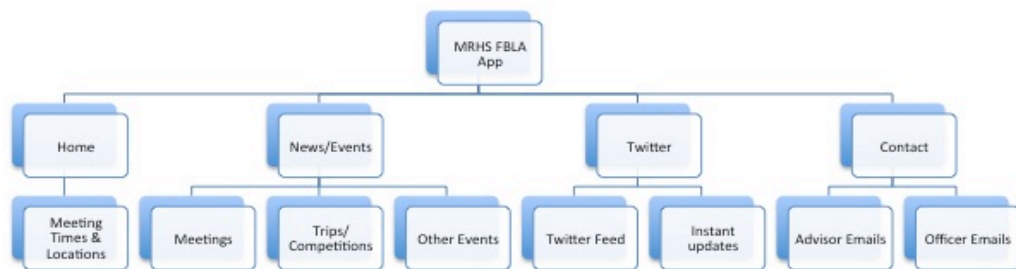


## Criterion E: Product development

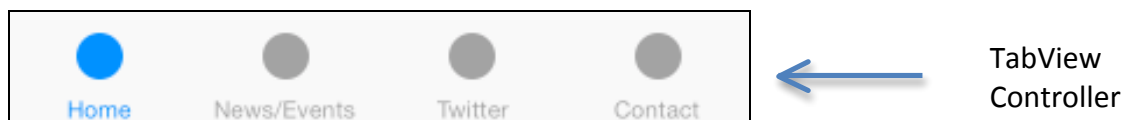
### Complex techniques used to address the client's requirements:

- Manipulation of Objective-C Code
- Manipulation of Graphics
- Navigation using frames and customized buttons
- Integration of components using advanced features from other applications

## Organization of the Application



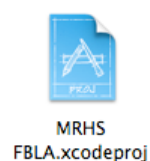
The content of the Application is set up in a tab form across the bottom of the application and managed by the TabView Controller. There are four tabs: Home, News/Events, Twitter, and Contact.



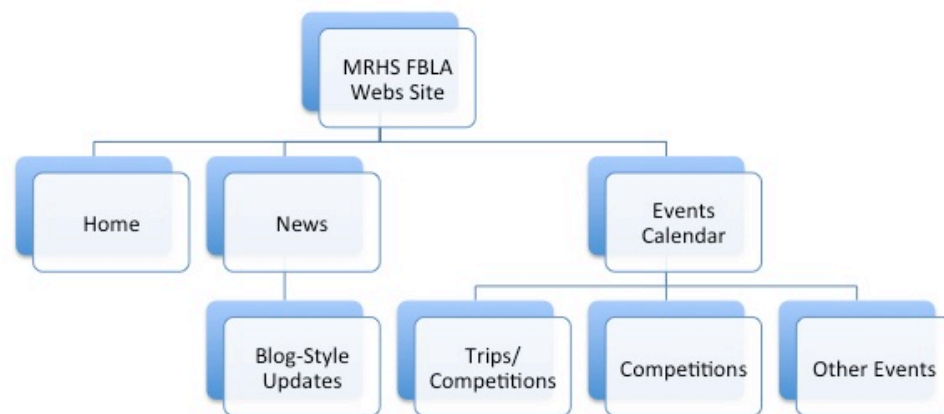
The News/Events feed and Twitter feed are both controlled by UIWebView Controllers, which display a Safari-like web frame. They are linked to their respective webpages and are represented in mobile view form.



The entirety of the app is contained within the MRHSFBLA.xcodeproj file with supporting files in the MRHS FBLA folder and MRHSFBLATests folder.

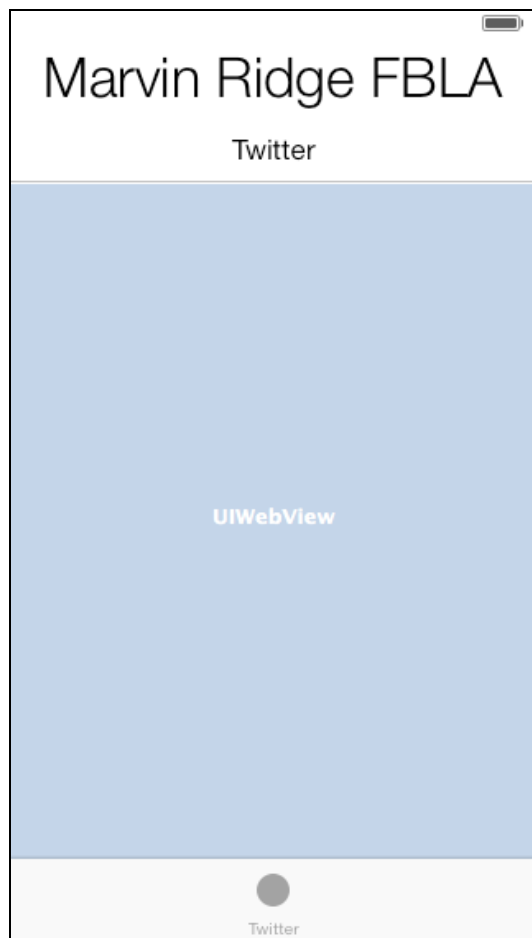


## Organization of the Backend website




The Webs website is set up in order to allow for a more accessible method for Ms. Gates to enter News/Event data into the News feed and Event Calendar rather than sifting through the mobile application's code and changing appropriate values and forcing users to update the application itself every time a new event is added. The application uses the mobile view version of this webpage to decrease unnecessary data loading.

## Manipulation of Objective-C Code



As the Home page was simply a reconciliation of labels to display information, code manipulation was not necessary.

With the Twitter view, the User-Interface was to be designed as shown to the left with two labels, the UIWebView, and the TabViewController.

 FirstViewController.h

```
#import <UIKit/UIKit.h>

@interface FirstViewController : UIViewController
@property (strong, nonatomic) IBOutlet UILabel *lblTwitter;
@property (strong, nonatomic) IBOutlet UIWebView *webTwitter;

@end
```

To code this interface, the header file (the FirstViewController.h title image) must be used to address the objects that are to be used within this view by associating themselves with the objects in the implementation file (on the following page). The label that simply is “Twitter” is declared as is the UIWebView titled “webTwitter”.

```

m FirstViewController.m
#import "FirstViewController.h"

@interface FirstViewController ()

@end

@implementation FirstViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.
    NSURL *twitterurl = [NSURL URLWithString: @"https://twitter.com/mrhs_fbla1"];
    NSURLRequest *twitterrequest = [NSURLRequest requestWithURL: twitterurl];
    [_webTwitter loadRequest: twitterrequest];
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

@end

```

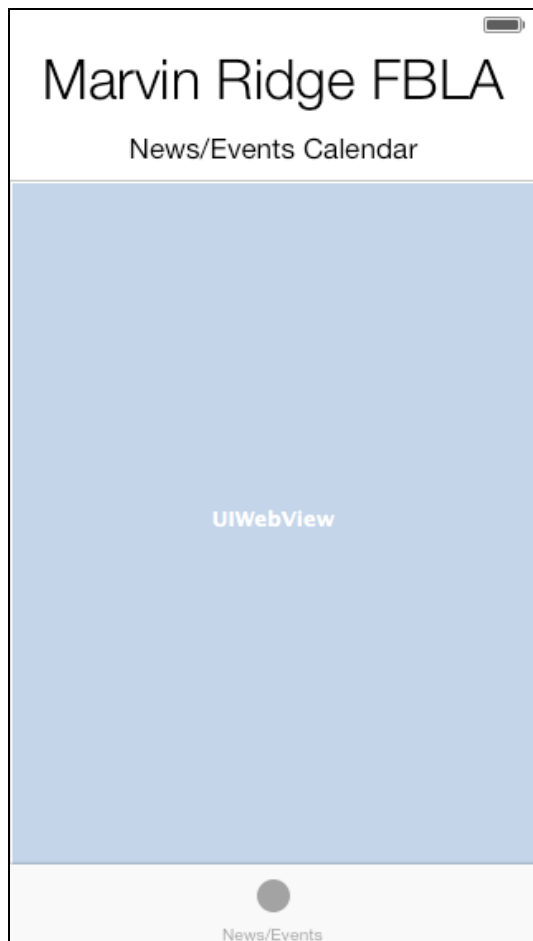
Upon initially opening the FirstViewController.m file (known as the implementation file) everything pictured above except for what is beneath the “//Do any additional setup after loading the view, typically from a nib.” command is given by the program as standard values.

The NSURL function was added to allow the pre load service to retrieve the web page that is to be displayed in the web view function (which is referenced to as “\_webTwitter”). This is then set equal to NSURL URLWithString which commands the website that is to be retrieved. Then loadRequest is assigned to \_webTwitter so that the webpage is displayed in the web view function in the application. This allows for the webpage to actually be visible to the user.

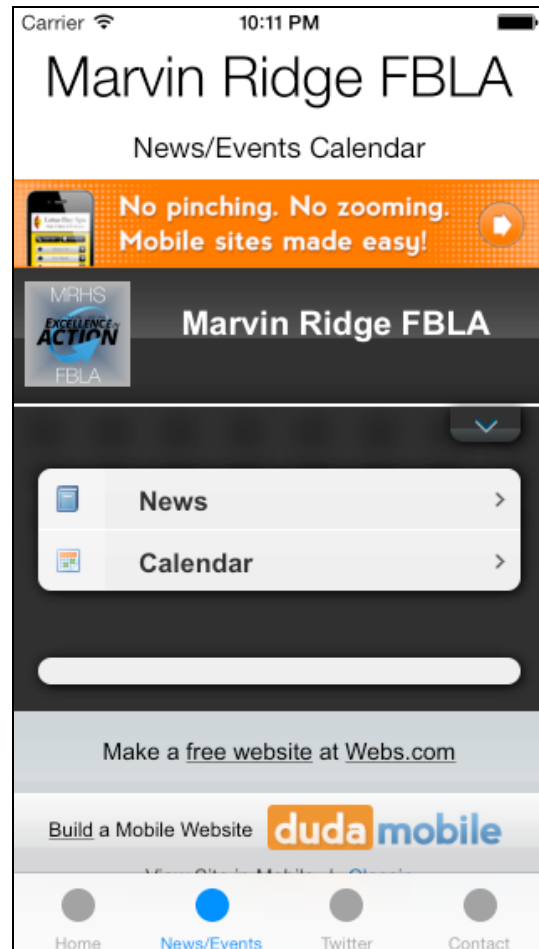


Completed Product Result

The entirety of this process is repeated with the News/Events view but replacing anything with “Twitter” to “News “ as represented below.



The User-Interface design view in Xcode



The completed News/Events view when using the application

## Manipulation of Graphics

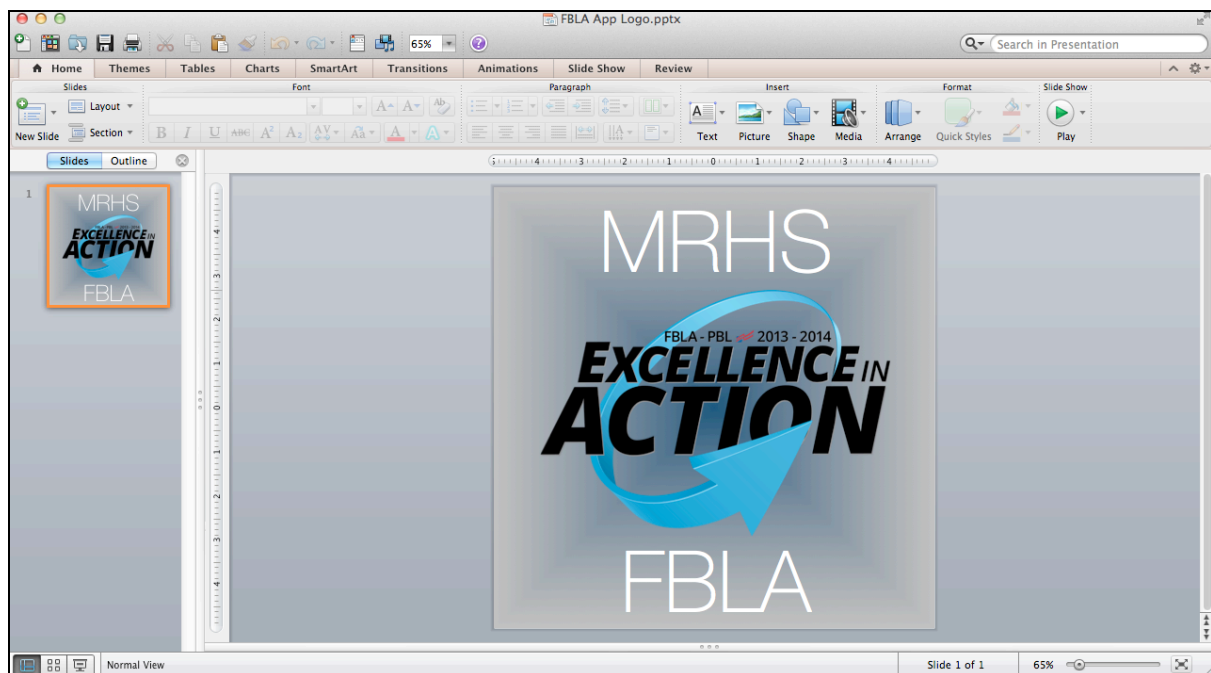


Image manipulation was used to create the App Icon so that no copyright violations would be made from a preexisting application in the App Store.

First, a blank PowerPoint slide was used as a basis. Then, a gradient background was added from the backgrounds pane.

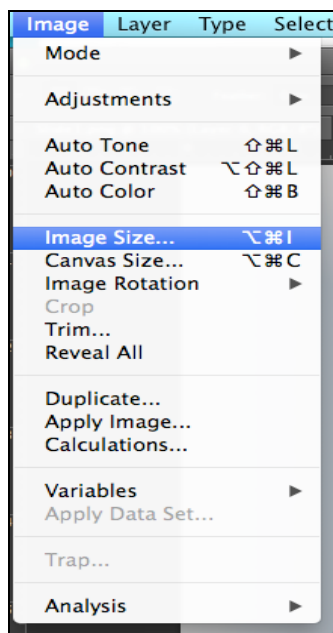
The “Excellence in Action” logo was taken from the FBLA national website ([www.fbla-pbl.org](http://www.fbla-pbl.org)). Then, the words “MRHS” and “FBLA” were added for personalization.

The project was then saved as a collection of .png image files and saved to the desktop.

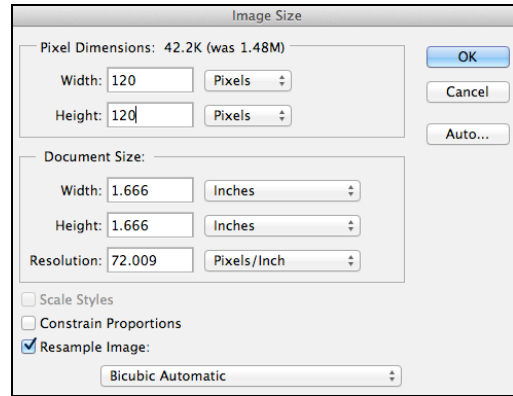
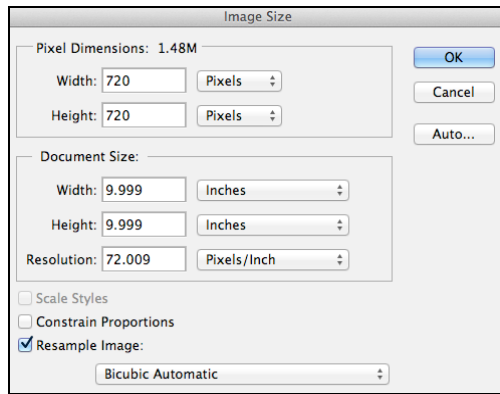
The image sizes of the slides were 720 by 720 and Xcode requires 120 by 120, 80 by 80, and 58 by 58 for icon images for the app itself, the spotlight icon in iOS, and the icon in settings. Thus, the images would need to be resized to meet Xcode requirements in order for the application to build correctly. Hence, Adobe Photoshop CS6 needed to be used.



Adobe Photoshop CS6 was used to resize the image down to Xcode standards. All that needed to be done was to access the “Image” drop down menu in the OS X taskbar and select “Image Size...”







After clicking “Image Size...” the above (left) window is displayed with the dimensions 720 by 720. The image size is then adjusted by changing 720 by 720 to 120 by 120 (as shown above, right), 80 by 80, and 58 by 58 to fulfill the aforementioned Xcode requirements.

These images can now be used as icons in the application.



120 by 120

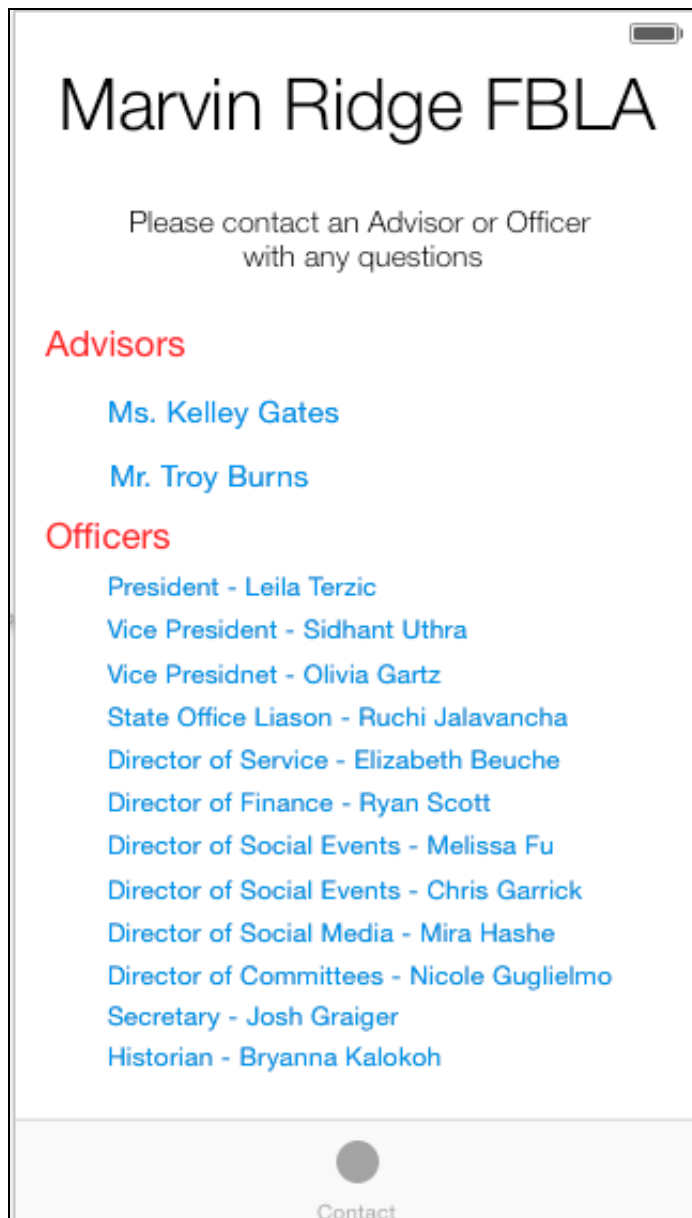


80 by 80



58 by 58

## Navigation using frames and customized buttons



Labels and buttons were used to organize the contact interface.

The labels serve as the title, the instructions, as well as the indication of Advisors and Officers in addition to being the hyperlinks that open the Mail Composition view.

This function allows members to communicate with officers and advisors directly and efficiently.

## FourthViewController.h

```
#import <UIKit/UIKit.h>
#import <MessageUI/MessageUI.h>
@interface FourthViewController : UIViewController <MFMailComposeViewControllerDelegate>
- (IBAction)btnMailGates:(id)sender;
- (IBAction)btnMailBurns:(id)sender;
- (IBAction)btnMailTerzic:(id)sender;
- (IBAction)btnMailUthra:(id)sender;
- (IBAction)btnMailGartz:(id)sender;
- (IBAction)btnMailJalavancha:(id)sender;
- (IBAction)btnMailBeuche:(id)sender;
- (IBAction)btnMailScott:(id)sender;
- (IBAction)btnMailFu:(id)sender;
- (IBAction)btnMailGarrick:(id)sender;
- (IBAction)btnMailGraiger:(id)sender;
- (IBAction)btnMailKalokoh:(id)sender;
- (IBAction)btnMailHashe:(id)sender;
- (IBAction)btnMailGuglielmo:(id)sender;
@end
```

In the FourthViewController.h header file, each button that comprises a name is declared as an action button with the naming scheme *btnMail(Name)*.

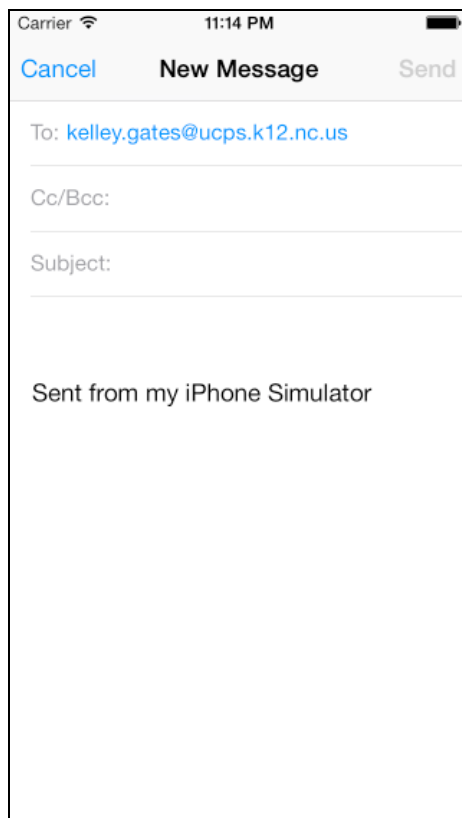
## FourthViewController.m

```
#import "FourthViewController.h"
@interface FourthViewController ()
@end
@implementation FourthViewController
- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
    if (self) {
        // Custom initialization
    }
    return self;
}
- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view.
}
- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}
- (IBAction)btnMailGates:(id)sender {
    // Email Subject
    NSString *emailTitle = @"";
    // Email Content
    NSString *messageBody = @"";
    // To address
    NSArray *toRecipients = [NSArray arrayWithObject:@"kelley.gates@ucps.k12.nc.us"];
    MFMailComposeViewController *mc = [[MFMailComposeViewController alloc] init];
    mc.mailComposeDelegate = self;
    [mc setSubject:emailTitle];
    [mc setMessageBody:messageBody isHTML:YES];
    [mc setToRecipients:toRecipients];
    // Present mail view controller on screen
    [self presentViewController:mc animated:YES completion:NULL];
}
```

In the FourthViewController.m implementation file, each button declaration is used to assign email functionality with an email address (an NSArray), subject line, and body. The subject line and body (which use NSStrings to determine the data that they carry) are left blanks so the user can self-define them. This coding process is repeated for each recipient specified as a button with the email address being replaced for each with its own respective address.

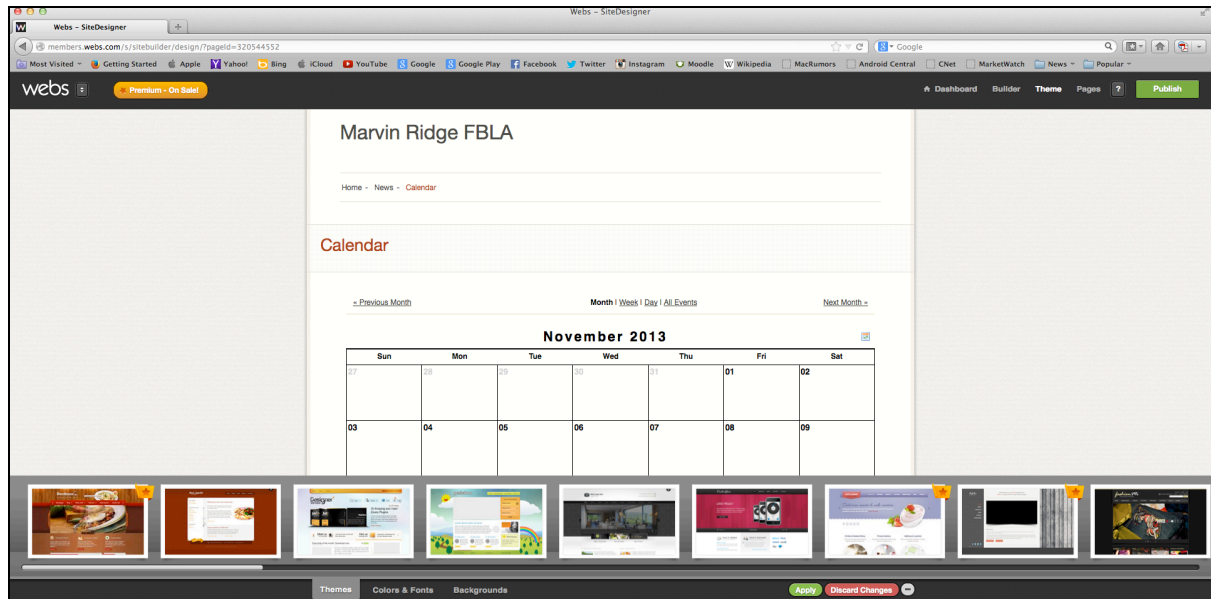
```
- (void) mailComposeController:(MFMailComposeViewController *)controller didFinishWithResult:(MFMailComposeResult)
    result error:(NSError *)error
{
    switch (result)
    {
        case MFMailComposeResultCancelled:
            NSLog(@"Mail Cancelled");
            break;
        case MFMailComposeResultSaved:
            NSLog(@"Mail Saved");
            break;
        case MFMailComposeResultSent:
            NSLog(@"Mail Sent");
            break;
        case MFMailComposeResultFailed:
            NSLog(@"Mail Sending Failure: %@", [error localizedDescription]);
            break;
        default:
            break;
    }
    // Close the Mail Interface
    [self dismissViewControllerAnimated:YES completion:NULL];
}
@end
```

The above code is then used to execute the command if the user choses to cancel the composed email. The void function commands the prompt when the user taps the “Cancel” button. The action then prompts the MailComposeViewController (mail interface) to close and displaying “Mail Cancelled”, “Mail Saved”, “Mail Sent”, and “Mail Sending Failure” in the command box on Xcode if the user’s device is connected.

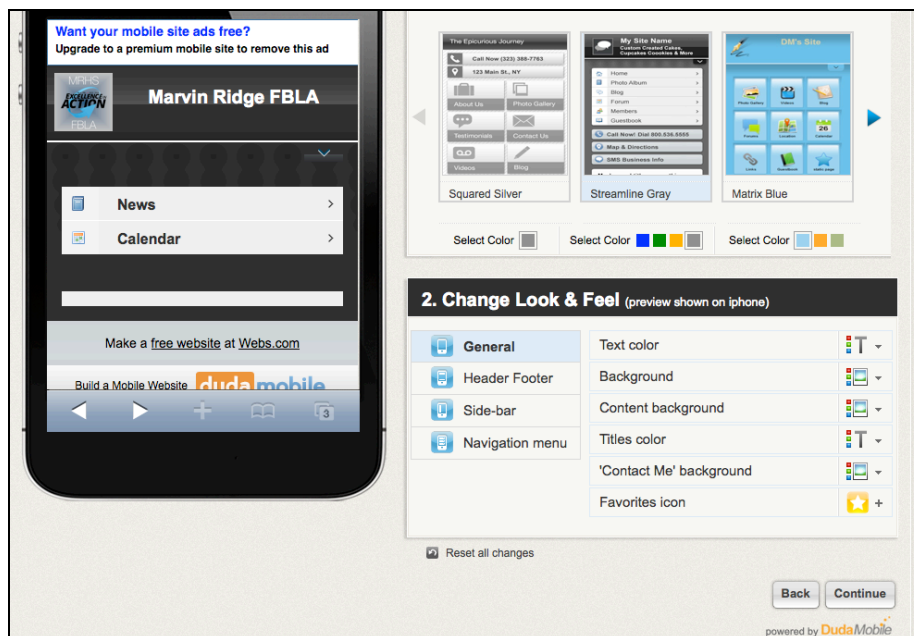


The final Mail Compose View  
(Sent from my iPhone Simulator  
will not appear on an iOS device)

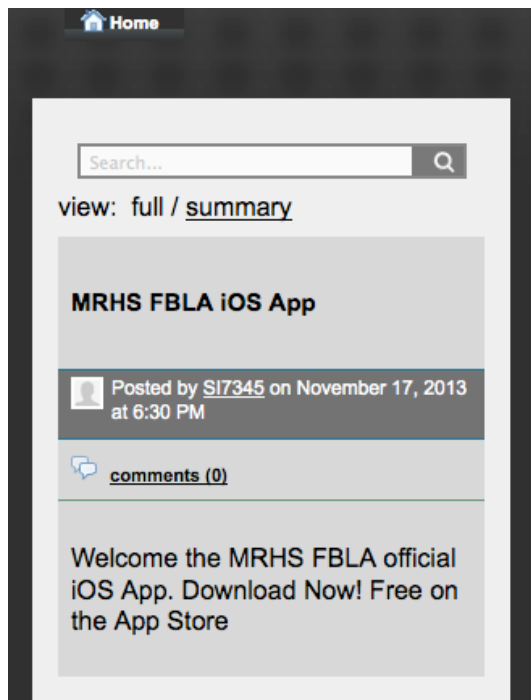
## Integration of components using advanced features from other applications



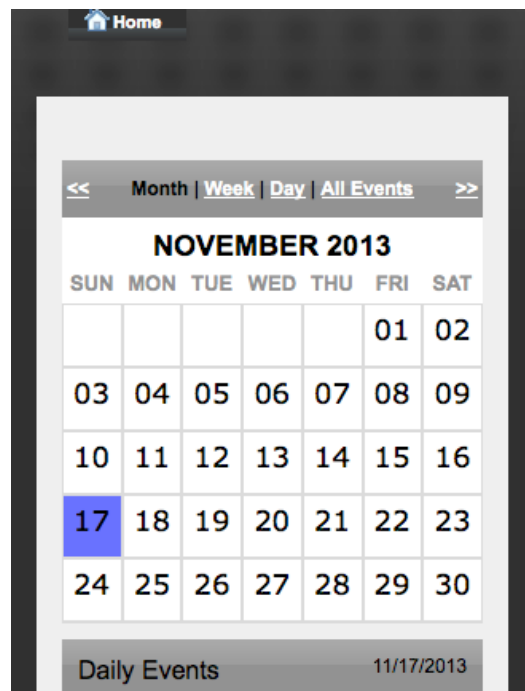
Using the webs.com service, a Webs website was created to host the News Feed and Event Calendar services. The above image shows the webpage on a computer running Mozilla Firefox.



The mobile version shown in the application was created using the DudaMobile tool available on webs.com. A template for the site was chosen and applied to the information and is displayed within the UIWebView on the News/Events tab.



Completed News View



Completed Event  
Calendar View

## Works Cited

Lewis, Rory, and Laurence Moroney. *iPhone and iPad Apps for Absolute Beginners*. 4th ed. New York: Apress, 2013. Print.

"iOS Developer Library." *iOS Developer Library*. Apple Inc., n.d. Web. 24 Feb. 2014. <<https://developer.apple.com/library/ios/navigation/>>.