the ability to add a new skill. This container is also filled with images that show the talent to be added. The user simply needs to click the icon to add the talent. The click submits an update statement that changes the user’s “is artist” information from “0” to “1” this time.

Profile Page, Upload/Display Artwork

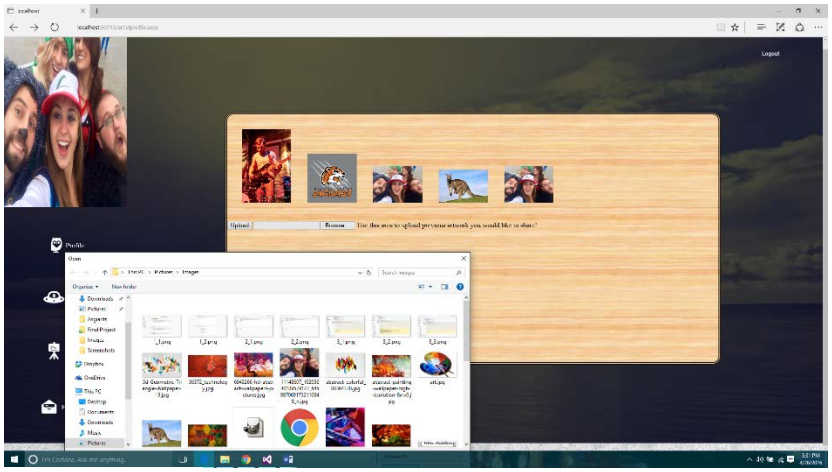


Figure 9

This subsection is obtained by the user clicking on the Artwork menu icon. On the initial load of the profile page, a function is ran that creates a file for that user (if it doesn't already exist). This container display artwork that can be uploaded to this file. The user is given the ability to browse his local machines contents for artwork to upload. The way this works is by having multiple blank images not displayed initially. In the user uploads table is several columns available for filenames. The page uses these filenames along with the file path already generated on page load to display the correct images. In other words, the C# code changes the source of the image from nothing to the value given by the database.

Profile Page, Project Manager

This container is accessed the same way as the previous two. The very first thing in the container is a label and a drop down menu. This drop down menu is populated using a MySQL data source object in the C# file. The data source queries the database for every project where the user is either first or second artist and then returns the project names in a data list. Every time the user selects a new drop down list item, a function is run, based on the drop down lists onselectedindexchange method that pulls the text value of the new item and uses it to query the database. This returns the user names, the project name and the project bio. This allows the user to quickly browse all of his projects.

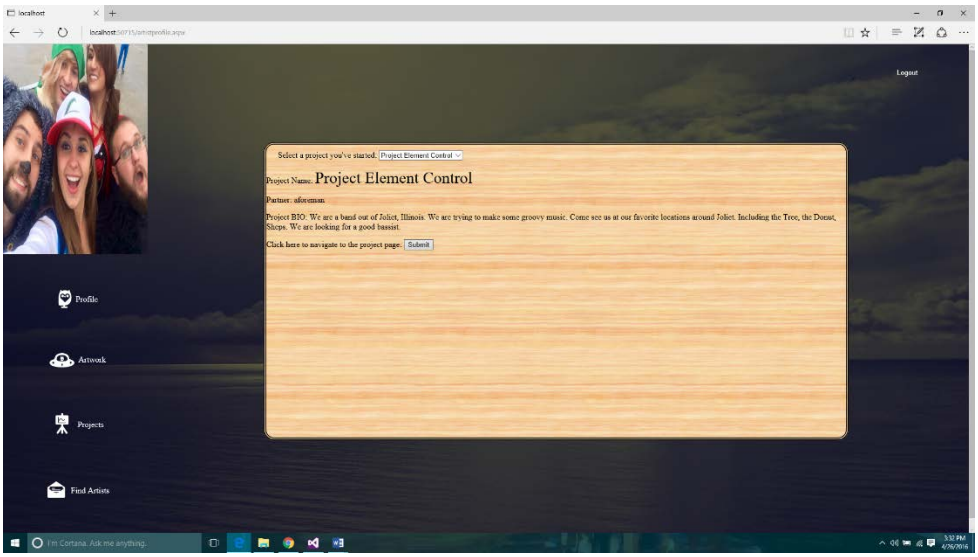


Figure 10

Just below the project bio is a submit button. This will actually redirect the user to the project page for the selected menu item. This submit function creates multiple session variables from its previous SQL queries. It creates an artist1 variable, an artist2 variable and a project name variable. This allows the project page to use this information for its own query later on.

Profile Page, Search Section

This container is where artist can go to actually find other artist. The drop down list supplied is pre-generated with a list of all of the available talents. On page load, this container is filled with a grid view of information for each different type of talent. A JavaScript hides these grid views. When the user

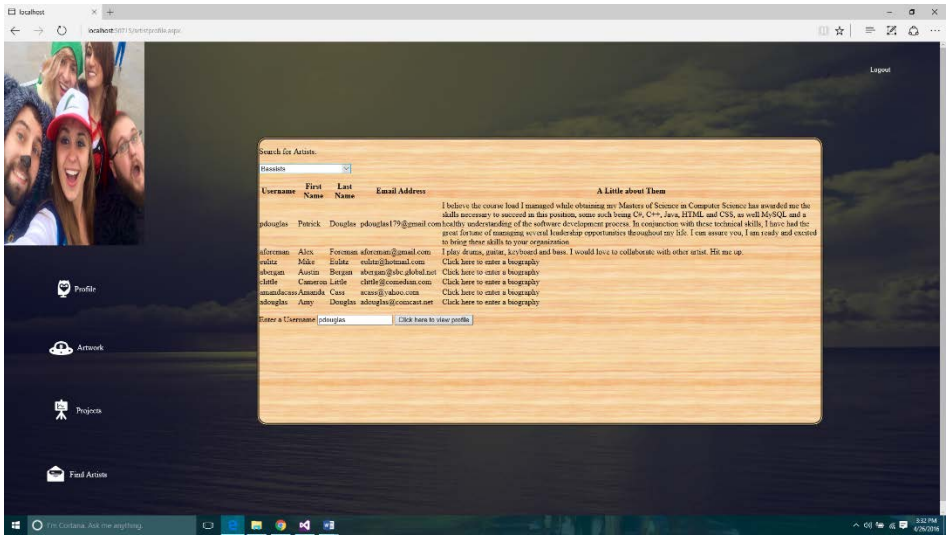


Figure 11

selects a new drop down list item, a JavaScript function grabs the value of the new item and uses it to show only the appropriate search information. For example, if the user selects “Actors” the JavaScript will be sure any grid view that isn’t the Actors grid will be hidden and that the Actors grid will be shown. These grids are populated with a sqldatasource object through the ASP.NET toolbox. The grids list every artist that has that particular talent as one of their own.

The remaining portion of the container is a user input box and button. The user is instructed to enter another user’s username. Following this, if the user presses the submit button, a function is ran that redirects the user to that requested user’s profile for viewing. The submit function sets up session variables to populate the requested users information.

Requested User Page

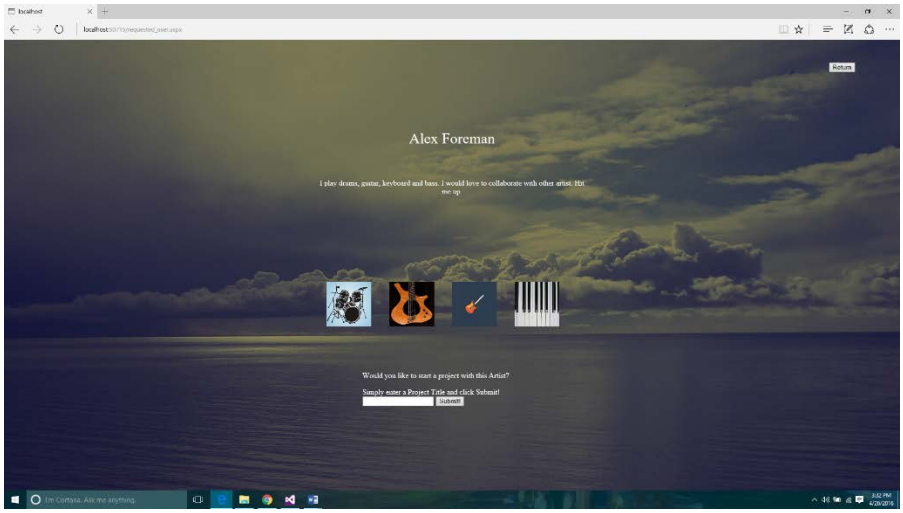


Figure 12

This page displays the requested user's name as well as his biography, which can be used to allure other artist. Also, the page displays the requested user's talents, also in picture form. Just below this is the portion of which a user can set up a project. There is an input field and a submit button. The Input field is designated for the project title and the submit button runs a function that updates the database's project table with this new information, including the artist1, artist2, and project name. The function also sets up several session variables so that on load of the redirected project page, information can be displayed.

Project Page

This page is accessible by either user involved in its creation. It allows the users to upload different materials, much the same way that they can to their own user profile page. They can also create and adjust their biography. This page comes equip with a return button that will redirect the user page to their profile page by adjusting the session variables.

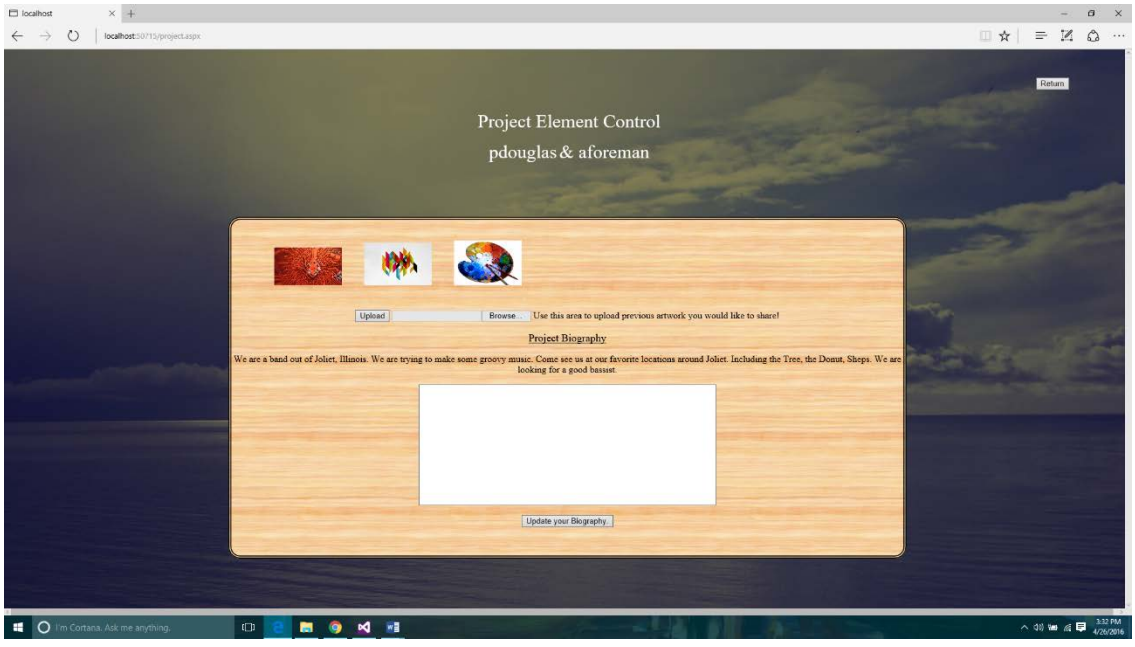


Figure 13

5 Internal/external Interface Impacts and Specification

This is the first iteration of both the interface and data objects. Drastic changes to their specification will happen throughout this project.

6 Design Units Impacts

There will be only one Design Unit. It will be Amanda Cobb and Patrick Douglas working as Design Unit A.

6.1 Functional Area A/Design Unit A

6.1.1 Functional Overview

This unit will be generated for the first time and responsible for the entire management and creation of the application.

6.1.2 Impacts

There are multiple areas of impact for the community. We hope project creates a stronger and more helpful artistic community through cooperation on projects. It should impact not only current artist but new artists looking for work or help.

7 Acknowledgements

We would like to thank the professors at Governors State University for giving us the knowledge needed to finish this project!

8 References

Delmater, Mary, and Anne Boehm. Murach's ASP.NET 4.5 Web Programming with C# 2012. 5th ed. Fresno: Mike Murach & Associates, 2013. Print.

Elmasri, Ramez, and Shamkant B. Navathe. Fundamentals of Database Systems. 7th ed. Pearson, 2016. Print.

Felke-Morris, Terry A. Web Development & Design Foundations with HTML 5. 6th ed. Pearson, 2013. Print.

Kroenke, David M., and David J. Auer. Database Processing: Fundamentals, Design, and Implementation. 13th ed. Pearson, 2014. Print.

McConnell, Steve. Code Complete. 2nd ed. Microsoft, 2004. Print.