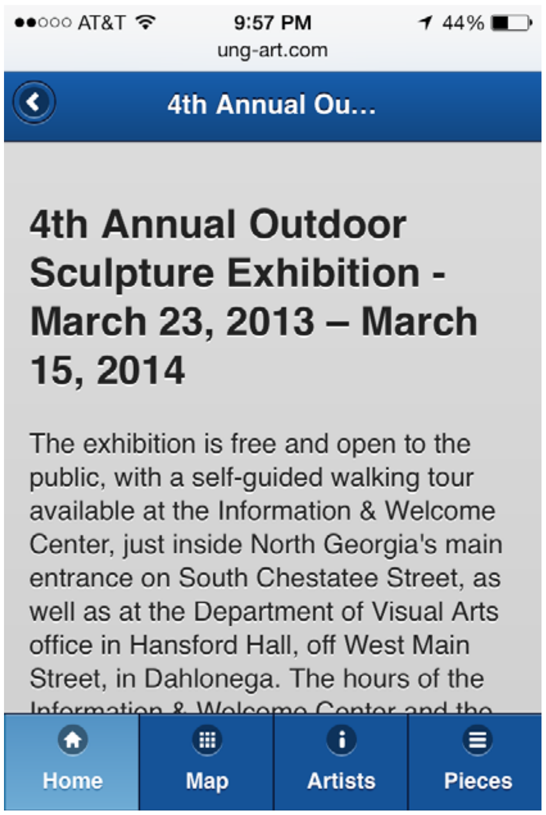


4th Annual Outdoor Sculpture Exhibition - March 23, 2013 – March 15, 2014

The exhibition is free and open to the public, with a self-guided walking tour available at the Information & Welcome Center, just inside North Georgia's main entrance on South Chestatee Street, as well as at the Department of Visual Arts office in Hansford Hall, off West Main Street, in Dahlonega. The hours of the

UNG ART DEPT. MOBILE WEB APPLICATION

Senior Project

ABSTRACT

The design and implementation details of a mobile web app.

Jordan Quattlebaum

CSCI 4950 – Senior Project

Contents

.... 1
Overview.. 2
Current Webpage.. 2
Requirements for App.. 3
Native vs. Web.. 4
JavaScript Geo-Location API.. 5
Google Maps API.. 8
Database Design.. 8
Administration Area.. 10
JQuery Mobile.. 12
Future Improvements.. 15

Overview

The UNG Art Department made a request for an iPhone application. It was described as a "walking tour" app. The purpose of the app would be to provide information about the Outdoor Sculpture Exhibition on the University's Dahlonega Campus.

Current Webpage

The Art Department's current informational webpage for the exhibition provides a map, photos of sculptures and artists' biographies. While the page is informative, it is difficult to navigate, especially on a mobile device.

The map provided displays locations for the sculptures in relation to the various buildings on campus. The locations of the sculptures are indicated with star icons which contain a number.

The map is perfectly suitable for a user familiar with the University campus; however, for users that are new to the campus or visiting, the map provides little useful information.

The map is followed by a numbered listing of the sculptures. Each listing displays a photograph, title and artist name. Users must refer to this list to pinpoint the location of a specific sculpture on the map, which requires scrolling between the two.

Artists' biographies are displayed below the list of sculptures. Again, users most scroll between the list of sculptures and the artists' biographies to discern the relation between sculpture and artist.

The width property of the webpage is obviously set to a specific numerical value causing text to be too small to discern on some mobile devices. In order to read the text users can zoom but this causes long lines of text to extend beyond the right edge of the display area on some mobile devices.

While these aspect do not make the site unusable, they do have the potential to cause frustration for users.

Requirements for App

The desired features for the requested app were:

- display information about the various sculptures
- display a map of the locations of the sculptures
- display the user's location relative to the sculptures
- alert the user when he or she comes in close proximity to a sculpture

- display artists’ biographies
- provide a means for the art department to easily update the information

Though the request was specifically for an iPhone app, Android phones now make up over fifty percent of the mobile device market share. It was eventually decided that the app should be multi-platform in order to appeal to as many users as possible.

Native vs. Web

Building native apps would provide unfettered access to all sensors available on devices, but considering the fractured nature of the android market share it seemed the possibility of depending on the presence of any particular sensor would be unwise. That aside, building natively would provide better performance. After the initial app download, devices would not be required to have internet access. However, the negative aspects of building natively seemed overwhelming. The basic functional logic of the app on each platform would be consistent, but presentation logic could be completely different. It would become necessary to become, at the very least, familiar with the native API for each type of device. This could be avoided by using a multi-platform API such as Sencha or PhoneGap, but it would still be necessary to have access to each type of device for testing purposes.

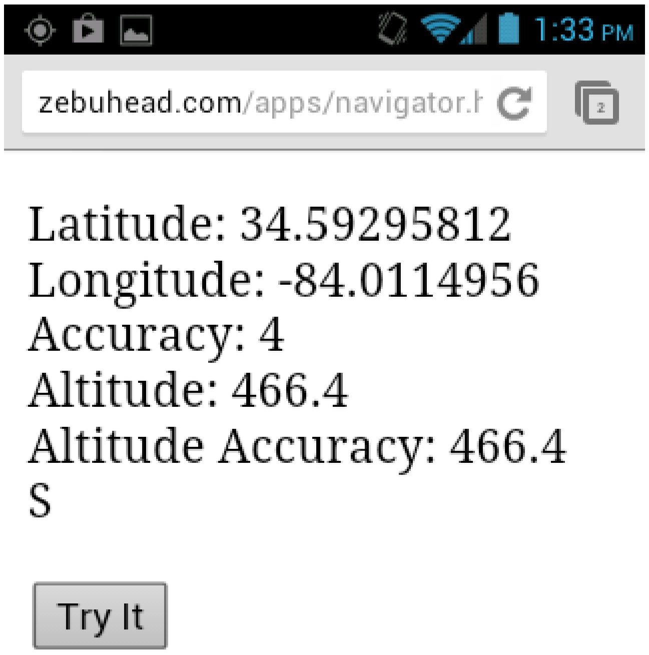
Building a web-based version of the app seemed to be the most time-efficient choice. Development and much of the testing could be performed on a PC. The same code would function nearly the same on all devices with a modern web browser. Furthermore, it would not be necessary to go through any app store approval processes. The disadvantages of a web-based app were still significant. Devices would be required to have nearly constant internet access. The app would have limited access to sensors and be dependent on the capabilities of various web browsers.

JavaScript Geo-Location API

The JavaScript geo-location API provides functionality to obtain latitude, longitude, accuracy, altitude, altitude accuracy and heading. A simple page was created using this API and several different types of devices were tested.

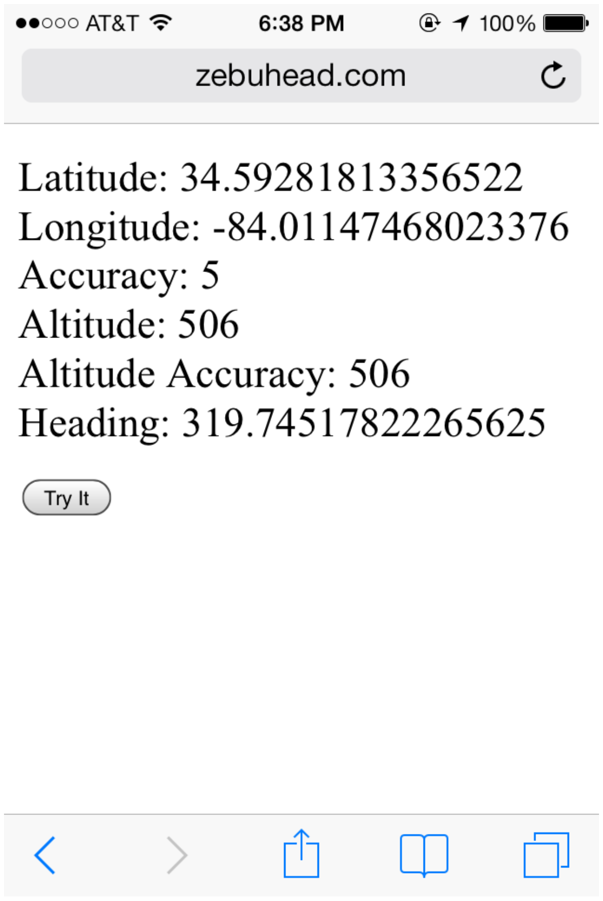
Source Code For JavaScript Geo-Location Test Page

```
<html>
 <body>
  <p id="demo">Click the button to get your coordinates:</p>
  <button onclick="getLocation()">Try It</button>
  <script>
   var dir = null;
   var x=document.getElementById("demo");
   function getLocation()
   {
     if (navigator.geolocation){
      navigator.geolocation.watchPosition(showPosition);
      //Get Orientation
      window.addEventListener('deviceorientation', getOrientation);
     }
     else{
      x.innerHTML="Geolocation is not supported by this browser.";}
   }
   function showPosition(position)
   {
    x.innerHTML="Latitude: " + position.coords.latitude +
    "<br>Longitude: " + position.coords.longitude +
    "<br>Accuracy: " + position.coords.accuracy +
    "<br>Altitude: " + position.coords.altitude +
    "<br>Altitude Accuracy: " + position.coords.altitude +
    "<br><span id='heading'>S</span>";
   }
   function getOrientation(eventData)
   {
     //just care about the compass heading
     var alpha;
     if (event.webkitCompassHeading){
      //this is an iOS device
      alpha = event.webkitCompassHeading;
     }
     else{
      alpha = event.alpha;
      }
      document.getElementById("heading").innerHTML= "Heading: "+alpha;
    }
  </script>
 </body>
</html>
```



zebuhead.com/apps/navigator.h

Latitude: 34.59295812
Longitude: -84.0114956
Accuracy: 4
Altitude: 466.4
Altitude Accuracy: 466.4
S



zebuhead.com

Latitude: 34.59281813356522
Longitude: -84.01147468023376
Accuracy: 5
Altitude: 506
Altitude Accuracy: 506
Heading: 319.74517822265625

SCREEN CAPTURE OF TEST PAGE ON ANDROID 4.1

SCREEN CAPTURE OF TEST PAGE ON IOS 7

The latitude and longitude of each device could be obtained with an accuracy between two and five meters when outdoors. Indoors the accuracy could vary greatly, with observed values between five meters and a few hundred meters.

The altitude and altitude accuracy could be obtained on some devices, but not all. The devices that did display altitude information were reasonably accurate when compared to standard GPS devices and accepted values for altitude. However, the altitude accuracy given by the API was usually close to the actual altitude. For example, the given altitude is 450 meters with an altitude accuracy of 450 meters. It seems reasonable to assume that this means “the device is around 450 meters, plus or minus 450 meters,” or the device could be at any altitude between zero and 900 meters above sea level. Though the observed values returned for altitude were accurate, it seemed that ultimately the value cannot be trusted to be accurate at all times.

In using the basic JavaScript geo-location API it is possible to obtain heading information once the device starts moving, which it accomplishes by using previously obtained coordinates to determine direction of movement. Because of this implementation, the geo-location API is not able to detect if a user is rotating in place. Therefore, it cannot be used to accurately determine the direction in which a user is facing. It is possible though, once heading information is obtained, to rely on the JavaScript accelerometer API to determine further heading information. If the device is rotated then the heading has changed. The potential problem with this approach is that several assumptions must be made. First, it must be assumed that the user was holding the device in a position that is conducive to being viewed when the initial heading information was obtained, i.e. the top of the screen was in an upward position or the user was looking downward on the device and the top of the screen was oriented toward the direction of movement. The next assumption that must be made is that the user will continue to hold the device in the same relative orientation. If these assumptions are not true then using the accelerometer to track heading will not work and we must rely on the Geo-location to provide heading information based on movement. iOS devices can immediately display heading information by using an Apple WebKit specific extension for JavaScript that makes uses of a device's compass.

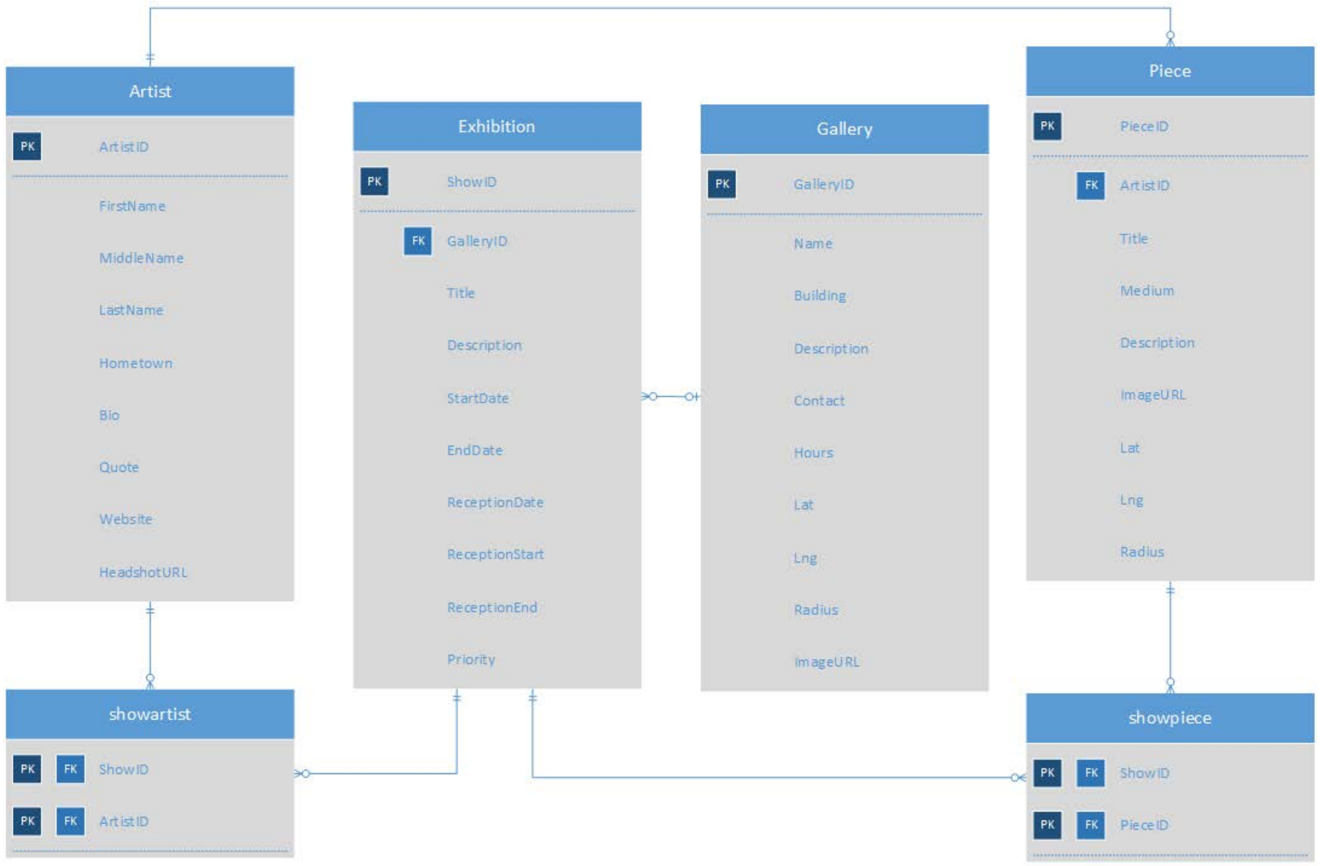
Google Maps API

The Google Maps API was utilized to fulfil the mapping requirements. The API provides a simple way to display a map within a browser window and also allows for markers to be placed on the map using latitude and longitude coordinates. Information from the database will be used to place markers for sculptures while coordinates obtained from the geo-location API will be used to display a user's location.

Despite the ease in using the API, there were issues encountered, though it was not a problem with the API. iOS devices have a "glitch" in which relative sizes are not properly processed. Setting the height or width of the element that acts as a container for the map to 100% causes iOS devices to display it much larger than the actual width or height of the screen. This also caused another realization: iOS 7 imposes a memory limit of 5 megabytes on all apps. The enormous size of the map easily caused the browser to exceed this limit and caused the browser to shut down. To avoid this issue, it became necessary to use JavaScript to poll the browser for the dimensions of the browser window before attempting to display the map.

Database Design

MySQL was chosen as the database system mainly because it is available on most web servers and is easily accessible with most server side scripting languages. The tables of the database where constructed with future possible uses in mind. Though the original request from the art department was for an app for the Outdoor Sculpture Exhibition, the database was constructed in such a way that the app could easily be used for multiple exhibitions with numerous attributes. The tables are in third normal form in order to easily allow many-to-many relations between the different entities. For example, each work of art can be associated with zero or more exhibitions while at the same time each exhibition can be associated with zero or more works of art (for the purposes of the database, a work of art is referred to as a “piece”). This was accomplished by having a separate table, called showpiece, “link” the two entities. This linking table has two columns, one for the exhibition and one for the piece. A similar table was used to link artists and exhibitions.

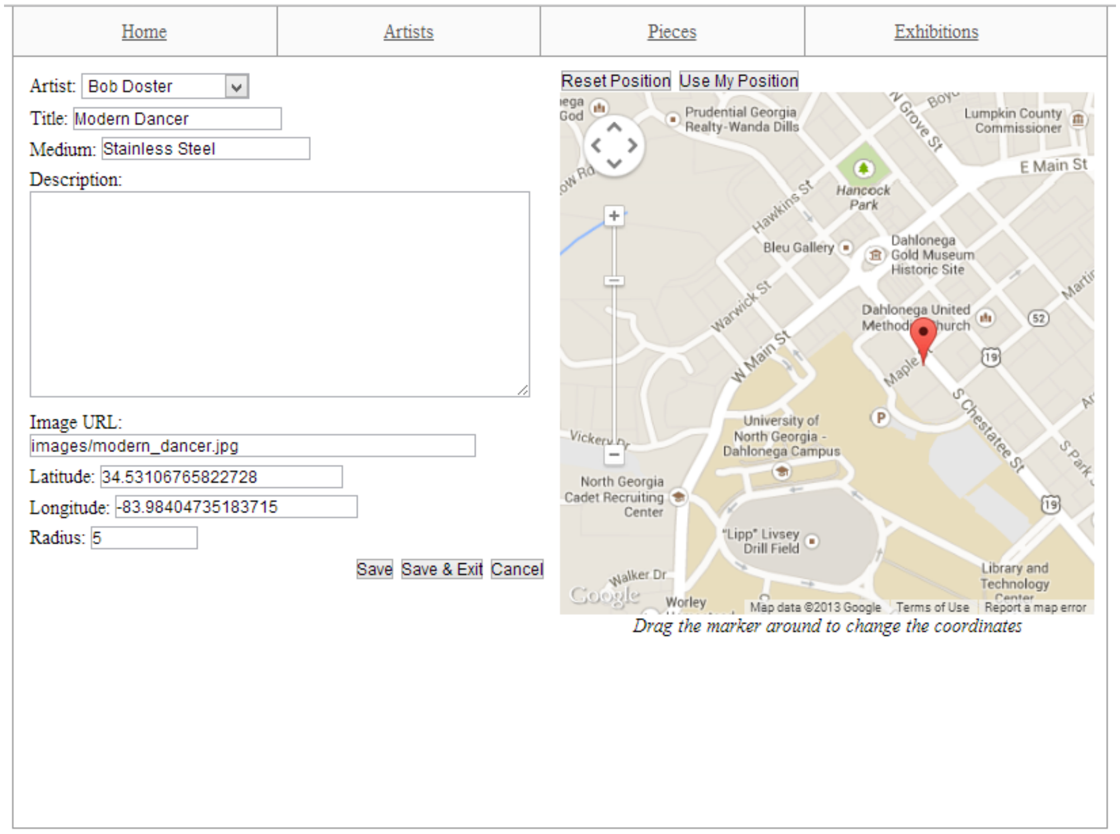


Entity Relation Diagram

A table was included for galleries in order to allow future exhibitions to be associated with a specific gallery. This provides a mechanism to allow users to see a map of their location relative to the gallery.

Administration Area

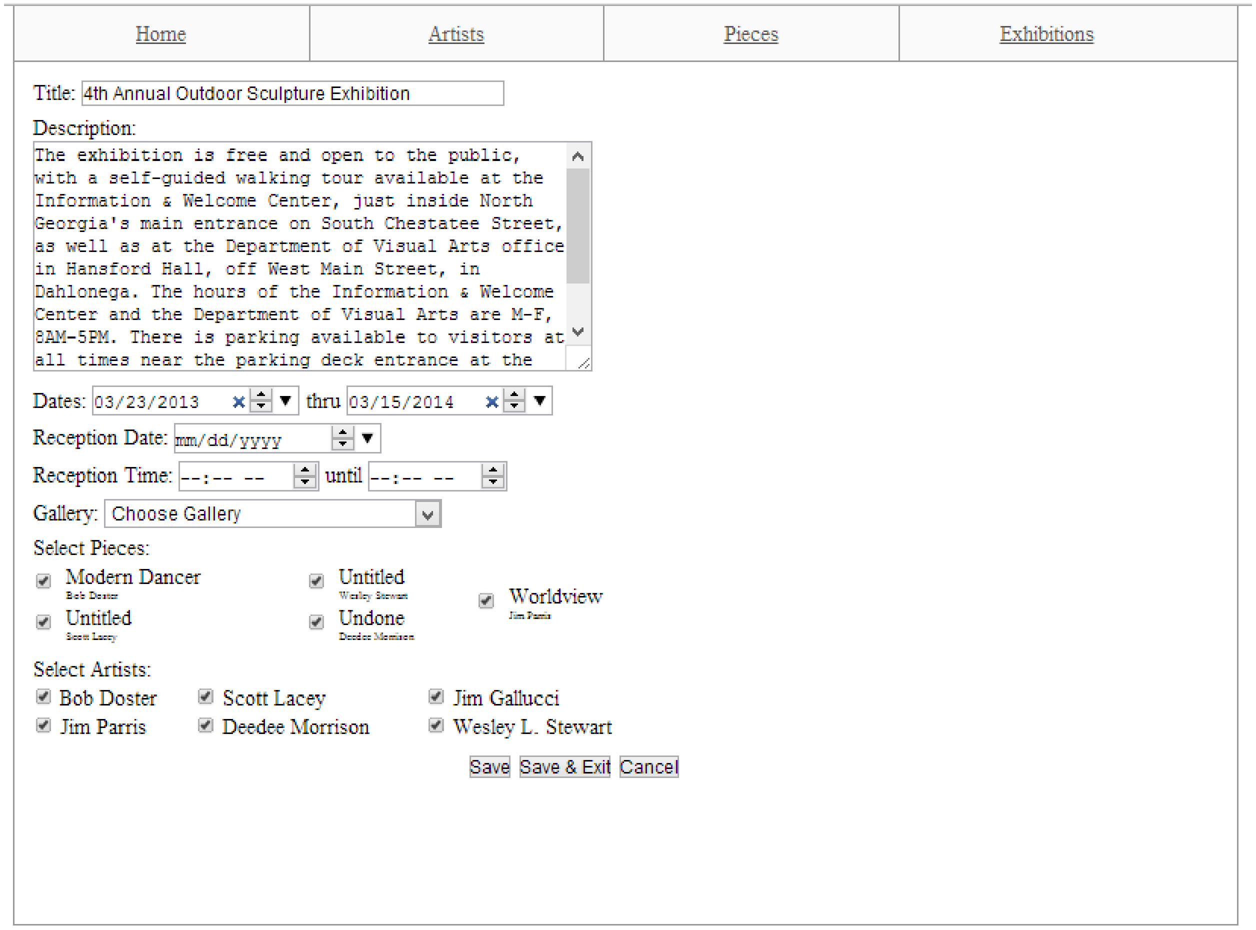
In order to provide a means for the art department to update the app, it was necessary to develop an administration area. This administration area is basically just a web interface that allows interaction with the database. This interface displays a section for each table in the database except the showpiece and showartist tables.



Artist: Bob Doster
Title: Modern Dancer
Medium: Stainless Steel
Description:
Image URL: images/modern_dancer.jpg
Latitude: 34.53106765822728
Longitude: -83.98404735183715
Radius: 5

Drag the marker around to change the coordinates

ADMINISTRATION AREA – EDITING A PIECE



Title: 4th Annual Outdoor Sculpture Exhibition

Description:

```
The exhibition is free and open to the public,
with a self-guided walking tour available at the
Information & Welcome Center, just inside North
Georgia's main entrance on South Chestatee Street,
as well as at the Department of Visual Arts office
in Hansford Hall, off West Main Street, in
Dahlonega. The hours of the Information & Welcome
Center and the Department of Visual Arts are M-F,
8AM-5PM. There is parking available to visitors at
all times near the parking deck entrance at the
```

Dates: 03/23/2013 thru 03/15/2014

Reception Date: mm/dd/yyyy

Reception Time: --:-- -- until --:-- --

Gallery: Choose Gallery

Select Pieces:

- ✓ Modern Dancer — Bob Doster
- ✓ Untitled — Scott Lacey
- ✓ Untitled — Wesley Stewart
- ✓ Undone — Deedee Morrison
- ✓ Worldview — Jim Parris

Select Artists:

✓ Bob Doster	✓ Scott Lacey	✓ Jim Gallucci
✓ Jim Parris	✓ Deedee Morrison	✓ Wesley L. Stewart

ADMINISTRATION AREA – EDITING AN EXHIBITION

Administration users can add, edit, and delete rows in any of the tables by going to the section for that table. Adding or editing an exhibition allows users to associate artists, pieces and galleries to the exhibition. This is accomplished by allowing users to place a check beside each artist that they wish to associate with the exhibition. When a piece is selected an entry is made into the showpiece table with the id of the exhibition and the id of the piece. The same is true of artists, except the entry is placed in the showartist table. In a similar manner, adding or editing a piece allows a user to associate an artist. However, since there can be only one artist associated with each piece, the user is presented with a drop-down list that only allows for one selection to be made.

Adding or editing a piece displays a map for the administration user to use to help determine the coordinates of the piece. The user can click a button to set the coordinates to the user’s current location. Of course, this will only give accurate results if the user is editing the information from a mobile device. The other option is for the user to drag the marker around on the map to set the piece’s location.

Currently the administration area is password-protected using an .htaccess file. The major disadvantage of this method is that it makes it impossible for the art department to add administrative users or change passwords on existing users without gaining access to the file structure of the server. Though this is not the most ideal way to implement a password-protected area, it was the most time-efficient method.

JQuery Mobile

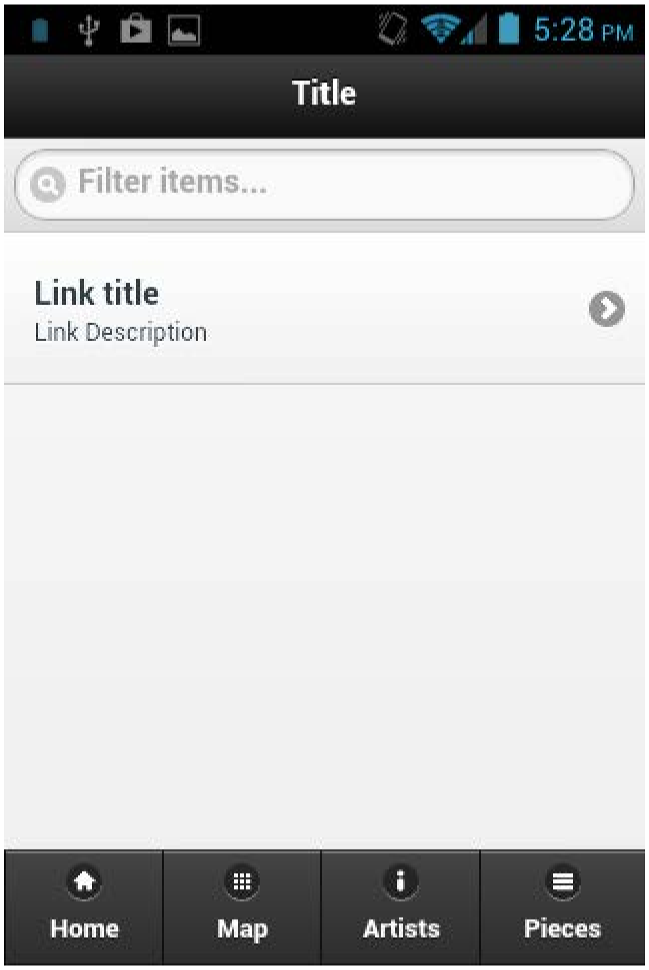
JQuery Mobile is a JavaScript API that uses the JQuery JavaScript library. The API provides a means to quickly generate a mobile compatible HTML user interface. The entire GUI can be developed with simple HTML. The only additional requirement is that the developer add special data attributes to some of the HTML elements. JQuery then process those data attributes to add CSS classes to the elements. These CSS classes are then used generate a mobile-compatible interface.

Source Code For Simple JQuery Mobile Page

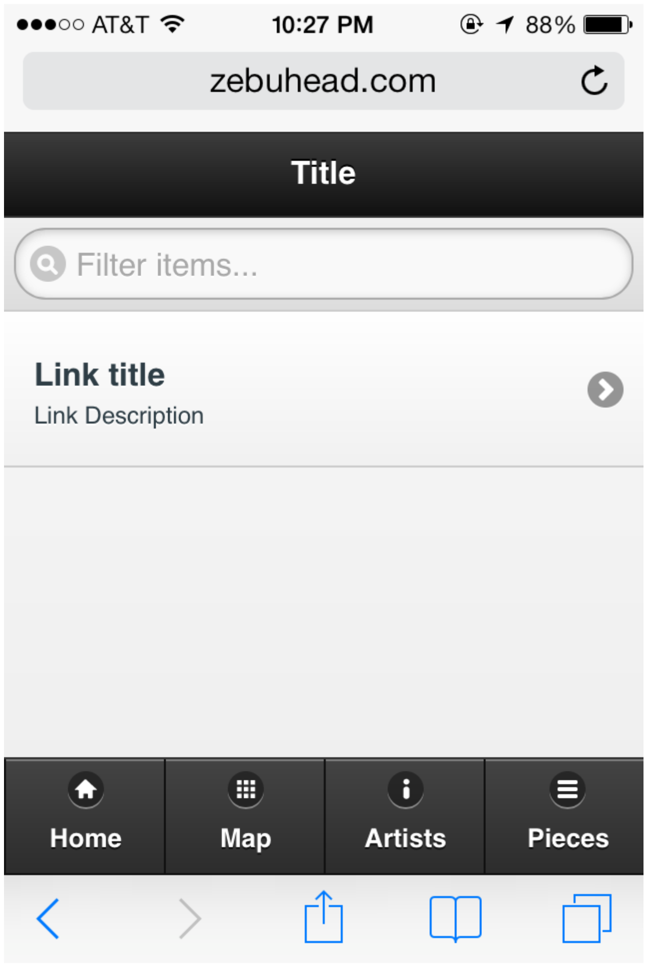
```
<html>
      <head>
            <meta name="viewport" content="width=device-width, initial-scale=1">
            <link rel="stylesheet"
href="http://code.jquery.com/mobile/1.3.2/jquery.mobile-1.3.2.min.css" />
            <script src="http://code.jquery.com/jquery-1.9.1.min.js"></script>
            <script src="http://code.jquery.com/mobile/1.3.2/jquery.mobile-
1.3.2.min.js"></script>
      </head>
<body>
    <div data-role="page" id="home">
        <div data-role="header" >
                  <h1>Title</h1>
        </div>
            <div data-role="content">
                  <ul data-role="listview" data-filter="true">
                        <li>
                              <a href="#">
                                    <h1>Link title</h1>
                                    <p>Link Description</p>
                              </a>
                        </li>
                  </ul>
            </div>
            <div data-role="footer" data-position="fixed">
                  <nav data-role="navbar">
                        <ul>
                              <li><a href="#" data-icon="home">Home</a></li>
                              <li><a href="#" data-icon="grid">Map</a></li>
                              <li><a href="#" data-icon="info">Artists</a></li>
                              <li><a href="#" data-icon="bars">Pieces</a></li>
                        </ul>
                  </nav>
            </div>
    </div>
</body>
```

The sample code above illustrates the ease with which a functional mobile-compatible webpage can be generated with HTML. The following screenshots display the output from the sample code on an iOS device and an Android device.

The effort required to generate a GUI for a web app is greatly minimized using a framework like JQuery Mobile. The API even makes attempts to provide a native-like experience for users by using AJAX for page loads and keeping previously viewed pages in the



SAMPLE CODE OUTPUT – ANDROID 4.1



SAMPLE CODE OUTPUT – IOS 7

document object model (DOM). This helps prevent unnecessary data usage. However, by using AJAX, only the content of subsequent pages are loaded. This means that a page cannot contain page specific JavaScript unless it is the first page loaded. There are two ways to solve this problem: One way is to disable the AJAX page loads and the other is to have all necessary JavaScript included on every page.

Future Improvements

There are features that would improve the user experience in the app, especially in administration area. One feature that would be extremely useful is to allow image uploads in the administration area. This is easily accomplished and was not included because of time constraints. Improvements could also be made in the way in which exhibitions are added and edited. As the database gets larger and more artists and pieces are added, the check list could become very unwieldy and cumbersome to use. Perhaps the most important change needed to the administration area is the way in which the users are authenticated. The most useful method for doing this would be to have an administration users table to store user names and hashed passwords. This table could be used to authenticated users as well as give the art department a means to add new administration users.

Conclusion

The completed mobile app satisfies all of the requested requirements. Although the complexity of the project was greater than originally anticipated, in the end the project came together in a fluid manner. The different APIs used function well together. The finished app is easily navigable and user friendly for both the front-end and back-end users.