AN IPHONE APP OF KONZA PRAIRIE LTER

By

LEELA ANUSHA NUKALA

B.Tech., Jawaharlal Nehru Technological University, 2010

A REPORT

Submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Computing and Information Sciences

College of Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

2012

Approved by:

Major Professor

Dr. Daniel Andresen

ABSTRACT

The objective of this project is to develop an iPhone application for the Konza Prairie LTER and play a significant role in the development of their website. The data of the Konza Prairie LTER is vast and includes the spatial datasets, weather reports, text archives, information about the flora and fauna at Konza Prairie Natural Area, integrated project areas, Dataset Codes, LTER Core Areas, Related projects, Konza Documents and Permits, and Publications.

The module which was developed by me provides users information like publications, personnel information and datasets. The titles of the publications along with their PIs will be retrieved from the database and formatted onto the webpage. The 'Personnel' page has links directing to Primary Contacts, Faculty and Staff and Graduate Students. The attractive feature of this personnel page is that each personnel in the list will have a '+' link which enables the user more information about that particular personnel like their field of interests, mailing address and a link to view their profile. As the website already has all the required data and information in detail, the mobile app only briefs about them. People who are working on-site can make of this app efficiently as its features includes list of upcoming events, map of Konza with GPS feature, Post Card feature, Primary personnel to be contacted and other features.

The website is developed using Visual Studio 2010 and SQL Server 2008 database. The iPhone app enhanced my knowledge and provided me with real-time exposure to iPhone SDK tools, X-code, Interface builder and its development environment.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

My special thanks to my major professor and advisor Dr. Daniel Andresen for giving me timely advice, encouragement, and guidance throughout the project, my course work and my Masters.

I would also like to thank Dr. Mitchell Neilsen and Dr. John Blair for graciously accepting to serve on my committee.

Finally, I would like to thank the administrative and technical support staff of the department of Computer and Information Sciences, my family and friends for their support throughout my graduate study.

DEDICATION

I would like to dedicate this project to my father N. Venkateswara Rao, my mother Madhavi, my uncle Narasimha Rao and my aunt Mahalakshmi.

Chapter 1 - INTRODUCTION

The internet has rapidly developed over the past years and everyone is jumping on to the Internet wagon today. It is used for banking, news, marketing through websites, social networking and communication and has become a major part of life. Smartphones, on the other hand, refers to mobile phones with advance features such as fully featured browsers, GPS receivers, multimedia capabilities and especially always-on connectivity to Internet. The manufacturers and vendors of the smartphones and mobile operating systems reported that record numbers of units were sold as these smartphones have become extremely popular.

1.1. Objective:

The main objective of this project is to collect data from Konza Prairie Long-Term Ecological Research (LTER) so that it will be useful in developing an iPhone app and a significant module of their website. As the older website of Konza had large amounts of stagnant data, the need of Konza for a new and better website has increased. The new version should be free of unwanted data which caused overload on the old website. Konza Prairie LTER program is one of the first six site-based LTER programs funded by the National Science Foundation (NSF). Its research comprises studies across multiple ecological levels (organismic, population, community and ecosystem) and spatial (plot-level, watersheds, regional landscapes) and temporal (days to decades) scales. The objective is to develop a website which contains precise data and an iPhone app with a user-friendly interface. All the research programs, personnel information and dataset details are categorized and arranged into separate catalogs and framed them as dropdown menus in the website where as each catalog is a separate view controller in the iPhone app. The new website is now available at www.konza.ksu.edu.

1.2. Intended Users:

Students: It would help students to know about the on-going research at the Konza Prairie Biological Station and learn about the findings and e-mail faculty investigators if they have any questions regarding the published articles. As the website has the faculty information, interests and contact details it will be easy for students to choose faculty and work with them.

Faculty: This website helps the faculty to view the published articles, datasets and other information that is required for their research. They can also download datasets and text archives from the website.

On-site researchers: On-site workers can know the upcoming events, meetings and conferences that happen at the Konza Prairie. The site also has a weather column using which the researchers can know about the weather and whether or not they can collect their samples.

1.3. Motivation:

The motivation in developing a mobile app is that today's smartphones have grown rapidly and have their own in-built RAM, processing unit and Internet connectivity making them more powerful than the desktops that were used 10 years before. These advanced features of the newly released mobile OSs made the users think of their smartphones as computers rather than normal mobile phones.

As it is always easier and much simpler to carry a smartphone than any other mobile device, the need for a mobile app of the Konza Prairie has increased. The reason for selecting iOS is because it is known for its standards and is registered as one of the best-selling

smartphone platforms worldwide. This not only provides the system with low-level services but also enforces a high level of security and manages memory as well.

The “Konza Prairie Mobile App” is a beta-release of the mobile application developed on iOS for the Konza. This is like a mobile version of the website that is available at www.konza.ksu.edu. The website includes the research work that is being done at the Konza Prairie, the educational services it offers, images of the tall grass and other species of plants that are found at the Konza Prairie Biological Station, the primary personnel that can be contacted, other personnel like the faculty, staff, graduate students and post-docs, their field of interests, and the data.

The data of the Konza Prairie LTER is vast and includes the spatial datasets, weather reports, text archives, information about the flora and fauna at Konza Prairie Biological Station, Eco trends, integrated project areas, Dataset Codes, LTER Core Areas, Related projects, Konza Documents and Permits, and Publications. Most of the above mentioned data contains previews of the trends and the datasets during selected intervals. Including all this information to the mobile app is not feasible as the memory in any mobile device will be limited. As the website already has this data in detail, the mobile app only briefs about them.

This app is targeted for a specific group of researchers and other users who work at the Konza Prairie Biological Station (KPBS) but can be used by anyone who is interested to know about the Konza Prairie. People who are working on-site can make use of this app efficiently as we added a new feature that displays the map of Konza Prairie Natural Area and also receives GPS signals which enables the user to find his way through the tall grasslands.

Chapter 2 - REQUIREMENT ANALYSIS

2.1. Requirements Gathering:

The aim of this project is to build a mobile app and a significant module of the website for Konza Prairie. The website is developed using Visual Studio 2010 with SQL Server 2008 as its database where information about each the datasets, publications, PIs and its research are stored. The front end is developed in ASPX pages and the business is written in C#. On the other hand, for the iPhone application, is done by using X-code, Interface Builder and Objective C. Object Oriented Programing is more flexible and pretty useful in developing programs and solving problems. Though the requirements of this app are gathered keeping on-site workers in mind, it should also be used by wide range of audience including many researchers and students from different universities.

In any mobile app, GUI plays a crucial role as the users don't have time to read every word that is displayed. Hence pictorial representation or highlighting of headings and sub-headings are used. The hardware and software requirements along with the installation instructions are given below.

2.2. Requirements Specification for iPhone app:

2.2.1. Hardware Requirements for User:

- *Device*: iPhone / iPod
- Minimum Space Required: 10 MB

2.2.2. Software Requirements for User:

- *Operating System:* iOS 4 or higher

2.2.3. Hardware Requirements for Developer:

- *Device*: iPhone / iPod/ iPhone Simulator
- Minimum Space Required: 40 MB
- Connecting Cable: if testing is done on iPhone/iPod

2.2.2. Software Requirements for Developer:

- *Operating System:* OS X
- *Tools:* X-code, Interface Builder [3]
- *Language:* Objective C
- *Debugger:* iPhone Simulator / iPhone

This app is developed in X-code, an IDE to develop an iPhone application. Unlike Android, a developer has to register with Apple to publish his app or even test the app on an iPhone device.

2.3. Requirements Specification for website:

2.3.1. Hardware Requirements for User:

- *RAM: 512 MB*
- *Processor: Core 2 Duo/i3/i5/i7*

2.2.2. Software Requirements for User:

- *Operating System:* Windows 2000 or higher

2.3.3. Hardware Requirements for Developer:

- *RAM: 512 MB*
- *Processor: Core 2 Duo/i3/i5/i7*
- *Hard Disk space: 10 GB*

2.2.2. Software Requirements for Developer:

- *Operating System:* Windows 2000 or higher
- *Browser:* Internet Explorer 8 or higher
- *Tools:* Visual Studio 2010
- *Language:* C#.NET
- *Database: SQL Server 2008*

Chapter 3 - SYSTEM DESIGN AND ARCHITECTURE

Considering the requirements that were specified, the system's layouts have been designed. This chapter describes the system architecture hierarchy, use case diagram and class diagram of the application.

3.1. Website Architecture:

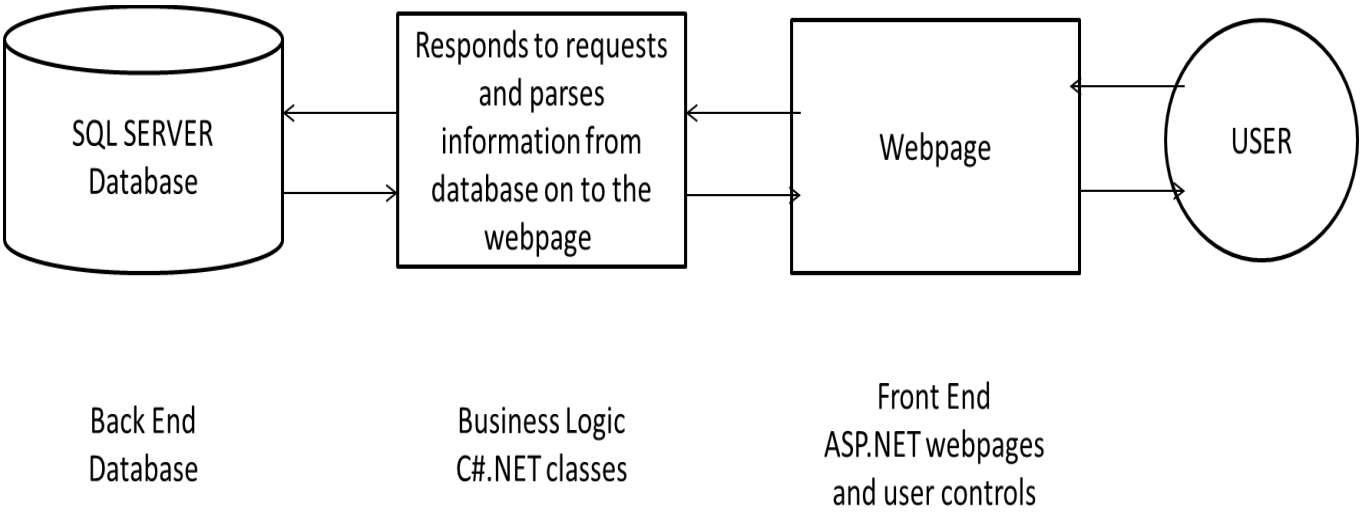

Figure 3.1 – Website Architecture

The website development was mainly focused on having clear-cut information and user-friendly interface. User can easily navigate from one page to another without much effort or providing input to the webpage. The user can browse through all the available datasets and categories and can download data in the form of text files. The website has a search box in the Site Master page which allows the user to search for any information, personnel and data. This data is retrieved from the database as queries and then displayed to the user in proper format.

3.1. IPhone Application Architecture:

The architecture of iOS can be divided into six layers as follows:

1. **The Application:** This is the iPhone application that is currently in use. This can only be purchased from the app store and runs completely within the user-space environment that is set up by the operating system of iPhone.

2. **Frameworks/API:** Many of these frameworks are written in Objective C and hence they reside on top of Objective-C layer. These APIs have dynamic links, which can be used at runtime.

3. **Objective C:** This layer is comprised of libraries of both Objective C and C. Here C libraries act as the base for the Objective C libraries and set up environment for Objective C libraries.

4. **iOS:** iPhone OS acts as an interface between the user-space in the application layer and the hardware. It has the kernel, services and drivers of iPhone OS.

5. **Processor:** It refers to the instruction and interrupt set descriptions that are set up by iOS during boot and driver installation.

6. **Hardware:** Its quite obvious of what is present in this layer. Anything and everything that the user can see or feel falls under this layer.

The architectural diagram of the iPhone operating system can be represented pictorially using the following diagram. It clearly looks like a stack of layers. Each layer has its own importance and should do its intended purpose for the application to work correctly.

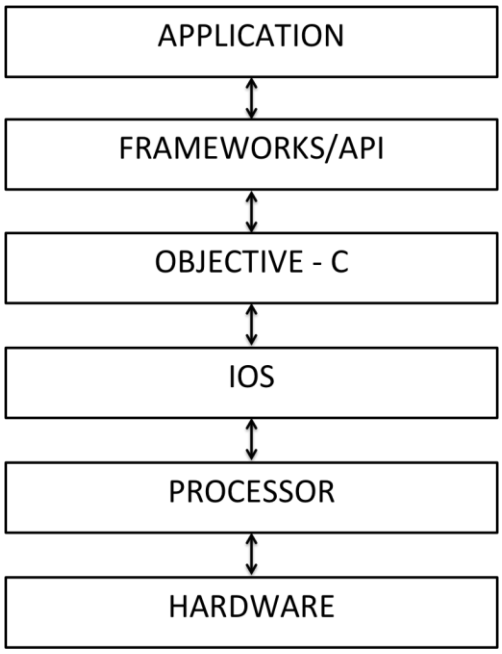

Figure 3.2 – iPhone Architecture

Though both iPhone architecture and the website architecture are 3-tier architecture, the architecture of iOS differs from that of website as it uses frameworks/API and objective C in the business logic.

As the user starts the app, the main page will be launched. It displays the options like information about the Konza Prairie (ABOUT), the primary personnel to contact (PERSONNEL), the research work at Konza LTER (RESEARCH), the Education (EDUCATION), the Post card feature for people who visit Konza to take pictures with their customized overlay, Events and GPS Feature (MAP). The user can choose a desired option. Depending on what option the user chose, the next layout will be displayed. For instance, if he chooses “ABOUT”, the screen will be redirected to the ABOUT page where the user can see details about Konza. The user can choose one of these options to view precise details about the Konza. Similarly, the personnel page displays four primary personnel with their phone numbers

and e-mail ids. If tapped on the name of any of these personnel, detailed information about their office address, publications and interests will be displayed. The Research illustrates their research program and the Goals of Konza Prairie LTER

3.2. System Design

The design of the system is illustrated by using UML design diagrams. The class diagram is drawn with the help of X-code itself and can be saved to disc. The following diagrams are used to demonstrate this app:

- Class Diagram
- Use Case Diagram

3.2.1. Class Diagram

The following class diagram depicts the classes along with the properties and operations in each class. In the below figure, every class inherits the “UIViewController” class along with its properties and operations.

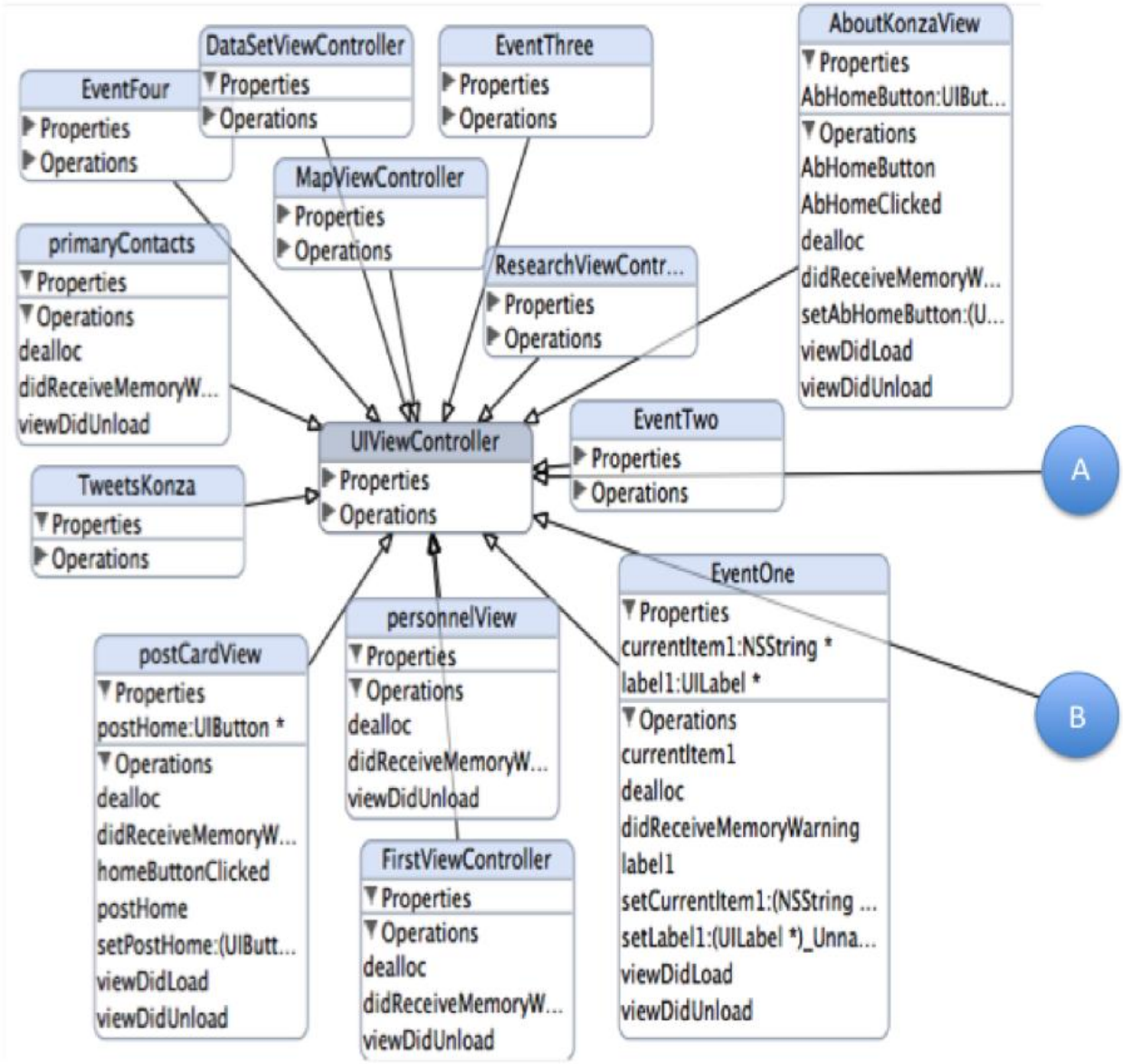

Figure 3.2.1 - Class Diagram (a)

Each class that is declared in the project is in this class diagram where the inheritance can be observed. Most of the classes inherit the UIViewController class, as it is one of the most commonly used classes in any view controller. Each class has its own properties i.e., variables and its operations i.e., methods.

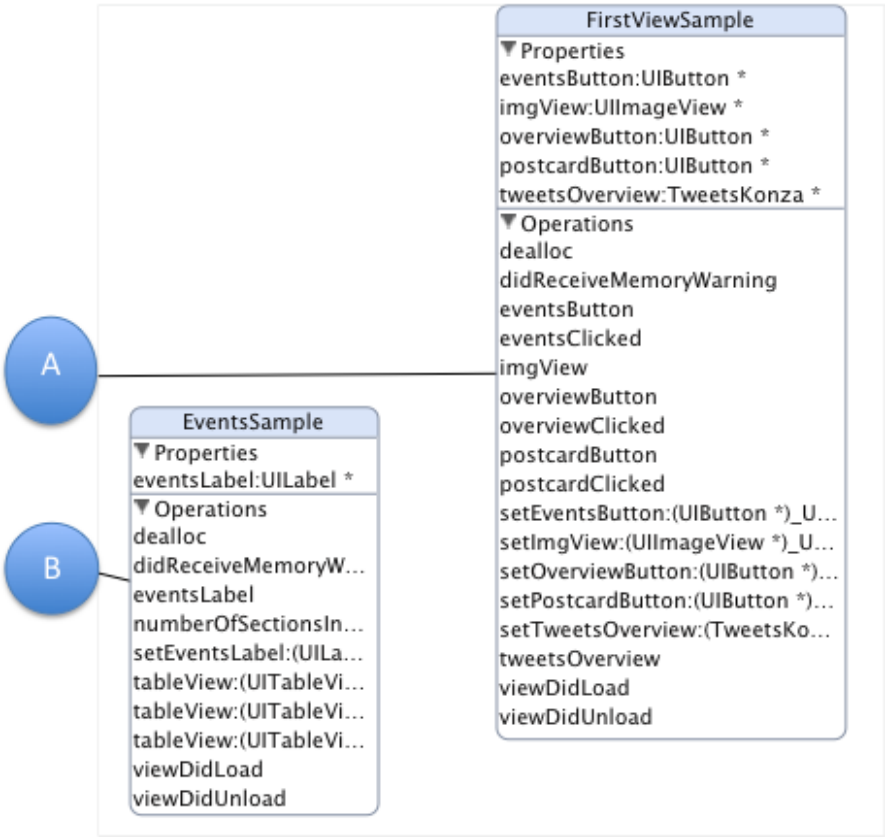

Figure 3.2.2 - Class Diagram (b)

This is the continuation of the above class diagram. These two classes also inherit the "UIViewController" class. For instance, the FirstViewSample class has the properties like the buttons and image-view and its operations are actions that should be performed when each of these buttons are clicked.

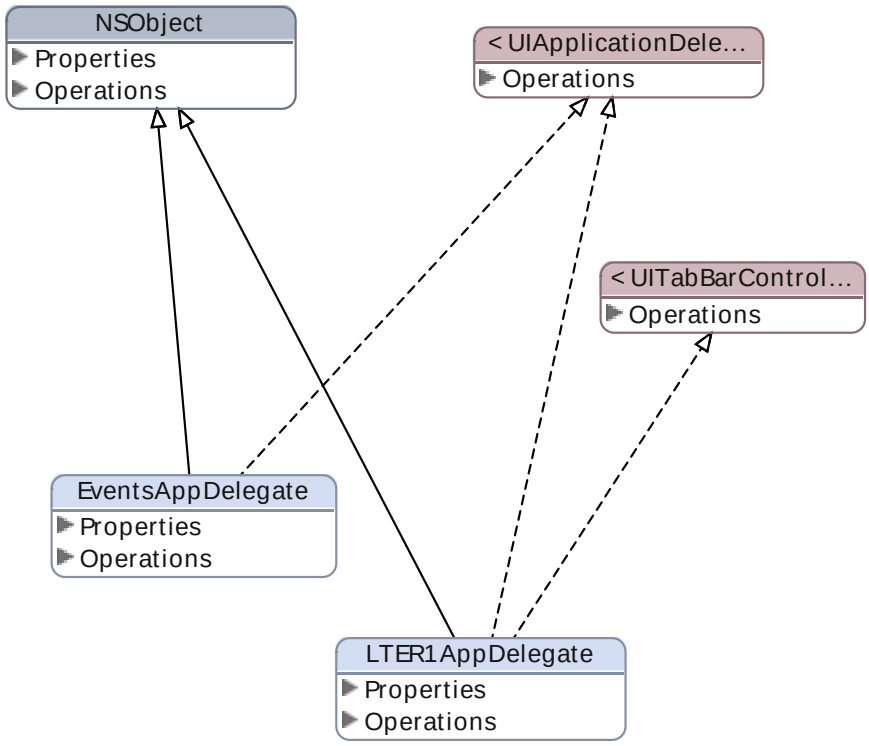

Figure 3.2.3 - Class Diagram (c)

This section of the class diagram indicates how the NS-Object class and the UI Tab bar class that inherits the APP Delegate class. The app delegate class contains all the required information about the IBOutlets and IBActions that are present in the project. The UI tab bar class deals with the tab bar items that are present in the Main Window.

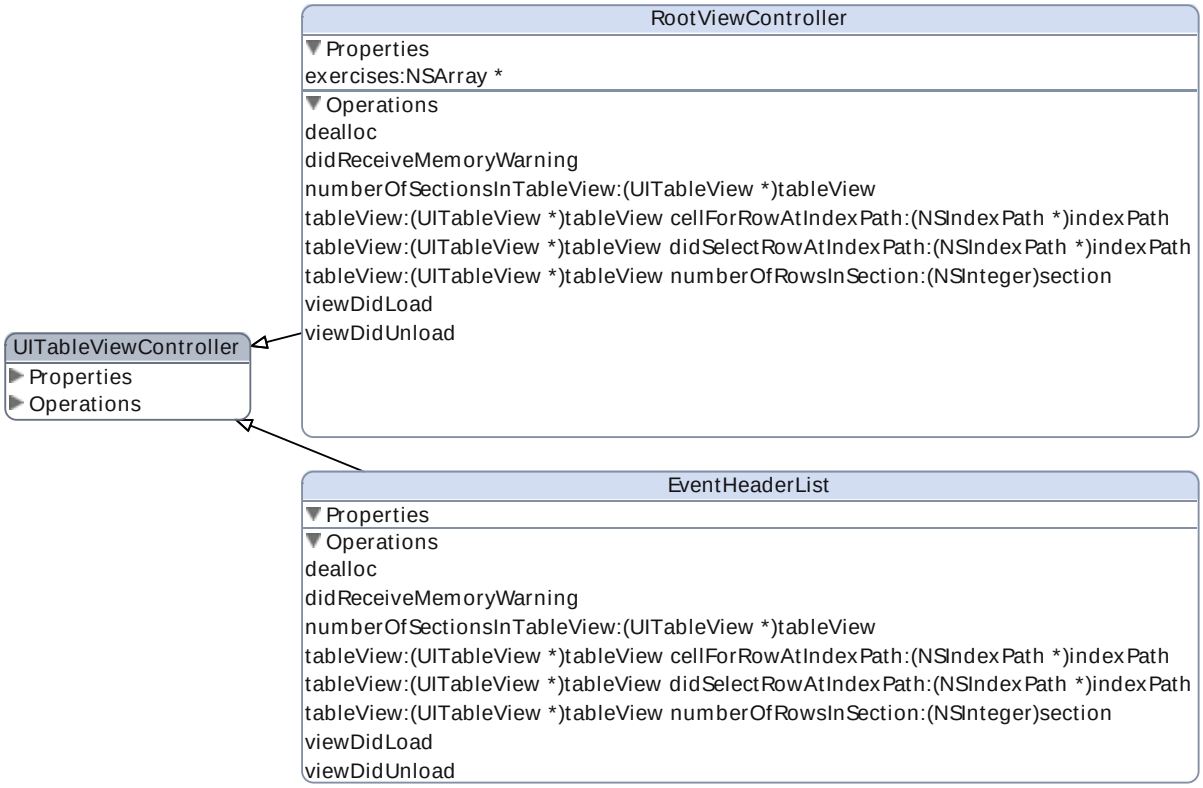

Figure 3.2.4 - Class Diagram (d)

This section of the class diagram indicates there are two classes in the application that inherits the properties and operations of the UI Table View Controller class. This is used there should a list displayed. The list in this application is used for displaying the list of upcoming events, conferences and seminars. Each table-view view controller has few mandatory operations that along with the normal operations that needs to be implemented in the implementation file.

The first method is “Number of Sections in Table View” operation. This is required to tell the table view controller about the number of sections in the table view. This method returns the total number of sections that should be there in the table view that is being displayed.

```
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
    return 1;
}
```

The second method is “Number of Rows in Each Section” operation. This is required to tell the table view controller about the number of number of rows that should be present in each section of the table view. The number of rows in each section need not be same and may vary.

```
- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section {
    return exercises.count;
}
```

The last method is “Cell for Row at Index Path” operation. This is required to tell the table view controller about the contents of each row in the table view. This method returns the cell value that is to be displayed in the table view. Here care is taken such that only the cells that are visible to the user occupies the memory and all other cells’ memory will be released thus managing memory efficiently.

```
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath
*)indexPath {

    static NSString *CellIdentifier = @"Cell";

    UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:CellIdentifier];
    if (cell == nil) {
        cell = [[[UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefault
reuseIdentifier:CellIdentifier] autorelease];
    }

        // Configure the cell.
        //cell.textLabel.text = @"Some Value";

        cell.textLabel.text = [exercises objectAtIndex:indexPath.row];
```

```
return cell;
  }
```

The classes that are associated with the “Main Window” are the options that the “Home” page offers to the users. Each class has its own variables that are used inside the class. Most of these variables are named after the unique id of the user controls like the text-views, buttons, list views and gallery. There is no overriding of these variables. Most of these variables implement their own methods where the functionality of what should happen when the user performs an action is implemented.

3.2.2. Use Case Diagram

It shows a graphical overview of the system functionality in terms of an actor and a set of use cases where the actor is the user and the set of use cases represents the actions that a user can perform.

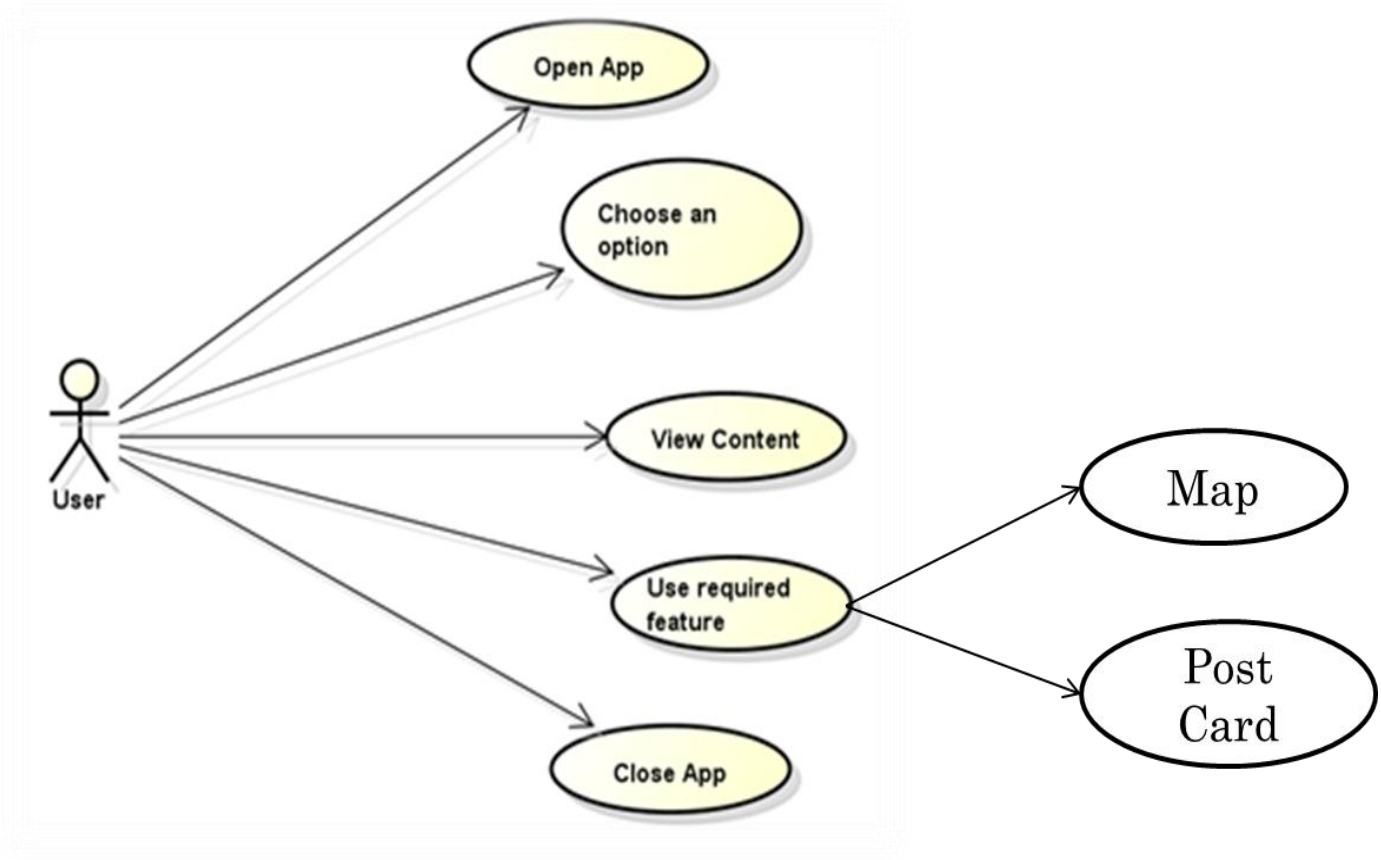

Figure 3.3 - Use case Diagram

The actions that can be performed by a user are:

1. *Open App:* The user can open the app or the browser.

2. *Choose an option:* Out of many options he has on the main page, user can choose one option.

3. *View Content:* After choosing his desired option, the user views the content of the launched page. There are many screens that the user can wish to view. Once he is

on the required screen, he can either view the content or proceed to use the required feature.

4. *Use required feature:* In this phase, if the user is using iPhone app, the user can either use the GPS feature of the ’Map’ or take a picture using the Post Card feature. If the user chooses the Map feature, then the Google maps will be displayed. If the user is using the website, he can preview the datasets in a tabular form or download the datasets onto his system.

5. *Close App:* Once he finishes using the app or the website, he just closes it.

Chapter 4 - GRAPHICAL USER INTERFACE

The Konza Prairie LTER mobile app is typically targeted for people who work on-site in the Konza Prairie Biological Station so that they can use the GPS feature. Hence the user interface of this app is designed to be very straight forward and simple. The user can just click on the tabs provided to get detailed information. The layouts contain normal buttons and links and are designed in such a way that very little input is prompted from the user.

According to a survey, it is found that the users spend 4.4 seconds for every extra 100 words on the page. Hence, care is taken to be precise in the content and avoid any additional information that is stagnant. Also, taking into account the low patience levels of the users, the text is made to be concise with headings and list-view items wherever possible.

Numbered lists are used for text when necessary instead of paragraphs. Also, the layout of each page is similar to that of main website with an image banner and colors used for text and background so that the user feels familiar with the app.

A detailed description of the graphical user interface is given in the screen-shots chapter of this document. The differences and similarities of the Home pages of the website and mobile app can be described as follows:

Figure 4.1 – Home page of the Website

The home page of the website contains a series of drop-down menus each of them having information about the Home, Research, Data, Publications, Education and Personnel. The page is static except for the events box which is like a slide show of the list of upcoming events. Similarly, the home page of the app contains buttons for these drop-down menus which when clicked, navigates the user to another view controller or screen where the user can view the content about the research, education, events and personnel pages. The list of events is given in a

tabular view. When the user selects a particular event, the user is navigated to screen which has more information about the event including a link to know more information about the event.

Also, the mobile app has other significant features like Post Card and Map where the user can take pictures with Konza logo as border which can be used as a post card by the user. User can use the GPS feature when he is on the Konza Trails or any other part of the site. Also, the user can directly call or e-mail the personnel by just clicking on the phone number or e-mail address that is displayed the screen.

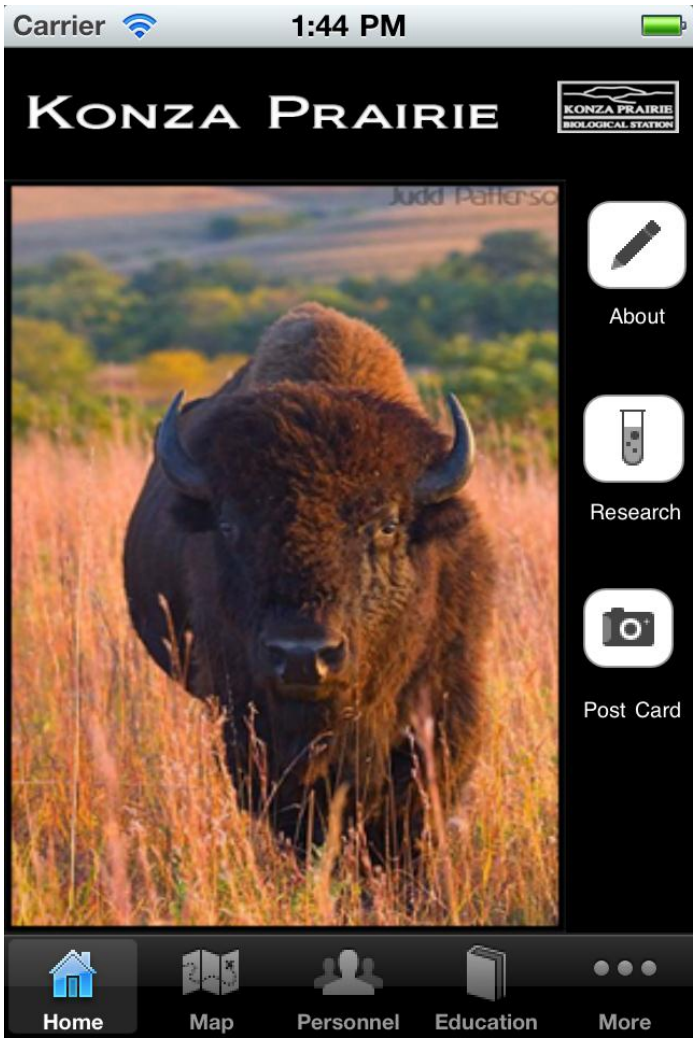

Figure 4.2 – Home page of the iPhone application

Chapter 5 - IOS CODE DESCRIPTION

This chapter summarizes the code description of iPhone app and explains about the development and implementation phases and describes the code of the application.

Each view controller is associated with three files namely a header file, an implementation file and a xib file. Each of these files has their own significance in the development of the iOS application.

5.1. Header File

The header file contains all the declarations. The image-views, buttons and labels are defined as IBOutlets. The methods should also be declared in the header file. The below code is taken from First View Sample View Controller. [2]

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

@interface FirstViewSample : UIViewController
{

        UIImageView *imgView;
        UIButton *eventsButton;
        UIButton *overviewButton;
        UIButton *postcardButton;


}

@property (nonatomic, retain) IBOutlet UIImageView *imgView;
@property (nonatomic, retain) IBOutlet UIButton *eventsButton;
@property (nonatomic, retain) IBOutlet UIButton *overviewButton;
@property (nonatomic, retain) IBOutlet UIButton *postcardButton;

        @end
```

Most of the classes inherit the UIViewController class which has the basic required methods that a view controller needs. Initially each UI Control is declared inside the class. For instance, “UIButton *postcardButton;” says that there is button with name “postcardButton”. Then we create a property for that button as shown above. Also, if we need to implement any methods for these IBOutlets, we have to define them in the header file as IBActions.

5.2. Implementation File

The implementation file is where the actual business logic of the application is present. Each IBOutlet that is created in the header file should be synthesized in this implementation file so that the X-code will implement its getter and setter methods automatically. This can be done as follows:

```
@synthesize postcardButton;
```

Now for this button, once we use the above code, all the in-built methods will be inherited so that we can hook these up to the File’s Owner.

Each implementation file has four methods each of which has its own purpose. These four methods are:

```
- (void)viewDidLoad
{
        [super viewDidLoad];
    [postcardButton addTarget:self action:@selector(postcardClicked)
forControlEvents:UIControlEventTouchUpInside];


}
```

This is the method where we can change the view that is displayed based on logic and conditions. Once the view controller is loaded, if the user clicks on any button or tab bar item, the action that is to be performed will be in this method. Here, when the user clicks the post card button, we need to redirect the user to another screen where he can take pictures. In the above code, for the event "TouchUpInside" i.e., when the user taps a button and lifts his finger away from the screen, an event called "postcardClicked" is called. This method will be as follows:

```
-(void) postcardClicked
{
        NSLog(@"PostCard button Clicked!");

        postCardView *pcb = [[postCardView alloc] initWithNibName:@"postCardView"
bundle:nil];
        [self presentModalViewController:pcb animated:NO];

        [pcb release];
}
```

Here NSLog is used to print statements on the console. A new variable pcb is created and initialized to the "postCardView" view controller and the user will be navigated to that view controller with the help of the variable pcb. But once the user is navigated, we no longer need to store the variable. Hence the memory at which the variable stored is released and thus the memory is well managed.

```
- (void)didReceiveMemoryWarning
 {
        // Releases the view if it doesn't have a superview.
         [super didReceiveMemoryWarning];

        // Release any cached data, images, etc. that aren't in use.
 }
```

This method is used to do any action if there is any memory leak n the application. If the user needs to be notified or any other action should take place once there was a memory warning,

then this method is used.

```
- (void)viewDidUnload
{
        [super viewDidUnload];

    }
```

ViewDidUnload method is used to release any sub-views of the main view that are retained. For instance, releasing the memory of an image view will be done here as follows:

```
self.imgView = nil;
```

```
- (void)dealloc
{
        [super dealloc];
}
```

This method has more significance than other methods because this method is used to release the memory that is attained by all the IBOutlets that are present in the view controller. A dealloc method that releases memory can be written as follows:

```
- (void)dealloc
{
    [imgView release];
    [eventsButton release];
    [overviewButton release];
    [postcardButton release];
    [super dealloc];
}
```

5.3. Xib File

The xib file is the file that is displayed to the user as a view controller. All the IBOutlets present in the view controller are aligned and layout is properly fixed using “Interface Builder”. The Main Window with the tab bar items will look in the interface builder as shown below.

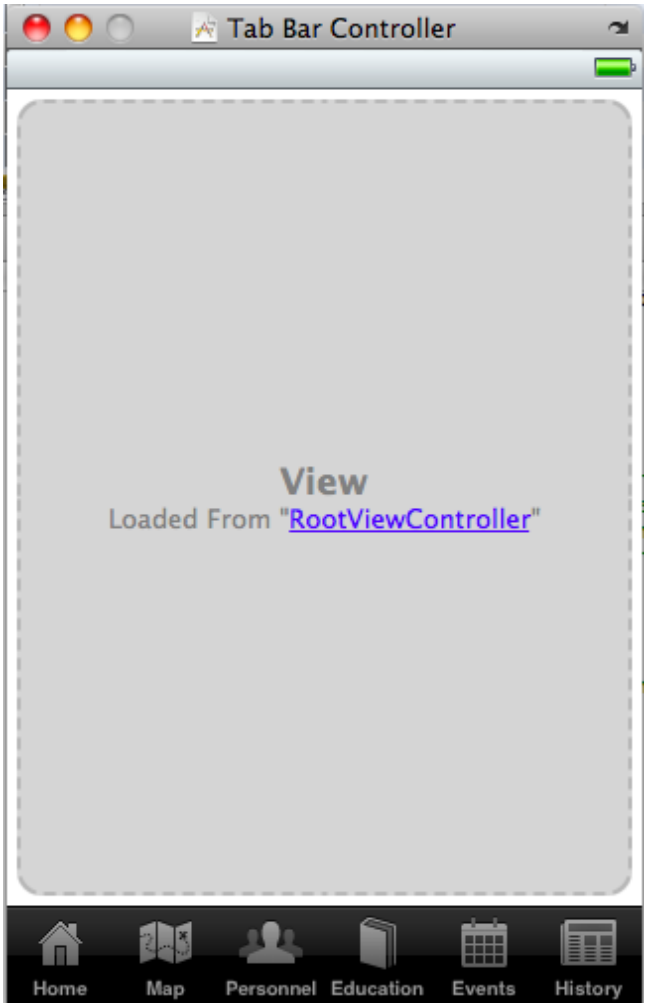

Figure 5.1 – Xib file in Interface Builder

This is how the Main Window looks when seen in an interface builder. The can be changed and, images, buttons, labels and other UI controls can be added to a view controller through this interface builder. It displays which tab bar item loads which view controller when clicked. Each Interface builder has a File's Owner view that can be shown as follows:

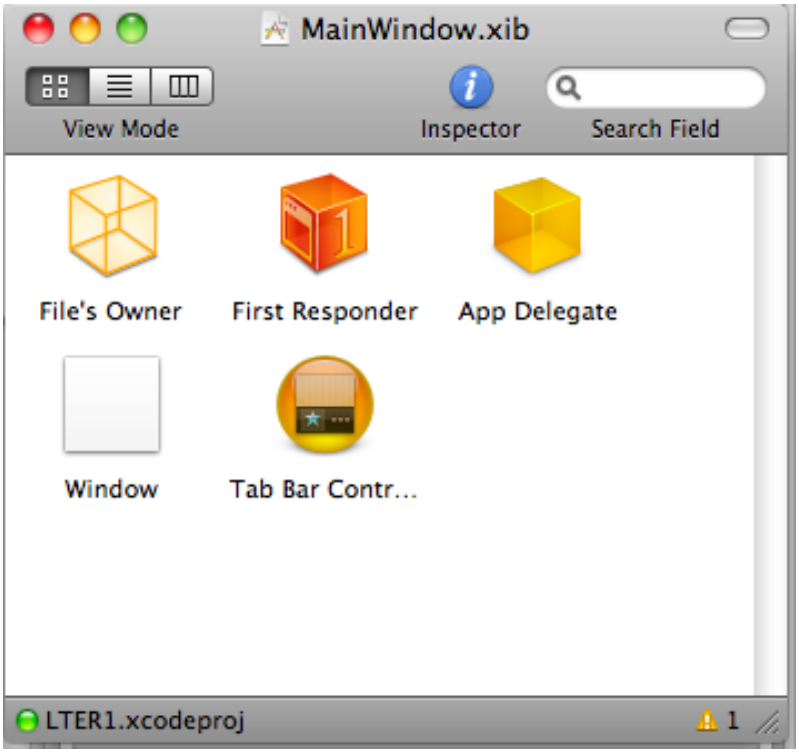

Figure 5.2 – File's Owner of a xib file in Interface Builder

It also has a library and an inspector and each of them serve their purpose. The library is the collection of all the UI controls from which the user can drag and drop the required controls onto the xib file in the interface builder. The inspector is divided into four tabs namely Attributes Inspector, Connections Inspector, Size Inspector and Identity Inspector. The attributes inspector has the properties Simulated User Interface elements like status bar, orientation etc. and View where details about background color and mode are discussed. The connections inspector talks about the methods that will be executed when a particular event occurs. The size inspector gives a description of the size and position of the elements or UI controls that are present in the nib file. The Identity Inspector talks about Class Identity, Accessibility and Interface Builder Identity. The library and the inspector looks as shown below.

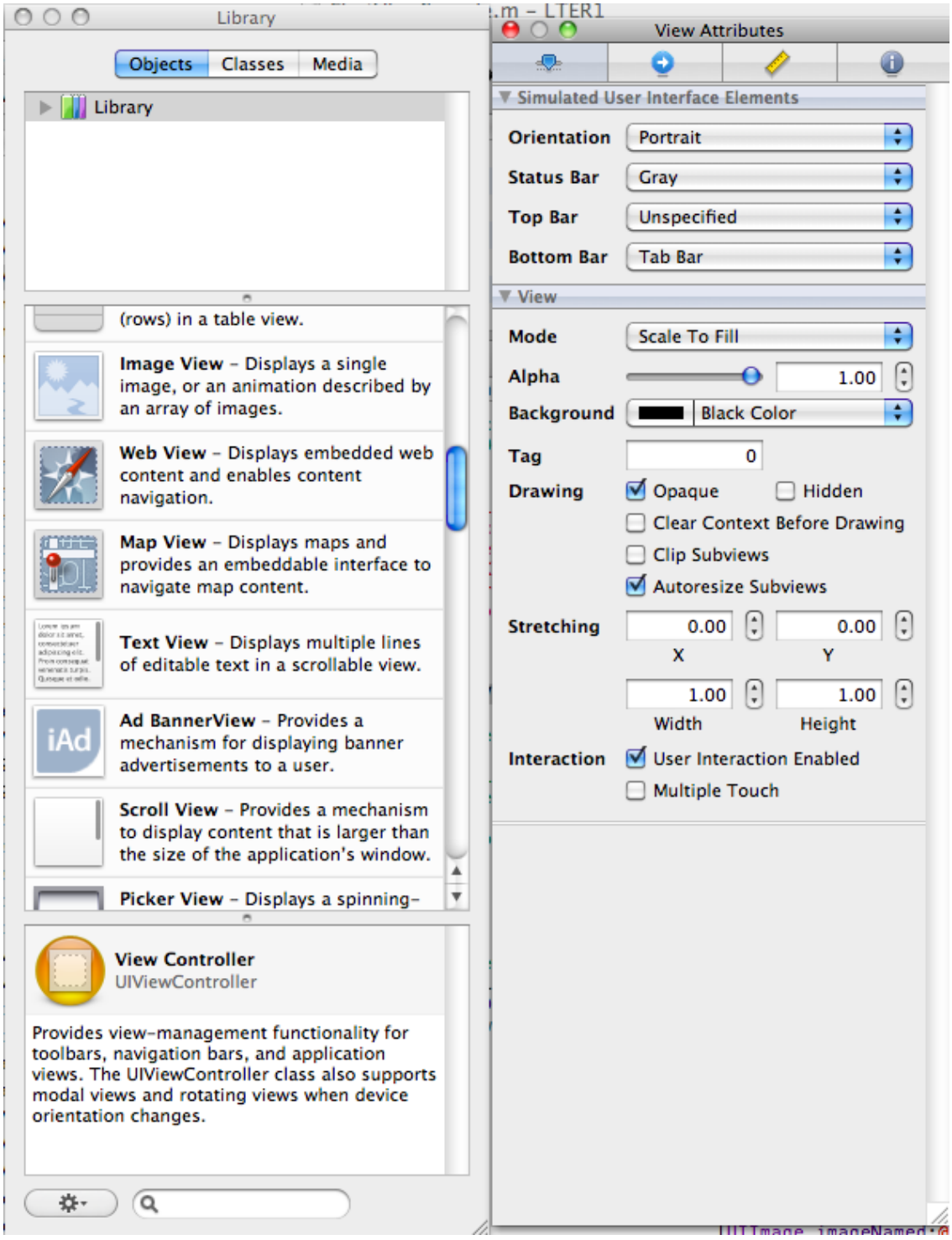

Figure 5.3 – Library and Inspector Interface Builder

Chapter 6 - TESTING OF THE APP

Software testing is the process analyzing a software item to detect the differences between existing and required conditions and to evaluate the features of the software that is developed [1]. It is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. In fact, testing is the one step in the software engineering process that could be viewed as destructive rather than constructive. It is a "Verification-Validation" where two predicates play a very important role: [1]

- Are we building the product right?
- Are we building the right product?

Now, the testing of the iPhone app is done in four phases namely unit testing, integration testing, functional and system testing, regression testing and performance testing.

6.1. Unit Testing

As unit testing focuses verification effort on the smallest unit of software design, starting from a loop, a module or a component, every class, constructor and methods inside the class are tested to check if they give the desired output. Every line of code inside a module is tested here.

Table 6.1- Test Cases of Unit Testing on the app

Class Name	Test Case	Expected Result	Obtained Result	Result
MainWindow	Check if the layout of the screen is properly aligned	The tab bar should be properly aligned	The tab bar is properly aligned	Pass
MainWindow	Check if tab bar items when clicked opens their respective screens	Appropriate screens should be displayed when the ta bar items are clicked	Screens are displayed correctly	Pass
First View Sample	Check if the banner image is aligned properly	The banner image should be aligned properly	The banner image is aligned properly	Pass
First View Sample	Check if "Research" screen is displayed when "Research" button is clicked	"Research" screen should be displayed	"Research" screen is displayed	Pass
First View Sample	Check if "Education" screen is displayed when "Education" button is clicked	"Education" screen should be displayed	"Education" screen is displayed	Pass
First View Sample	Check if the image slide show is displayed properly on the main page	The image slide show should be displayed properly on the main page	The image slide show is displayed properly on the main page	Pass
First View Sample	Check if "Google Map" is displayed when "Map" button is clicked	"Google Map" should be displayed	Google Map is displayed	Pass
First View Sample	Check the layout alignment	Buttons should be aligned in vertical orientation	Buttons are aligned in vertical orientation	Pass
First View Sample	Check if the slide show images of the layout overlap with the image banner	Slide show images should not overlap with image banner	Slide show images does not overlap with image banner	Pass

First View Sample	Check if “About” screen is displayed when “About” button is clicked	“About” screen should be displayed when “About” button is clicked	“About” screen is displayed when “About” button is clicked	Pass
Personnel View	Check if all the headings, names, phone numbers and e-mail ids are aligned properly	The spacing and alignment should fit the screen without any overlap	All views does not overlap	Pass
Personnel View	Check if the respective personnel’s number is being dialed if tapped on it	A call should be made to the number tapped	Call is made to that number	Pass
Personnel View	Check if an e-mail is composed when tapped on the e-mail id of a personnel	An e-mail should be composed to the tapped id	E-mail is composed	Pass
Research View Controller	Check if the Home screen is displayed if Home button is clicked	Home screen should be displayed when Home button is clicked	Home screen is displayed when Home button is clicked	Pass
Map View Controller	Check if the Google Map is displayed when “Map” button is clicked	Google Map should be displayed	Google Map is displayed	Pass
About Konza View	Check if the Home screen is displayed if Home button is clicked	Home screen should be displayed when Home button is clicked	Home screen is displayed when Home button is clicked	Pass
First View Sample	Check if Post Card screen is opened when “Post Card” button is clicked on the Home page	Post Card screen should be displayed.	Post card screen is displayed	Pass
Post Card View	Check if the Home screen is displayed if Home button is clicked	Home screen should be displayed when Home button is clicked	Home screen is displayed when Home button is clicked	Pass
First View Sample	Check if more tab bar items are displayed when “More” is clicked	More tab bar items should be displayed	More ta bar items are displayed	Pass
History	Check if History screen is displayed when History option is selected	History screen should be opened	History screen is opened	Pass
Root View Controller	Check if the list of events are displayed when “Events” tab is clicked	List of events should be displayed	List of events are displayed	Pass

Root View Controller	Check if the list of events are displayed in correct order	The List of events should be displayed in correct order	List of events are displayed in correct order	Pass
Root View Controller	Check if more information about the event is displayed when an event is clicked	More information about the event selected should be displayed	More information about the event selected is displayed	Pass
Personnel	Check if clicking '+' displays more information about that personnel	More information about the personnel should be displayed	More information about that personnel is displayed when '+' is clicked	Pass
Personnel	Check if the user is redirected to the personnel's profile page when clicked on "Full Details" link	The user should be redirected	The user is redirected to the profile page of the personnel when he clicks on the '+' sign	Pass

6.2. Integration Testing

Integration testing is where both the software and components are tested together to check the compatibility and other issues that are involved to evaluate the interaction between them. This testing verifies that the units, in this case, the views work together when they are integrated. Both low-level and high-level designs should be checked thoroughly to understand the flow of control from one view controller to another.

Table 6.2- Test Cases of Integration Testing on the app

Class Name	Test Case	Expected Result	Obtained Result	Result
Main Window	Check if linking the layouts of view controller files causes any logical errors	Adding layouts shouldn't affect the main window	Adding layouts doesn't affect the main window and exceptions are not	Pass

			thrown	
Main Window	Check if navigation from one view controller to another and back gives any errors	Navigation should not give any errors or exceptions	Navigation doesn't give any errors or exceptions	Pass

6.3. Functional and System Testing:

This testing verifies that the software does what is in the requirement specification of the customer. The entire app tested for any logical errors and bugs to ensure that the app is bug-free and provides the desired output.

Table 6.3 - Test Case of Functional and System Testing on the app

Class Name	Test Case	Expected Result	Obtained Result	Result
Main	Check if all view controllers are linked properly	Only selected view controller should be displayed	Only selected view controller is displayed	Pass

6.4. Regression Testing:

This testing, like the functional and system testing uses both white and black box techniques. This testing is performed to make sure that the modifications made to the code does not cause any unintended effects which leads to the failure. This type of testing is actually the subset of all other test cases to ensure that the system still complies with the requirement specification. This testing is done throughout the development cycle of the iPhone application.

Table 6.4 - Test Cases of Regression Testing on the app

Class Name	Test Case	Expected Result	Obtained Result	Result
Main Window	Check if modifying the layout contents and adding of new layouts has unexpected results	No errors or unexpected results should be found	No errors or unexpected results found	Pass
Main Window	Check if adding new Images to the slide show throws any exceptions	Adding new Images should not rise any exceptions	No exceptions are thrown when new images are added to the slide show on the main page	Pass

6.5. Performance Testing:

6.5.1. Loading Test:

The app is testing for its performance and response time. Each screen takes about the 0.1 seconds to load except for the Main Screen which takes 0.2 seconds to load as it has to initialize the images for the slideshow.

6.5.2. Lines of Code:

Also, the app contains a total of 1449 lines of code. The Main Screen alone has 504 lines of code as it has the tab bar items, navigation view controllers and image slide show. The next in place view controller that has 297 lines of code are the events list as it's a table view and hence implements the required methods.

Chapter 7 - SCREENSHOTS:

This is the Home page of the app. The user can select any of the available options that are displayed. The “Map” tab displays the map and shows directions to user on the Konza Prairie trails and provides GPS feature to the user. The “Personnel” tab has the contact details of four primary personnel to contact. The “Research” tab contains information on the research work at Konza. The “About” tab contains detailed information about the Konza LTER. The “Post Card” tab allows the user to take pictures with Konza’s overlay.

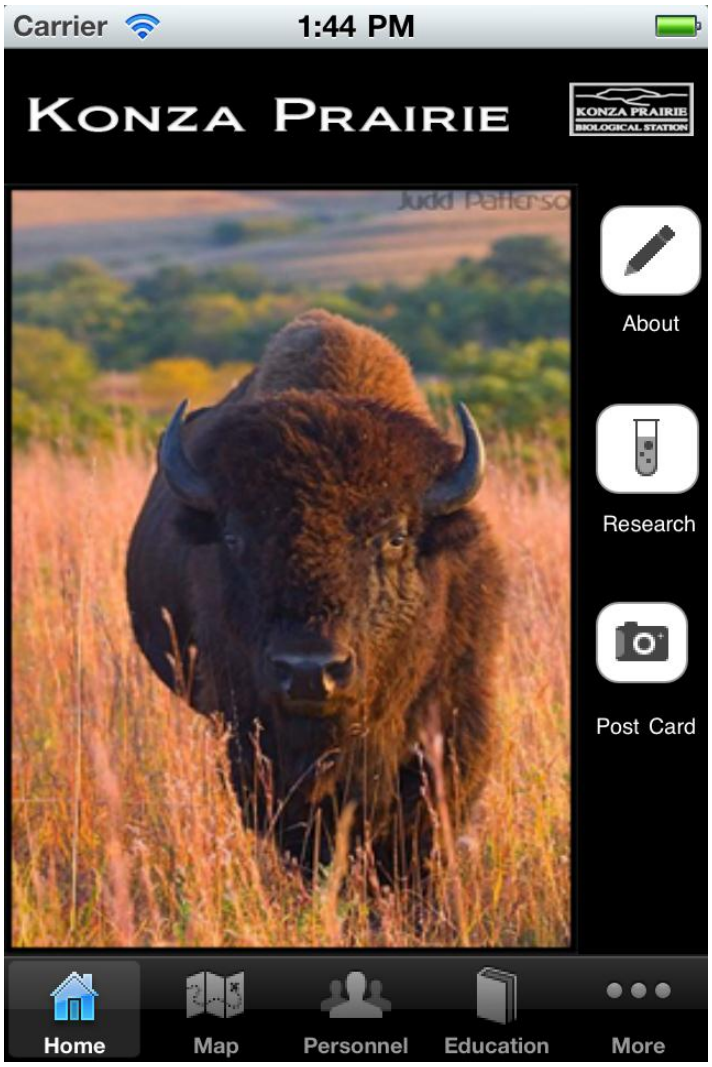

Figure 7.1 - Screenshot of Home Page of the App

When “About” tab is selected, the user is re-directed to another screen. This screen gives the information to the user about Konza Prairie LTER, Konza Prairie Biological Station, its location and environment etc. It also briefs about the research work at Konza, the programs and meetings at the Base Station.

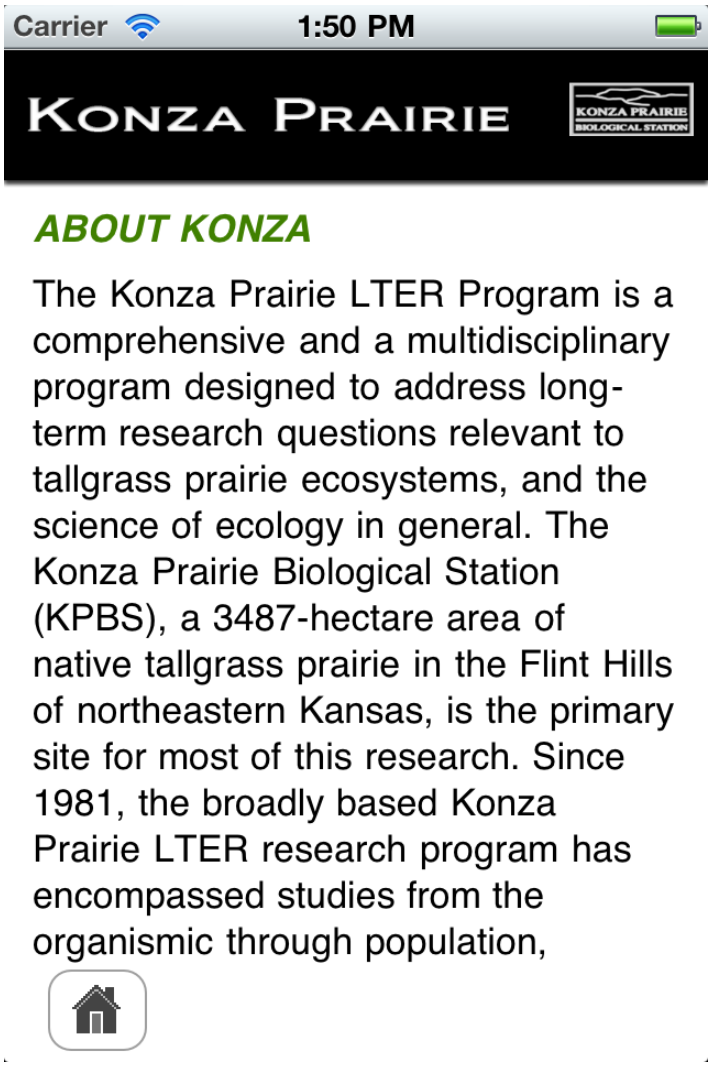

Figure 7.2 - Screenshot of About Page

This screen is displayed when the “Personnel” tab is selected in the Home page. They are the people to be contacted to obtain any information regarding datasets, documents published by Konza and related projects. If he wishes to call or e-mail, then he simply has to tap on the phone number or e-mail id. Also, each person’s designation is mentioned as a high-lighted text so that the user can contact the appropriate person.

Figure 7.3 - Screenshot of Primary Personnel Page

This screen is displayed when the user opts for the GPS feature. This feature is obtained by embedding Google maps to the map. If the user wants to use the zoom controls, he can simply tap on the map and the zoom controls will appear. Then the user can zoom-in and zoom-out. Once he is done with the zooming controls, the controls disappear (auto-hide) again. This feature tracks the user location through the GPS service provider and updates the user on his location updates. In the simulator, the default location is set to California.

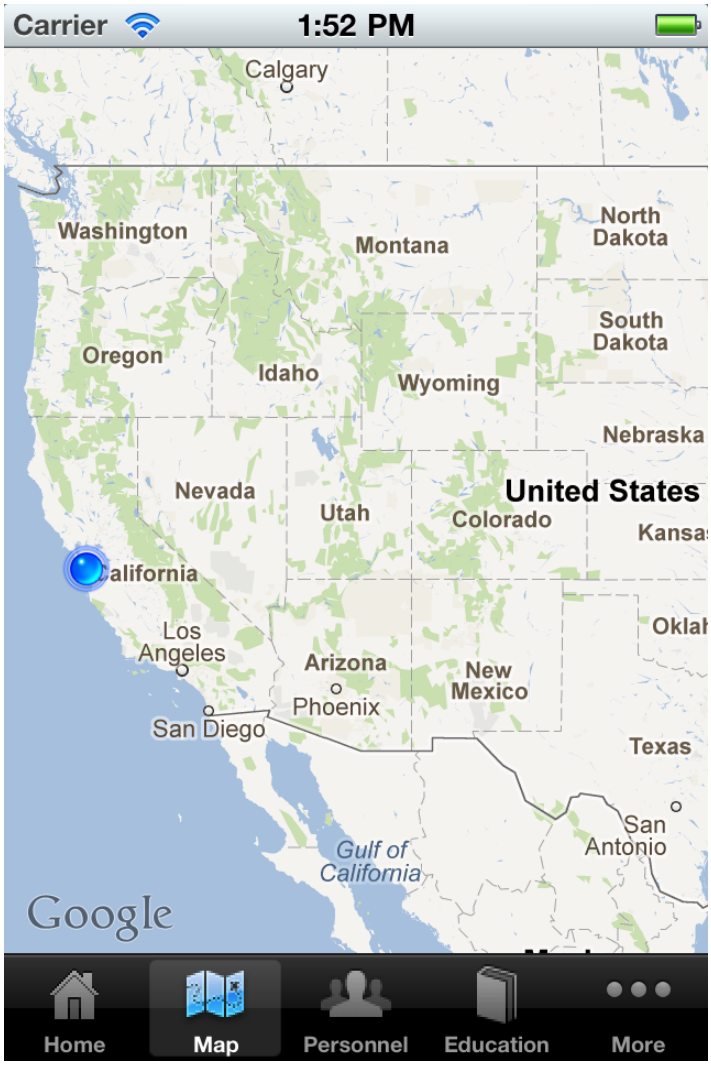

Figure 7.4 - Screenshot of Map View

When the user taps on the “Past Card” tab bar item that is showed in the main page of the app, the user will be re-directed to a new screen that will look as shown below. It enables the user to take pictures and it has a message “Greetings from Konza prairie” overlay in top so that the user can keep it as a souvenir. The pictures taken are stored on the iPhone device in the albums where the other pictures are stored. The user can view the pictures by simply clicking on the pictures folder of the iPhone.

Figure 7.5 – Screenshot of Post Card page

When the user taps on the “Education” tab that is displayed on the “Home” page, he will be redirected to the screen shown below. This screen describes the opportunities provided by Konza to the graduate and under-graduate students. It helps the students to do their research work n their field of interest. It has a special program known as “Konza Environmental Education Program” or in short known as “KEEP” whose goals are listed in this page.

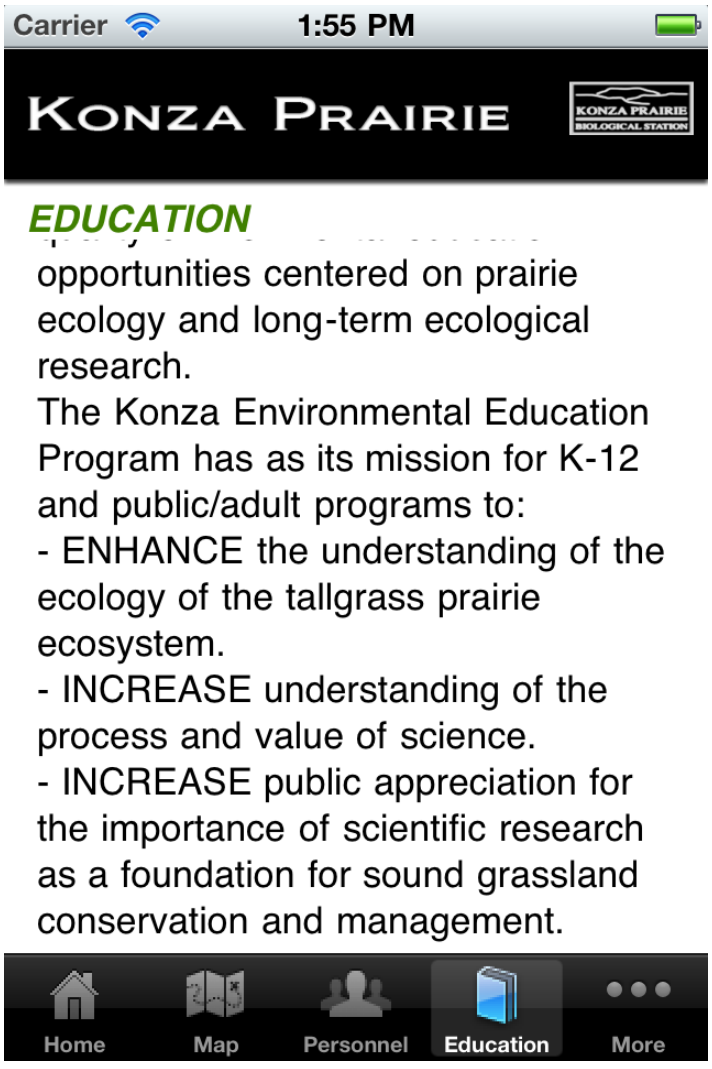

Figure 7.6 – Screenshot of Education page

When the user taps on the “Research” tab that is showed in the main page of the app, this screen is displayed. It briefly states about the research work at Konza, the climatic conditions and the research program of Konza. Also, it lists the current goals of LTER VI and its current and upcoming research projects.

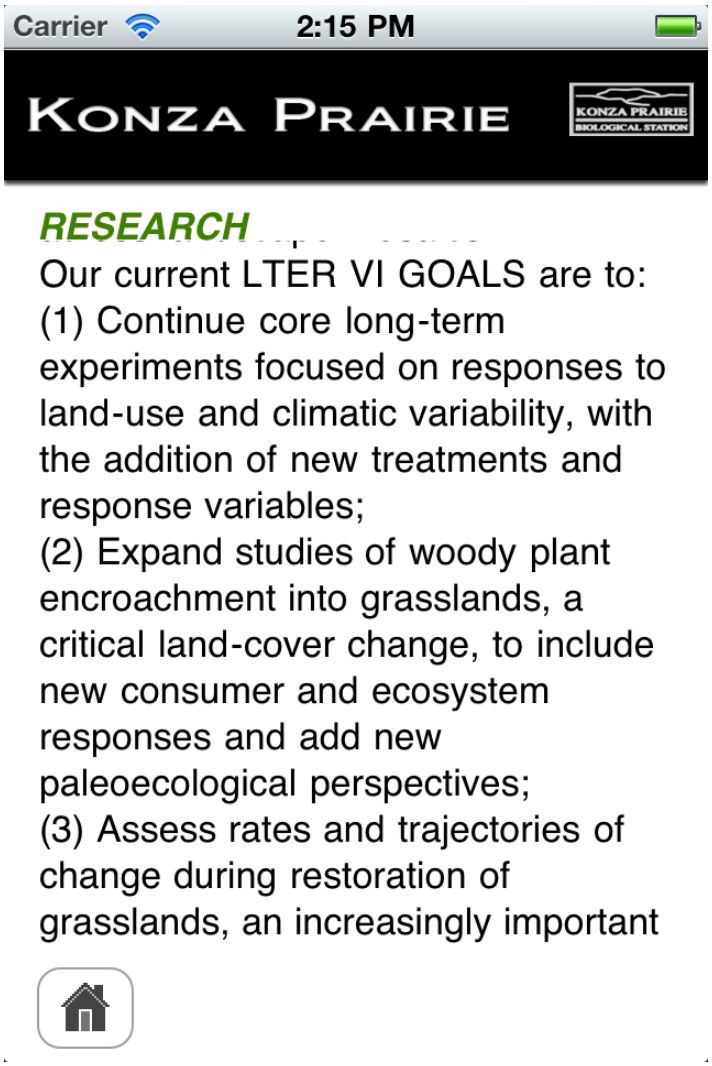

Figure 7.7 - Screenshot of Research page

This screen is displayed when the user selects “More” option in the tab bar. It has two sections namely Events and History. Each of these pages sections when clicked redirects the user to another screen where the appropriate content is displayed. When the user selects the “History” item, a screen with Konza’s history will be displayed to the user. It describes briefly about Konza, the burn history and its LTER I, II, III, IV and V.

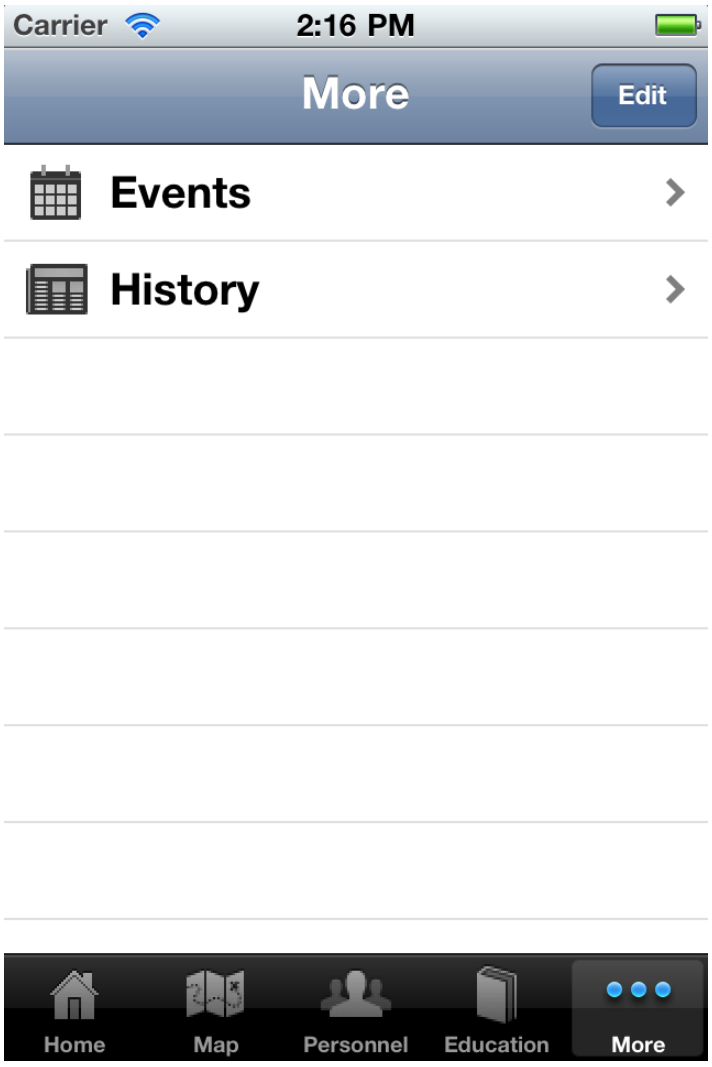

Figure 7.8 - Screenshot of More page

When the user selects the “Events” item that is displayed in the “More” tab bar item of the Home page, this screen will be displayed. It contains a list of upcoming events. This list is generated dynamically from a property list file. Instead storing this data in a separate database and cause more overhead to the app, a property list file is used. Each event when clicked will navigate the user to another screen that has detailed information about that particular event.

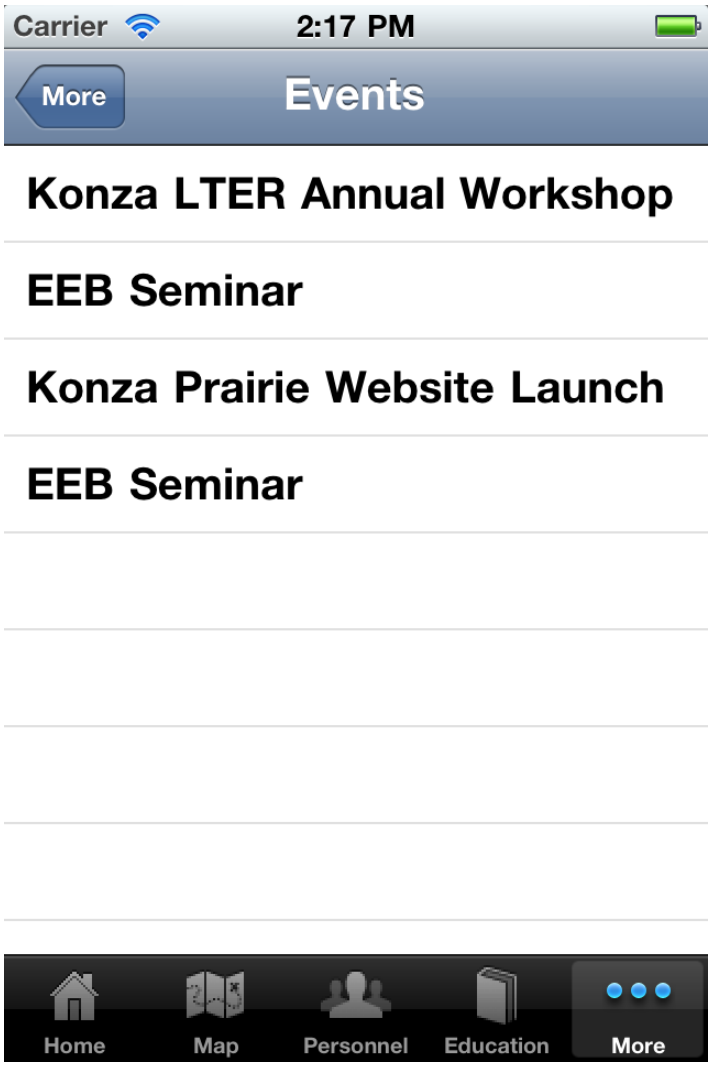

Figure 7.9 - Screenshot of Events List

This screen is displayed when the user selects a particular event from the events list. It has s detailed description of the event like the date on which the event takes place, the place at which the event takes place, the title of the event i.e., the topic under discussion and the details. It also a link which when clicked by the user navigates him to open a page which will have more information about that particular event.

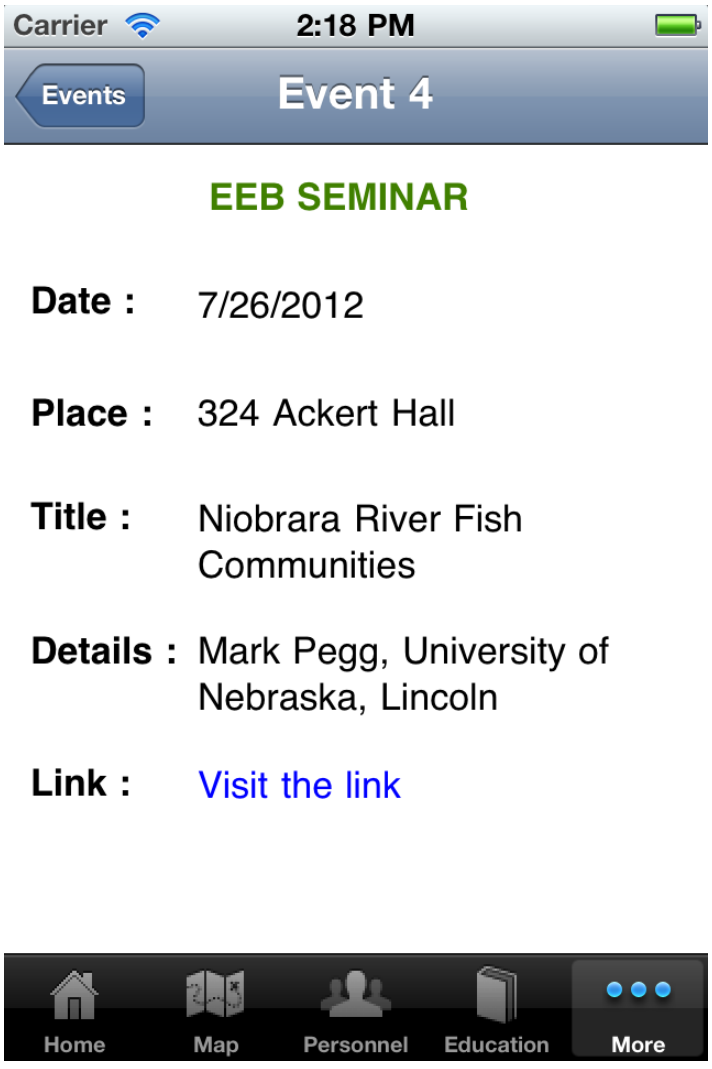

Figure 7.10 - Screenshot of Event Detail

This screenshot is taken from website of Konza Prairie LTER. It can be viewed under 'Personnel' tab of the available drop-down menus. It has the primary contacts information that the users can use to contact if they need any information regarding the datasets or publications. Each Personnel has a designation, their name, phone number, e-mail id and a '+' sign. This '+' when clicked gives more information about that particular personnel.

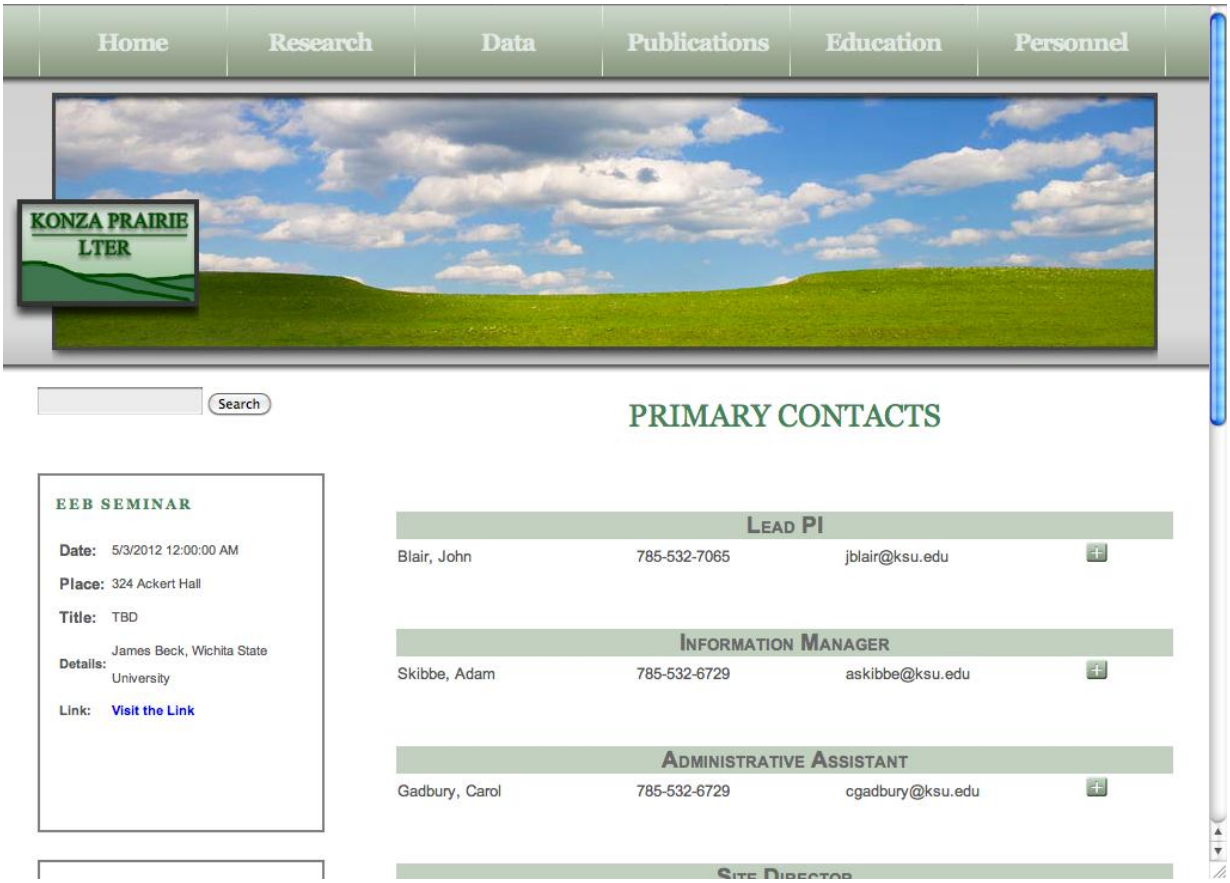

Figure 7.11 – Screenshot of Primary Contacts page on website [4]

The screen is displayed when the user clicks on the '+' on the personnel page. It gives opens a small rectangular box on the same page which displays the information of the personnel's mailing address, their interests and a link 'Full Details' which when clicked navigates the user to the profile page of that personnel which has detailed information about the personnel.

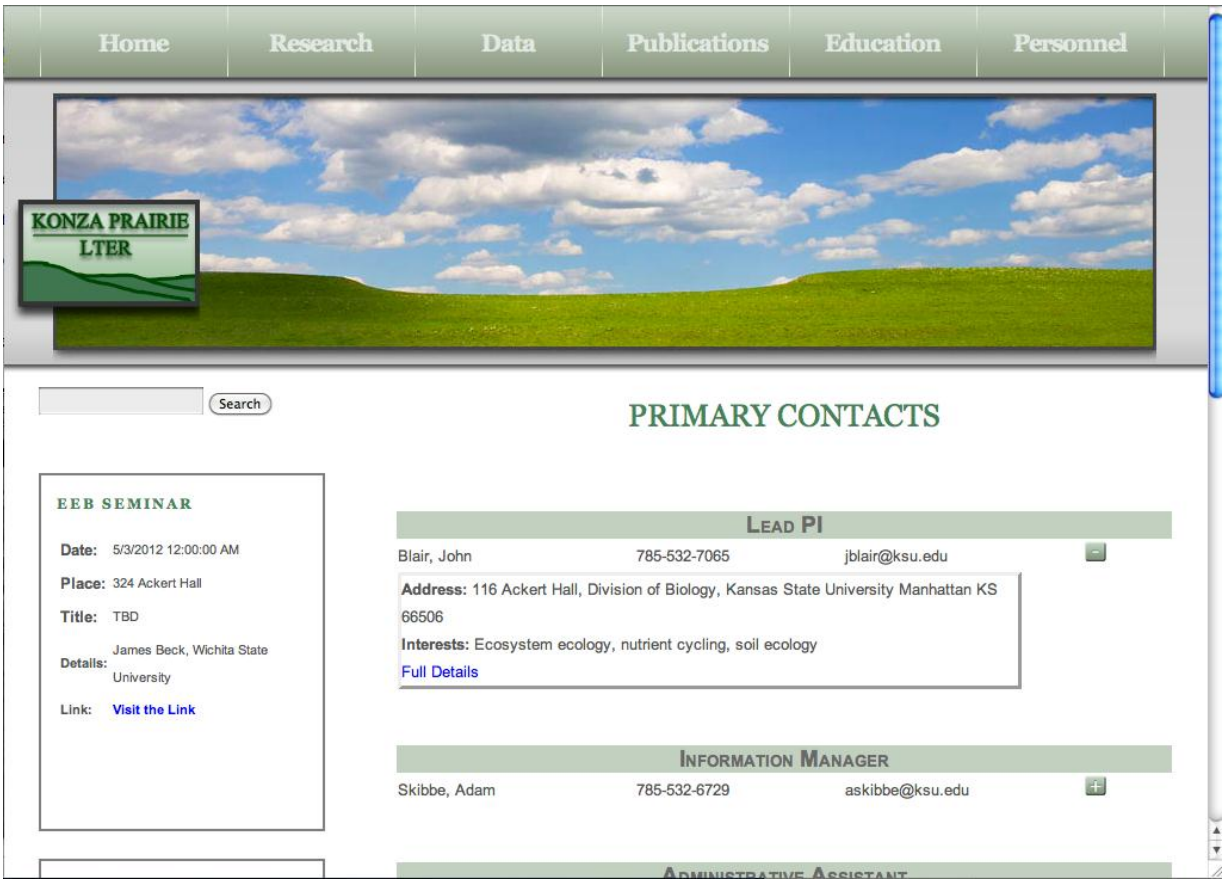

Figure 7.12 – Screenshot of Primary Contacts when '+' is clicked

Chapter 8 - CONCLUSION AND FUTURE WORK

This project provided me with an exciting and challenging environment as I was a beginner in iOS development. It provided me with a real-time experience on iOS development and how things work inside the box. As iPhone SDK is something that I didn't use in my past, this project enhanced my knowledge in iPhone SDK tools, its development environment and platform.

The future work of this project involves registering this app in iTunes and implementing the augmented reality through the device's camera that can be used to know the directions when the user is on Konza's Trails.

Chapter 9 - REFERENCES:

[1]. E. F. Miller, "Introduction to Software Testing Technology," *Tutorial: Software Testing & Validation Techniques,* Second Edition, IEEE Catalog No. EHO 180-0, pp. 4-16

[2]. iOS Developer Library, *UI Controls,* Accessed 20 MAY 2012 – 16 JUN 2012

http://developer.apple.com/library/ios/navigation/

[3]. iOS Developers, *Xcode 4 Download,* Accessed 06 MAY 2012

https://developer.apple.com/xcode/

[4]. Konza Prairie LTER, *Personnel Information,* Accessed 27 MAY 2012

http://www.konza.ksu.edu/KNZ/pages/personnel/main_personnel.aspx