```
//
//  ViewController.m
//  SFHS News
//
//  Created by Harish Kamath on 12/20/14.
//  Copyright (c) 2014 Harish Kamath. All rights reserved.
//

#import "ViewController.h"
#import "SWRevealViewController.h"
#import "EventInfo.h"
#import "SQLDatabase.h"

//Note: SDWebImage is a free, github project that is considered standard in many
    apps, including Facebook. This class is only used for faster loading of
    images from URL's. It is not used in any other ways.
#import "SDWebImage/UIImageView+WebCache.h"

@interface ViewController ()

@end

@implementation ViewController
@synthesize facebookButton, twitterButton, instagramButton,scrollView,awardsLabel
    ,pageControl,descLabel,sofoLogo,annNameLabel;

#define RGBColor(a,b,c) [UIColor colorWithRed:(a/255.0) green:(b/255.0) blue:(c/
    255.0) alpha: 1]

//First method called from class.
- (void)viewDidLoad {
    [super viewDidLoad];

    //Pertaining to the Banner View
    [scrollView setDelegate:self];
    pageControlBeingUsed = NO;
    [NSTimer scheduledTimerWithTimeInterval:8.0 target:self selector:@selector
        (flipPage) userInfo:nil repeats:YES];

    //Begins to fetch content from database.
    array = [[NSMutableArray alloc] init];
    [self fillArray];
    [self fillResults];
    timerArray = [NSTimer scheduledTimerWithTimeInterval:3.0 target:self
        selector:@selector(fillArray) userInfo:nil repeats:YES];
    timerViews = [NSTimer scheduledTimerWithTimeInterval:5.0 target:self
        selector:@selector(fillResults) userInfo:nil repeats:YES];

    //Aesthetic code
    self.view.backgroundColor = [UIColor blackColor];
    self.animator = [[UIDynamicAnimator alloc] initWithReferenceView:self.view];

    [self.navigationController.navigationBar setBackgroundImage:[UIImage new]
        forBarMetrics:UIBarMetricsDefault];
    self.navigationController.navigationBar.shadowImage = [UIImage new];
    self.navigationController.navigationBar.translucent = YES;
    activityIndicator = [[UIActivityIndicatorView alloc]
        initWithActivityIndicatorStyle:UIActivityIndicatorViewStyleWhite];
    activityIndicator.center = CGPointMake(scrollView.center.x, scrollView.center
        .y);
    [self.view addSubview:activityIndicator];
    [activityIndicator startAnimating];

    //Creates sideBar Menu
    SWRevealViewController *revealViewController = self.revealViewController;
    if ( revealViewController )
    {
        [self.navigationItem.backBarButtonItem setTarget: self.
            revealViewController];
        [self.navigationItem.backBarButtonItem setAction: @selector(
            revealToggle: )];
        [self.view addGestureRecognizer:self.revealViewController.
            panGestureRecognizer];
    }

    [NSTimer scheduledTimerWithTimeInterval:0.5 target:self selector:@selector
        (addAnimations) userInfo:nil repeats:NO];

    animation = [CATransition animation];
    [animation setDuration:0.5];
    [animation setType:kCATransitionPush];
    [animation setSubtype:kCATransitionFromLeft];
    [animation setTimingFunction:[CAMediaTimingFunction functionWithName:
        kCAMediaTimingFunctionEaseInEaseOut]];

    [facebookButton setTag:0];
    [twitterButton setTag:1];
    [instagramButton setTag:2];

    [facebookButton addTarget:self action:@selector(goToSocialMedia:)
        forControlEvents:UIControlEventTouchUpInside];
    [twitterButton addTarget:self action:@selector(goToSocialMedia:)
        forControlEvents:UIControlEventTouchUpInside];
    [instagramButton addTarget:self action:@selector(goToSocialMedia:)
        forControlEvents:UIControlEventTouchUpInside];


}

//Method used to fill array with banner objects.
-(void)fillArray{
    [array removeAllObjects];
    array = [[NSMutableArray alloc] init];
    [NSURLConnection sendAsynchronousRequest:[NSURLRequest requestWithURL:[NSURL
        URLWithString:@"http://sfhsnews.com/News/bannerPull.php"]] queue:
        [NSOperationQueue mainQueue] completionHandler:^(NSURLResponse *response,
        NSData *data, NSError *connectionError){
        if(connectionError){
            NSLog(@"%s: sendAsynchronousRequest error:%@", __FUNCTION__,
                connectionError);
            UIAlertView *alertView = [[UIAlertView alloc] initWithTitle:@"No
```

```
                Connection Found!" message:@"There seems to have been a problem
                with your connection. Please exit the app and try again."
                delegate:self cancelButtonTitle:nil otherButtonTitles:@"Okay",
                nil];
            [alertView show];
            return;
        }
        //Note: Data gathered from the database is in JSON Encoding.
        // NSLog(@"raw JSON: %@", [[NSString alloc] initWithData:data
            encoding:NSUTF8StringEncoding]);
        NSError *parseError = nil;
        NSArray *dataArray = [[[NSString alloc] initWithData:data encoding:
            NSUTF8StringEncoding] componentsSeparatedByString:@"}"];

        for(NSString *string in dataArray){

            //Divides JSON data into readable strings.
            if(string.length>=1){
                //After JSON
                NSString *finishedString = [NSString stringWithFormat:@"%@}",
                    string];
                NSData *tempData = [finishedString dataUsingEncoding:
                    NSUTF8StringEncoding];
                NSDictionary *temp = [NSJSONSerialization JSONObjectWithData:
                    tempData options:kNilOptions error:&parseError];
                NSDictionary *results = [self nullFreeDictionaryWithDictionary:
                    temp];
                [array addObject:[results objectForKey:@"imageLink"]];
                [array addObject:[results objectForKey:@"imageName"]];

                arraysReady = YES;
                if (parseError) {
                    NSLog(@"Parse Error for Banner! %@", parseError);
                    return;
                }
            }
        }

    }];

    //NSLog(@"Finding array:%@", array);
}

-(void)fillResults{

    //Checks to make sure array has data within it; this effectively stops null
        pointer errors.
    if([array count] >= 1){

        [timerViews invalidate];
        [timerArray invalidate];
        [activityIndicator stopAnimating];
        timerArray = nil;
        timerViews = nil;

    //NSLog(@"%@", array);
```

```
        //Implementation of the banner view.
    [scrollView setContentSize:CGSizeMake(320*3, 104)];
    for (int i = 0; i <6; i+=2) {
        NSString *stringLink = [array objectAtIndex:i];
            UIImage *image = [UIImage imageWithData:[NSData
                dataWithContentsOfURL:[NSURL URLWithString:stringLink]]];
            UIImage *tempImage = nil;
            CGSize targetSize = CGSizeMake(320,104);
            UIGraphicsBeginImageContext(targetSize);

            CGRect thumbnailRect = CGRectMake(0, 0, 0, 0);
            thumbnailRect.origin = CGPointMake(0.0,0.0);
            thumbnailRect.size.width  = targetSize.width;
            thumbnailRect.size.height = targetSize.height;

            [image drawInRect:thumbnailRect];

            tempImage = UIGraphicsGetImageFromCurrentImageContext();

            UIGraphicsEndImageContext();
            image = tempImage;

        UIView *view = [[UIView alloc]initWithFrame:CGRectMake(320*(i/2),0, 320,
            104)];
        [view setBackgroundColor:[UIColor colorWithPatternImage:image]];
        [view.backgroundColor colorWithAlphaComponent:0.3f];

        [scrollView addSubview:view];

    }
    [scrollView setPagingEnabled:YES];
    [scrollView setShowsHorizontalScrollIndicator:NO];
    [scrollView setShowsVerticalScrollIndicator:NO];
    }
}

//Automatic method to change the page of the banner regularly. Called by a
    timer(found at beginning of viewDidLoad) or by the user swipe.
-(void)flipPage{
    if(arraysReady){
    if(pageControl.currentPage !=2)
        pageControl.currentPage++;
    else
        pageControl.currentPage = 0;
    [self changePage];
    }
}

//Used to update the pageControl whenever the scrollView is changed.
- (void)scrollViewDidScroll:(UIScrollView *)sender {
    // Update the page when more than 50% of the previous/next page is visible
    pageControlBeingUsed = YES;
    CGFloat pageWidth = scrollView.frame.size.width;
    int page = floor((scrollView.contentOffset.x - pageWidth / 2) / pageWidth) +
        1;
    pageControl.currentPage = page;
    [annNameLabel setText:[array objectAtIndex:2*pageControl.currentPage + 1]];
}

//Actually changes the scrollView page.
- (IBAction)changePage {
    // update the scroll view to the appropriate page
    pageControlBeingUsed = YES;
    [annNameLabel setText:[array objectAtIndex:2*pageControl.currentPage + 1]];
    CGRect frame;
    frame.origin.x = scrollView.frame.size.width * pageControl.currentPage;
    frame.origin.y = 0;
    frame.size = self.scrollView.frame.size;
    [scrollView scrollRectToVisible:frame animated:YES];
}


- (void)scrollViewWillBeginDragging:(UIScrollView *)scrollView {
    pageControlBeingUsed = NO;
}

- (void)scrollViewDidEndDecelerating:(UIScrollView *)scrollView {
    pageControlBeingUsed = NO;
}


//Animations called during beginning of screen open; aesthetic appeal.
-(void)addAnimations{



    UIGravityBehavior *gravityBehavior = [[UIGravityBehavior alloc]
        initWithItems:@[self.sofoLogo]];
    [self.animator addBehavior:gravityBehavior];



UICollisionBehavior *collisionBehavior = [[UICollisionBehavior alloc]
    initWithItems:@[self.sofoLogo]];
    [collisionBehavior addBoundaryWithIdentifier:@"Center" fromPoint:CGPointMake
        (0, 120) toPoint:CGPointMake(320, 120)];
    [self.animator addBehavior:collisionBehavior];

    UISnapBehavior *snapBehaviorF = [[UISnapBehavior alloc] initWithItem:
        facebookButton snapToPoint:CGPointMake(46,510)];
    UISnapBehavior *snapBehaviorT = [[UISnapBehavior alloc] initWithItem:
        twitterButton snapToPoint:CGPointMake(160,510)];
    UISnapBehavior *snapBehaviorI = [[UISnapBehavior alloc] initWithItem:
        instagramButton snapToPoint:CGPointMake(261, 510)];
    [self.animator addBehavior:snapBehaviorF];
    [self.animator addBehavior:snapBehaviorT];
    [self.animator addBehavior:snapBehaviorI];



}

//Method used for access to social Media(buttons at bottom of page.
-(IBAction)goToSocialMedia:(UIButton *)sender{
    NSURL *facebookURL = [NSURL URLWithString:@"https://www.facebook.com/pages/
        South-Forsyth-High-School/104017019634029"];
    NSURL *twitterURL = [NSURL URLWithString:@"https://twitter.com/
        SouthForsythHS"];
    NSURL *instagramURL = [NSURL URLWithString:@"https://twitter.com/sfhsfbla"];
    if (sender.tag == 0) {
        [[UIApplication sharedApplication] openURL:facebookURL];
    }
    else if(sender.tag == 1){
        [[UIApplication sharedApplication]openURL:twitterURL];
    }
    else{
        [[UIApplication sharedApplication] openURL:instagramURL];
    }

}

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

//Used to "safeguard" dictionaries from containing a null string.(There is a
    difference between a "null"(no memory allocated) string and a "nil"(empty
    string "") string).
- (NSDictionary *)nullFreeDictionaryWithDictionary:(NSDictionary *)dictionary
{
    NSMutableDictionary *replaced = [NSMutableDictionary
        dictionaryWithDictionary:dictionary];
    // Iterate through each key-object pair.
    [dictionary enumerateKeysAndObjectsUsingBlock:^(id key, id object, BOOL *stop
        ) {
        // If object is a dictionary, recursively remove NSNull from dictionary.
        if ([object isKindOfClass:[NSDictionary class]]) {
            NSDictionary *innerDict = object;
            replaced[key] = [self nullFreeDictionaryWithDictionary:innerDict];
        }
        // If object is an array, enumerate through array.
        else if ([object isKindOfClass:[NSArray class]]) {
            NSMutableArray *nullFreeRecords = [NSMutableArray array];
            for (id record in object) {
                // If object is a dictionary, recursively remove NSNull from
                    dictionary.
                if ([record isKindOfClass:[NSDictionary class]]) {
                    NSDictionary *nullFreeRecord = [self
                        nullFreeDictionaryWithDictionary:record];
                    [nullFreeRecords addObject:nullFreeRecord];
                }
                else {
                    if (object == [NSNull null]) {
                        [nullFreeRecords addObject:@""];
                    }
                    else {
                        [nullFreeRecords addObject:record];
                    }
                }
            }
            replaced[key] = nullFreeRecords;
        }
        else {
            // Replace [NSNull null] with nil string "" to avoid having to
                perform null comparisons while parsing.
            if (object == [NSNull null]) {
                replaced[key] = @"";
            }
        }
    }];

    return [NSDictionary dictionaryWithDictionary:replaced];
}


@end
```