Spring 2016

Knoxville Museum of Art Audio Tour

Sharvari Sanjiv Desai
sdesai7@vols.utk.edu

Knoxville Museum of Art Audio Tour

Sharvari Desai

April 26, 2016

1 Executive Summary

The Knoxville Museum of Art is an art museum located in the heart of downtown Knoxville. The museum currently has descriptions of its many art pieces written beside the displays in paragraphs of text. However, this current setup only allows the museum to include a certain amount of text description about the pieces, and not all museum visitors want to take the time to read the small print to the side of each piece. As a result, the museum has requested a website application that allows the museum to upload snippets of audio for selected pieces in order to provide an audio tour of the museum galleries. When a museum visitor navigates to this website application on the Knoxville Museum of Art website, they should be able to use a smartphone to play these audio snippets as he or she navigates the museum, making the museum experience more fulfilling and enriching.

At the end of our project, we have a working product that has been tested to ensure that it meets all of the customer's requirements. Visitors to the museum will be able to scan barcodes next to each piece in the museum and have the application navigate them to the same piece within our application. They can then play the audio file associated with that piece and continue through the audio files in that exhibit to create an audio tour of the museum. The Knoxville Museum of Art staff is able to add, edit, and delete all art pieces, artists, and exhibits through a content management system, allowing them to easily maintain the app for years to come.

2 Requirements

1. Mobile Web Application
    - 1.1. Users must be able to view a gallery view of all art pieces.
    - 1.2. Users must be able to sort by exhibits.
    - 1.3. Users must be able to scan a QR code, which will forward them to the address of the piece.
    - 1.4. Users must be able to view an individual piece.
        - 1.4.1. Users must be able to view a photo of the piece (where allowed by copyright).
        - 1.4.2. If painting is copyrighted, display a default "image not available" image.

1.4.3. Users must be able to view the piece's artist.

1.4.4. Users must be able to view the piece's title.

1.4.5. Users must be able to view the piece's description.

1.4.6. Users must be able to listen to the .mp3 audio file for the piece.

1.5. Users must be provided search mechanisms

1.5.1. Users must be able to search pieces by title.

1.5.2. Users must be able to search pieces by art tag.

1.5.3. Users must be able to search pieces by artist.

2. Content Management System

2.1. The customer must be able to add a new piece.

2.1.1. The customer must be able to add a piece's title.

2.1.2. The customer must be able to add a piece's artist.

2.1.3. The customer must be able to add a photo of the art piece.

2.1.4. The customer must be able to add a brief description of the art piece.

2.1.5. The customer must be able to add tags for a search mechanism to find the art piece.

2.1.6. The customer must be able to add .mp3 files related to the art piece.

2.1.7. The customer must be able to add categories and/or museum exhibits.

2.1.8. Creating a new piece must generate a QR code link to the piece's information.

2.2. The customer must be able to edit an existing piece.

2.2.1. The customer must be able to edit a piece's title.

2.2.2. The customer must be able to edit a piece's artist.

2.2.3. The customer must be able to edit a photo of the art piece.

2.2.4. The customer must be able to edit a brief description of the art piece.

2.2.5. The customer must be able to edit tags for a search mechanism to find the art piece.

2.2.6. The customer must be able to upload replacement .mp3 files related to the art piece.

2.2.7. The customer must be able to edit categories.

2.2.8. The customer must be able to edit museum exhibits.

2.3. The customer must be able to delete a piece.

2.4. The customer must be able to browse all art pieces.

2.4.1. The customer must be able to view the information associated with each piece.

2.5. The customer must have administrative abilities.

2.5.1. The customer must be able to login to the content management system (CMS).

2.5.1.1. The customer must be able to enter a username in order to login to the CMS.

2.5.1.2. The customer must be able to enter a password in order to login to the CMS.

2.5.2. The customer must be able to create a new user for CMS privileges.

2.5.3. The customer must be able to delete a user for CMS privileges.

2.5.4. The customer must be able to change a password for an existing user for CMS privileges.

3. Additional Design Requirements

3.1. The application must be usable.

3.1.1. Functionality described in this requirement document must be properly implemented.

3.1.2. Through user experience testing, ensure that users of all Knoxville Museum of Art demographics are able to use the web application.

3.1.3. User interface must rate at or above 4 out of 5 start in testing groups for usability.

3.1.4. User interface must rate at or above 4 out of 5 stars in testing groups for aesthetics.

3.1.5. Interface must meet Section 508 standards.

3.1.6. Interface must be accessible on screen readers.

3.1.7. App must display at appropriate scale at various screen sizes.

3.1.8. App must meet usability requirements on Safari, Google Chrome, and Opera.

3 Change Log

April 4, 2016

Instead of the application generating QR codes and having the application's users scan a QR code beside each painting, the application will now generate barcodes and have the application's users scan a barcode beside each painting in order to navigate to the correct art piece within the application.

4 Documentation of the Design Process

4.1 Analysis of the Requirements

When beginning our project, our client initially wanted to create a native mobile application that would provide all the desired features. However, following much discussion, our group and the client came to the agreement that a web application would better meet their needs. Very few visitors to the Knoxville Museum of Art would be willing to download an entirely new mobile app simply for an audio tour of the museum, particularly if they do not visit the museum very often. Also, developing a native mobile application would require us to develop one application for iOS and another application for Android. Even if we did use technologies such as the Xamarin or Ionic frameworks to develop the applications for both platforms in parallel, we would need to do some testing and user interface development on each platform separately. Due to these numerous reasons, the customer and our project team came to the conclusion that a web application would be the ideal route for the museum.

The Knoxville Museum of Art website is currently hosted by a separate web development company called BigWheel. Because our customer would like us to host our web application as a portion of the already-existing website, it was important for us to reach out to BigWheel to figure out what kind of database system we would have available to us. We were not able to

begin our project until they had responded that we would have access to a MySQL database.

Considering the fact that the Knoxville Museum of Art staff has no direct access to the website's database, we needed to develop our entire web application on a separate database before transferring it over to BigWheel for hosting when our application was complete. Our first idea was to request an account on the University of Tennessee Office of Information Technology's public MySQL database server. Unfortunately, we were not able to use this solution because the Office of Information Technology requires that users be connected to the University of Tennessee's internet connection in order to access the database server. This solution would simply not allow for development of our application from any locations away from campus and therefore was not feasible. Following further research, we found that through the University of Tennessee's partnership with the DreamSpark program, Microsoft provides Azure services to students at no charge, providing us a way to set up our own MySQL database.

On the front-end, it was important for us to begin considering the user interface and user experience of our application right away. In order to properly visualize what we wanted our final web application to look like, it was vital that we find a tool that would allow us to efficiently create application prototypes. After researching our options, we found a beautiful prototyping tool called Pixate that met all of our needs. Luckily, Pixate also offers free student plans to allow us to create accounts and begin prototyping our application right away.

In addition to presenting information to museum visitors, we needed to provide a simple way for museum staff to add, edit, and delete the information that describes the art pieces within the application. This simplicity is critical because the program is unlikely to be used if it is not intuitive, and many members of museum staff are not technology-focused in their jobs. In order to accomplish this, we have set up our content management system in such a way that museum staff will be able to fill out a simple form in order to add, remove, and edit content on the site. Adding new content to the site will cause the form to create a barcode that can be printed and displayed next to the exhibit, providing a link to the newly added information.

We decided to use barcodes in order to link users to the correct art piece in our web application due to their low cost, simplicity, and ease of use. Other options that were considered included image recognition, proximity sensors, and the manual entry of an ID for each art piece, but all of these

options were lacking in at least one aspect. For instance, considering its complex implementation, image recognition does not provide much of an improvement over barcode usage. For either system, the user will need to point their phone in the direction of the art piece, and image recognition is less reliable and could require the museum to address copyright issues that can arise from uploading art piece images. Existing image recognition software is also very expensive, likely outside of the budget of the Knoxville Museum of Art. Proximity sensors also proved to be unacceptable due to cost. Finally, simply labeling each piece of art with an ID is just as much work for the museum staff as providing a barcode, and it requires significantly more work on the part of the visitor, so barcodes proved to be the most promising technique for user access of content.

In order for our project to be successful, we needed to ensure that our team would be able to work together efficiently. At first, we assumed that Github would be the best way for us to collaborate, but following extensive research, we settled on using Visual Studio Team Services due to its ease of integration with Visual Studio, our integrated development environment of choice. The Github plugin for Visual Studio is developed by a third party and therefore has many bugs and problems with its integration, while Team Services is developed by the Visual Studio team itself and therefore runs seamlessly within Visual Studio.

The last major design piece our group needed to research was how to incorporate the audio within our web application. We wanted to ensure that the audio clips would continue to play at the bottom of our web application regardless of whether the customer browsed away to another page, similar to how Spotify keeps music playing throughout its application even if the user hits the back button to browse for more music. In order to figure out how to make this possible, we had to research the process of developing a universal audio player in our web application.

4.2 Design Prototype

Once we had a basic architecture for our project design based on our customer requirements, we needed to prototype our design. For the front-end of our application, we did all of our prototyping through Pixate. Pixate allowed us to create a design prototype in Photoshop, which could then be transferred into Pixate. From there, Pixate allowed us to load the prototype onto our smartphones, and we could see exactly how the design would look

on our phones. We could then go back into Photoshop to tweak our design based upon the results from the Pixate demo.

The user experience for an application like this is extremely important. For an art museum, the experience of the art can be just as important as the art itself. Therefore, when we considered the user experience of the application, everything needed to be seamless and distraction free. We accomplished this by combining several design methods. The most important was the decision to use a universal audio player, ensuring that even if the user navigates away from the art piece page, the audio would remain playing. We accomplished this by using Ajax to generate our pages.

As far as the layout goes, we planned on creating several ways of accessing a specific art piece. When a user goes to the main page, they will see the main exhibit list. Clicking on an exhibit will then take them to a listing of all the art pieces in the order in which they are meant to be viewed. This order would have to be determined by the museum curator. Clicking on an art piece will then bring up a description of the art piece with a title, description, and an audio player if audio is available.

The overall design is based on Google's Material Design, which values classic principles of good design. With this design, we also incorporated the Knoxville Museum of Art branding into the design of the web application and elements of the overall look. In Material Design, a list would look like a normal box, but we incorporated the Knoxville Museum of Art logo's shape and the primary colors in the selected art piece. The incorporation of primary colors from the art piece will allow for seamless interaction between the piece itself and the web application. For instance, the Knoxville Museum of Art's primary logo color is a bright pink, but if the user is looking at an art piece that is similar to Picasso's blue period, the user may experience a bit of a disconnect if they see the logo's bright pink within our application. Therefore, we take the primary colors from the art piece and incorporate that color into the view of the individual art piece. We wanted to ensure that when the user looks at a blue painting, they will see a blue color in our application that is a similar shade to the one in the art piece. All of these design elements and features were incorporated into our prototype on Pixate.

4.3 Application Development

Following the prototyping phase of our project, we focused on the development phase of our application. In order to begin the actual development

process and stay on track, our first step was to develop a feature list and user stories implementing the Agile methodology for software development.

The following is our list of features we need to include:

1. View a gallery of all art pieces
2. Sort pieces by exhibit
3. Scan barcode to go to the URL of a piece
4. View individual piece
5. Search pieces by title
6. Search pieces by art tag
7. Search pieces by artist
8. Add a new piece
9. Edit an existing piece
10. Delete a piece
11. Browse all pieces in content management system (CMS)
12. Log into the content management system (CMS)
13. Create a new user for content management system (CMS) privileges
14. Delete a user with content management system (CMS) privileges
15. Change the password for an existing user with content management system (CMS) privileges

Our corresponding user stories for these features are:

1. As a user, I can view a gallery view of all art pieces.
2. As a user, I can sort pieces by exhibits.
3. As a user, I can scan a barcode to go to the URL of a piece.
4. As a user, I can view an individual piece.

5. As a user, I can search pieces by title.
6. As a user, I can search pieces by art tag.
7. As a user, I can search pieces by artist.
8. As the customer, I can add a new piece.
9. As the customer, I can edit an existing piece.
10. As the customer, I can delete a piece.
11. As the customer, I can browse all the pieces.
12. As the customer, I can log into the content management system (CMS).
13. As the customer, I can add a user to the content management system (CMS).
14. As the customer, I can delete a user with content management system (CMS) privileges.
15. As the customer, I can change a password for an existing user with content management system (CMS) privileges.

For the back-end of our application, we needed to ensure that our solution was feasible by first setting up a MySQL database on Azure services. Once our database was created, we used an application called Toad for MySQL to manage the database. Toad for MySQL allowed us to individually create each of the MySQL tables we would need for the web application and associated content management system that would allow the Knoxville Museum of Art staff to create and edit the content within our web application.

For our C# .NET web application, we hooked into the database using Entity Framework by Microsoft. This allowed us to easily connect to MySQL using C#, and if the organization ever migrates to a different database like Oracle or SQL, they will not have to rewrite our application. It also allows an easier way for us to program quickly as we can easily perform search functions using C# .NET library methods instead of SQL queries that may be difficult to read, giving us an easily maintainable product.

In order to maintain the security of our content management system, we needed to ensure that only the Knoxville Museum of Art staff would be able to

edit the content within our application. In order to achieve this, we needed to password-protect our content management system. Luckily for us, the technology we were already planning to use for our web application made it easy for us to build user authentication into our content management system. Each ASP.NET MVC 5 application provides the option to use the ASP.NET Identity membership system, which takes care of the majority of the work required to allow for user accounts, email confirmation, and password reset.

Our next step towards the completion of our application involved figuring out how to allow the museum staff to upload image and audio files associated with each art piece and artist from within the content management system. Due to the fact that none of us had extensive knowledge of how to work with a MySQL database, we were unsure whether we should store our image and audio files directly in the MySQL tables or if we should store them on our web application's server. Following a great deal of research, we decided it would be most efficient to upload the image and audio files to our server and then store the URL's to their location on the server within our MySQL database. In order to accomplish this, we first had to create a new database table specifically for images and another database table specifically for audio. Within these tables, we created a column to specify which art piece or artist the image or audio file should be associated with. With the backend complete, we next needed to create forms within the content management system of the application to allow for the Knoxville Museum of Art staff to upload image and audio files. For the audio upload form, while we could have simply used an HTML file upload form, we wanted our upload form to have drag and drop capabilities in order to make the process as simply as possible for the Knoxville Museum of Art staff. In order to create this drag and drop functionality in an efficient manner, we turned to the Javascript library called DropzoneJS. DropzoneJS is an open source Javascript library that provides drag and drop file uploads, as well as image previews. We chose DropzoneJS due to the fact that the library is incredibly lightweight, doesn't depend on any other libraries, and is highly customizable. Using DropzoneJS's configuration methods, we were able to seamlessly fit DropzoneJS file upload forms into our application while preserving our application's existing look and feel.

A major functional piece of our application is the barcode scanner that allows users to scan a barcode beside each piece within the Knoxville Museum of Art and have our application take them to the correct piece within our application. When we first began development on this project, we initially intended to use a QR code scanner to accomplish this, but due to the fact that

we were building a web application instead of a mobile application, we needed to use Javascript for our code scanning. After conducting extensive research, we realized that there simply were not any viable Javascript QR code scanners available on the web at the moment. This put us in the position of having to either write our own Javascript library to scan QR codes or completely pivot our plans for implementing this functionality. Because none of us have the Javascript experience necessary to develop a QR code scanner on our own, we needed to investigate other options. The most obvious of these was to instead use a barcode scanner. Barcodes would allow us to provide a similar experience for the user as with QR codes, but due to the popularity of barcodes, more Javascript libraries are available for scanning barcodes. Although many Javascript libraries do exist, it was still incredibly difficult to find one that worked well and worked consistently, particularly on mobile devices. We finally settled upon using QuaggaJS, a barcode scanner written entirely in Javascript that supports the real-time localization and decoding of various barcode types, in particular the Code-128 barcode type that we intended to use. After deciding to use QuaggaJS, it was not too much trouble to integrate the Javascript library into our own application to allow users to navigate to the correct art pieces within the application.

4.4 Test Plan

Before we turned our finished product over to our customer, it was vital that we planned and executed tests for all aspects of our completed web application. This involved going through our initial requirements list and developing tests to meet those requirements.

The first important functionality within our application to test is our barcode scanner. In order to test that the scanner works correctly, we planned to print out various barcodes and ensure that our application is able to correctly identify the art piece IDs from the barcodes. The application should then navigate the user to the page in our application that is associated with that art piece.

A second vital piece of our application is the audio player. The audio player within our application must be able to play the audio file associated with an art piece, as well as provide forward and backward buttons to navigate through the art exhibit.

In the full list of art pieces and artists, the users must be able to search through the list. In order to test this, we will input various correct and

incorrect art piece and artist names into the search bar and see if they produce the expected result.

Moving to the side of the content management system, we first needed to test if the customer, the Knoxville Museum of Art staff, would be able to add, edit, and delete any art piece, artist, or exhibit from the application. We also needed to ensure that only the Knoxville Museum of Art staff would be able to access the content management system and create, edit, or delete users.

4.5 Tests

Barcode testing consisted of reading code 128 barcodes across a variety of devices. We deployed our scanner in the web application on multiple platforms. Multiple types of barcodes were used, including small barcodes, large barcodes, barcodes with text surrounding them, and low-quality printed barcodes.

We deployed the audio player on multiple devices. We then tested media controls. Next, we maneuvered to another page and confirmed continued media playback. Controls were tested again on the new page.

CMS testing was once again deployed on numerous devices. Success would be determined by appropriate responses to keywords. In addition, multiple users on different devices will attempt to add, modify and delete content. They will attempt to do this with different pieces of content. This includes art pieces, artists, and exhibits.

Finally, different users of varying levels of skill attempted to add accounts. Email verification to confirm account creation would be sent. They then attempted to edit their account, e.g., changing name, permissions, and email addresses. Lastly, they deleted a fake account and then their own and were automated logged out.

4.6 Test Results

The application was able to scan code 128 barcodes using a variety of devices. The code read from the barcode was used to navigate to the webpage related to the associated art piece.

On the webpage associated with an art piece an audio player appeared. The audio player can be maximized for increased functionality or minimized in order to allow the user to better see information related to the art piece.

The audio player persists within the web application such that audio will continue playing while other pages can be visited (such as the web page containing the artist's information).

Users were able to search the database of art pieces and artists. The web application will auto-complete searches and alerts users if their search returned no results. Users could click on the artist or art piece they wanted to learn more about and be taken to the appropriate web page.

Users with valid authentication were capable of adding, editing, and deleting artists, art pieces, and exhibits. Users lacking authentication were not able to alter any of this information.

It is possible to add new users to the system, although there is not currently any functioning email verification due to the fact that we do not currently have access to the Knoxville Museum of Art's email server.

Our application currently works exactly as expected, and we are prepared to hand over a completed, tested product to the Knoxville Museum of Art for deployment.