Term Project CS698

iRun (Walkout Monitor)

By Kijit Desai (kd227) and Nanda Kishore(ny44)

Introduction

- Our application lets the users monitor their current and average speed while running.
- It allows the users to monitor their past workouts as well.
- Another feature to this app is that the user can calculate lifts, i.e. the number of time the user lifts the phone up and down

Project Workflow

- Frameworks
- Individual fragments
- Data Storage
- UI
- Getting it all together

Frameworks

- CoreLocation framework
- Mapkit framework
- CoreData framework
- UIkit framework (accelerometer Delegate)

Calculating Running speed

We made use of the GPS for calculating the running speed of the users. This gave us the advantage of having the exact precise location of the user which helped us not only measure speed but also keep a track of the user. Below is a snippet of the code used for the same.

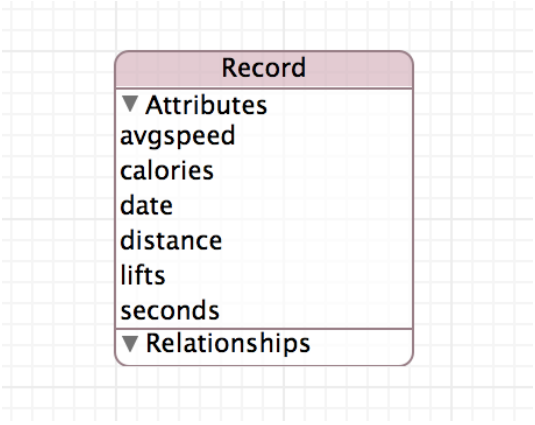
- iRun › Project › RunViewController.m › -averageSpeed

```
@interface RunViewController ()

@property (strong,nonatomic) NSMutableArray *averageSpeed ;
@property (readwrite,nonatomic) double *average ;
@property float calories;
@end

@implementation RunViewController

-(NSMutableArray *)averageSpeed
{...}


- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{...}

- (void)viewDidLoad
{...}
- (IBAction)runButton:(id)sender
{



    NSString *title=self.runButton.titleLabel.text;
    if ([title isEqualToString:@"Run !"])
    {
        //For Timer
        self.secondsTimer = [NSTimer
                             scheduledTimerWithTimeInterval:1.0
                             target:self
                             selector:@selector(timerFireMethod:)
                             userInfo:nil
                             repeats:YES];
        self.seconds = 0;
        self.minutes = 0;
        self.hours = 0;
        _startDate=[NSDate date];
        self.locationManager = [[CLLocationManager alloc] init];
        _locationManager.delegate= self;
        _locationManager.desiredAccuracy = kCLLocationAccuracyBest;
        [_locationManager startUpdatingLocation];
        [self.runButton setTitle:@"Stop" forState:UIControlStateNormal];
    }else
    {
        [_locationManager stopUpdatingLocation];
        [_secondsTimer invalidate];
        [self.runButton setTitle:@"Run !" forState:UIControlStateNormal];
        _save.hidden=false;
```

CoreData:

Our project required us to store the information and have the option for user for retrieval. So we had to store the data that is non-volatile and accessible until the user had the application installed in his phone. So , we went ahead with implementing the core data. We created entity named 'Record' that has all the attributes like the date of the walkout, average speed, calories lost, date of the workout, lifts he has done, time the walkout lasted. All these values are stored and retrieved when the user want to compare them with his past walkouts.

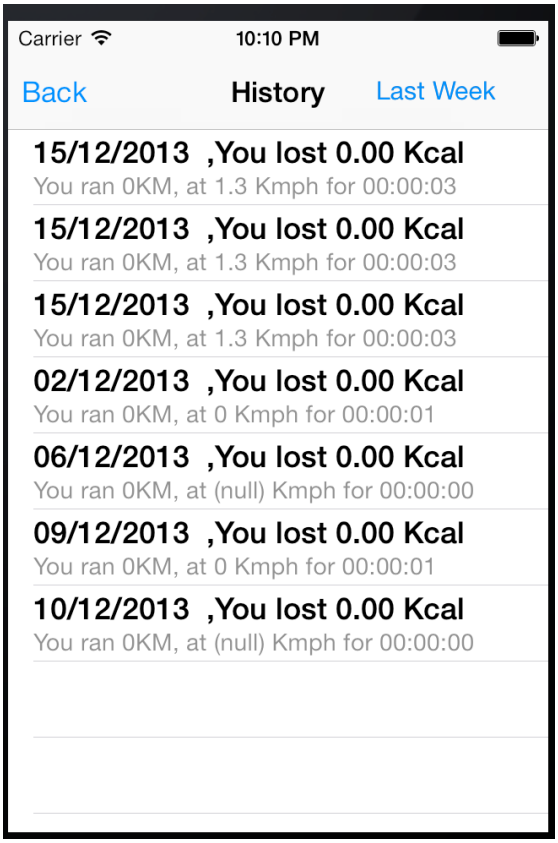
All the implementation for the core data is present in the app delegate. All the three `managedObjectContext`,`managedObjectModel`,`persistentStoreCoordinator` Methods exist to provide the core data support. Also the methods `getAllPhoneBookRecords`,`getModifiedRecords` are present to provide the Fetching capability from the data store. Here is the screenshot of the data store from our project.



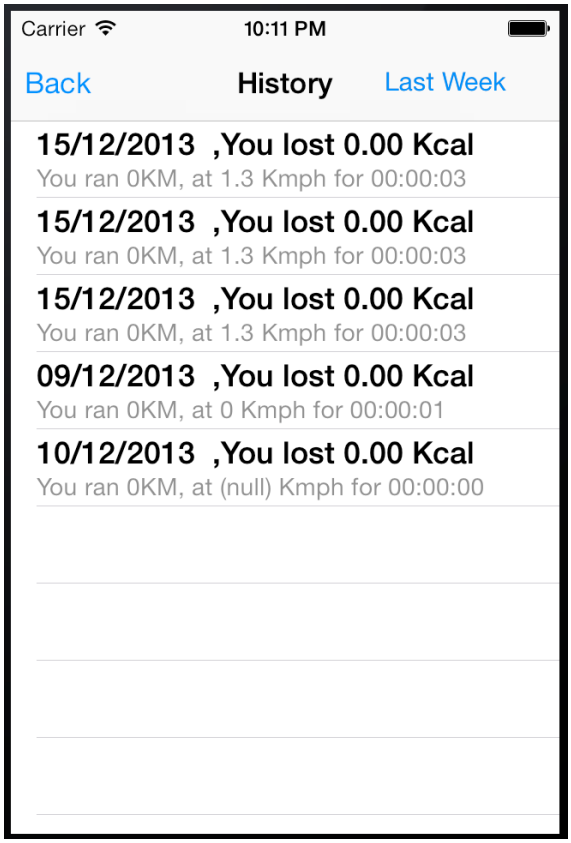
Implementing the Lifts:

For calculating the lifts we have used the accelerometer sensor with the high pass filter. We used the high pass filter because it was found to be giving better results as the linear acceleration component is filtered from the results. We had finally achieved the right values for the axis to detect the lifts of the user properly.

Below are the screenshots of the application where the user has an option to view and compare the data from last week in contrast to all the data recorded by the application.



All the Data



Data From past 7 days

Below screenshot shows how we calculated the lifts of the user.

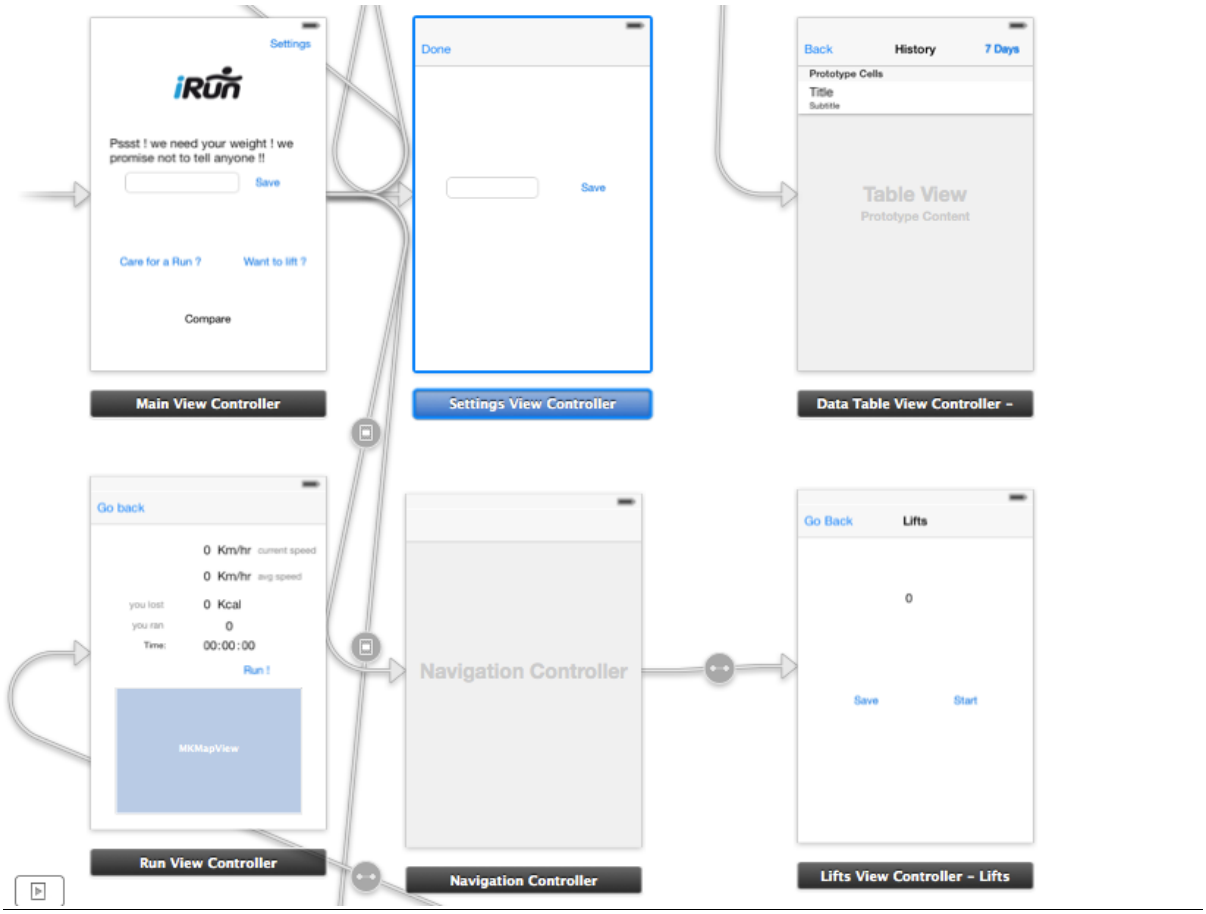
```
-(void)accelerometer:(UIAccelerometer *)accelerometer didAccelerate:(UIAcceleration *)acceleration{
    float z =0,y=0;
    float x=0;
    // High pass filter
    accelerationArray[X_DIR] = acceleration.x - ((acceleration.x * HIGHPASS_FILTER) + (accelerationArray[X_DIR] * (1.0 -
        HIGHPASS_FILTER)));
    accelerationArray[Y_DIR] = acceleration.y - ((acceleration.y * HIGHPASS_FILTER) + (accelerationArray[Y_DIR] * (1.0 -
        HIGHPASS_FILTER)));
    accelerationArray[Z_DIR] = acceleration.z - ((acceleration.z * HIGHPASS_FILTER) + (accelerationArray[Z_DIR] * (1.0 -
        HIGHPASS_FILTER)));

    // Was the device moved away or toward you...
    if (z - accelZ > 0.5 || z -accelZ < -0.5)
    {

    }
    // Was the device moved up or down
    else if (y - accelerationArray[Y_DIR] > 0.9 || y - accelerationArray[Y_DIR] < -0.9)
    {

    }

    // Was the device moved left or right
    else if (x - accelerationArray[X_DIR] > 0.2)// || x - accelerationArray[X_DIR] < -0.2)
    {
        lifts=lifts+1;

    }
    liftsLabel.text = [NSString stringWithFormat:@"%d",lifts];
}
```

Table View To display the data:

We have used the table view controller to display the data to the user that he saved. For this we used the NSFetchRequest method to get the contents form the data store. The user also has an option to view the data from past seven days. We have implemented that by using the NSDate. The screenshot below illustrates that.

```
NSDate *today = [NSDate date];
NSDateFormatter *dateFormat = [[NSDateFormatter alloc] init];
[dateFormat setDateFormat:@"dd/MM/yyyy"];
NSString *dateString1 = [dateFormat stringFromDate:today];

NSDateComponents *dateComponents = [[NSDateComponents alloc] init];
dateComponents.day = -7;
NSCalendar *calendar = [NSCalendar currentCalendar];
NSDate *previousDate = [calendar dateByAddingComponents:dateComponents
                                                 toDate:today
                                                options:0];


NSString *laterDate = [dateFormat stringFromDate:previousDate];


NSPredicate *predicate = [NSPredicate predicateWithFormat:@"((date > %@) AND (date <= %@))",laterDate,dateString1];


NSFetchRequest *fetchRequest = [[NSFetchRequest alloc] init];
fetchRequest.sortDescriptors=@[[NSSortDescriptor sortDescriptorWithKey:@"seconds" ascending:NO]];
[fetchRequest setPredicate:predicate];
//fetchRequest.fetchLimit = 2;

NSEntityDescription *entity = [NSEntityDescription entityForName:@"Record"
                                           inManagedObjectContext:self.managedObjectContext];
[fetchRequest setEntity:entity];
NSError* error;


NSArray *fetchedRecords = [self.managedObjectContext executeFetchRequest:fetchRequest  error:&error];

return fetchedRecords;
```

User Interface (Storyboard)



Storyboard 1