Location Based Notification of Events

Manikanth Reddy Lekkala

CIS-695, Fall and 2017

Dr. Solomon Antony

Contents

[Acknowledgements 3](#_Toc500536631)

[Project abstract 4](#_Toc500536632)

[Glossary of terms 5](#_Toc500536633)

[Organizational Background 6](#_Toc500536634)

[Project Requirements 6](#_Toc500536635)

[Design and development of solution: 7](#_Toc500536636)

[User feedback and Business value of the solution: 9](#_Toc500536637)

[Conclusion and future directions 11](#_Toc500536638)

[Appendices: 12](#_Toc500536639)

# Acknowledgements

I would like to express my special thanks of gratitude to my project supervisor Dr. Solomon Antony for giving me the opportunity to do the Location Based Notification of Events project and supporting me throughout this project, which also helped me learning new technologies by doing a lot of research and I also came to know a lot of new things. I am thankful to him.

Secondly, I would like to thank my friends and other professors who helped me a lot in finalizing this project with in the time frame.

# Project abstract

The purpose of my project is to provide notifications or alerts to the users regarding the event that are near to him based on his location. Our Murray State University website has a Helios calendar which is integrated into the website and a lot of people need to visit website all the time to check and keep updated regarding the events and we don’t have an application to remind those events based on location. Here we are using the same public calendar powered by Helios Calendar, where we post the events and its details, and those details are accessed by an android application developed using Xamarin and C# through API. This application run in background to see if user is in the location of any events that are published in Helios calendar.

# Glossary of terms

|  |  |
| --- | --- |
| Helios Calendar | This is an open source public calendar, can be used by any organization or person. This can help us create, save, update and delete events according to the user requirements. Here any one can create an event, but admin must approve it to see that on calendar. |
| Geofence Library | This is a visual studio NuGet package (library), which help us monitor the location of the user by running in background and alerts the user if he/she is in with the monitoring range. |
| Xamarin | Visual studio is incorporated with Xamarin, which help us build cross-platform applications with a C#-shared codebase. Developers can use Xamarin tools to write native Android, iOS, and Windows apps with native user interfaces and share code across multiple platforms, including Windows and macOS. |

# Organizational Background

This project is developed for Murray state university, which will help Murray state Students, faculty and staff remind their events based on their location. The main aim of this project is to provide push notifications to users regarding the events near them based on their preferences using their location (Location based notifications) and notifying user the deadline for the event (Time based notifications), through an Android and IOS application. Here even the user can create an event, but it must be approved by the web admin to provide push notifications for all the users. Users can also sync their Murray state Gmail calendar to get push notifications regarding their submissions. This project provides me an opportunity to learn C# with Xamarin for deploying Hybrid application.

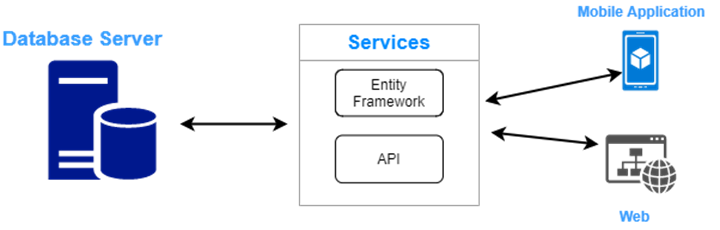
# Project Requirements

This project requirements are briefed out as follows:

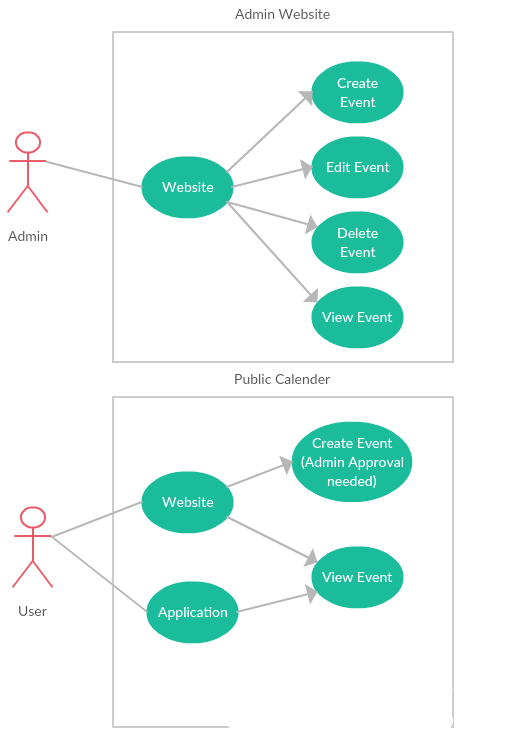
1. Analysis of the current calendar and identifying the flaws of current calendar.
2. Searching for an open source calendar that our college uses so that we don’t find any problem in migrating the calendar.
3. Search for an open source location based notification library for Xamarin to monitor the locations in the application.

Functional Requirements

1. This project will help Murray state Students, faculty and staff remind their events based on their location and time of the event.
2. Architecture diagram, Here I will be using Xamarin and C# to do mobile application and SQL is the database which communicates with App using WCF service.



1. Use case diagrams is shown below:



Design and development of solution:

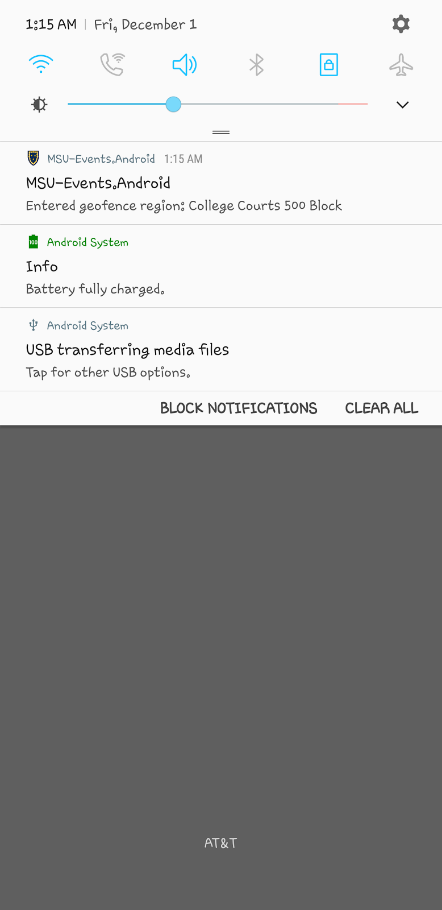
1. User interface of the hybrid application.



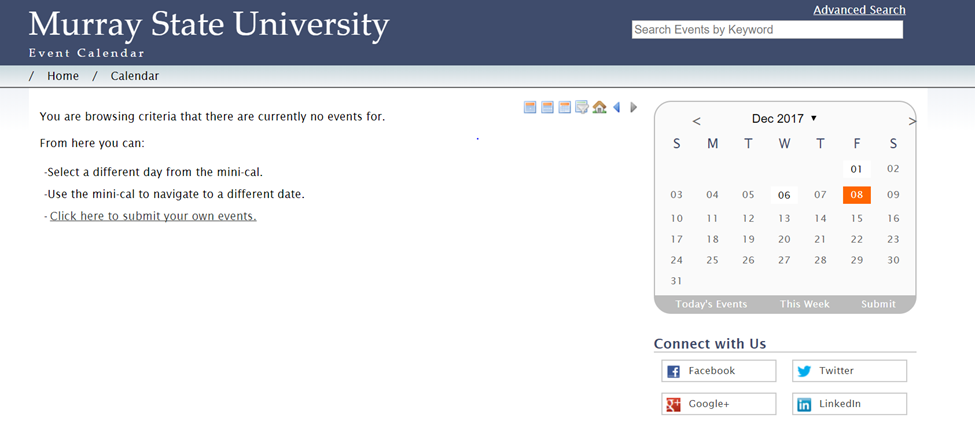
*Boot Screen of Application*



*Main Screen of Application*



*Notification by Application when we enter the monitoring region*



*Helios calendar hosted on lab server*

Calendar URL: **csclab.murraystate.edu/mlekkala/**



*Admin portal of the website*

User name and password of the admin portal are:

URL: **csclab.murraystate.edu/mlekkala/admin**

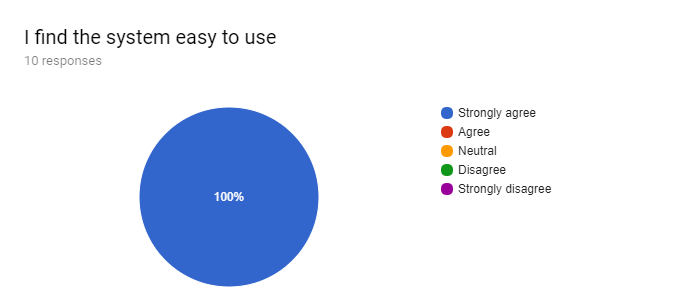
User name: [mlekkala@murraystate.edu](mailto:mlekkala@murraystate.edu)

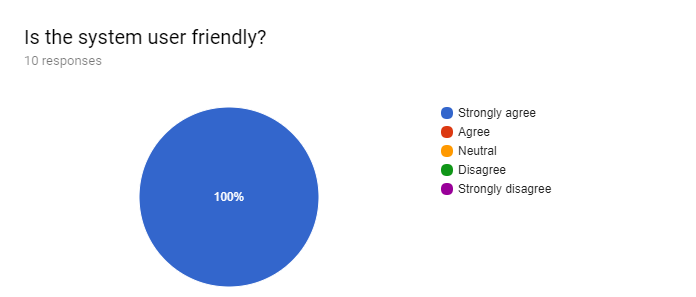
Password: 123456

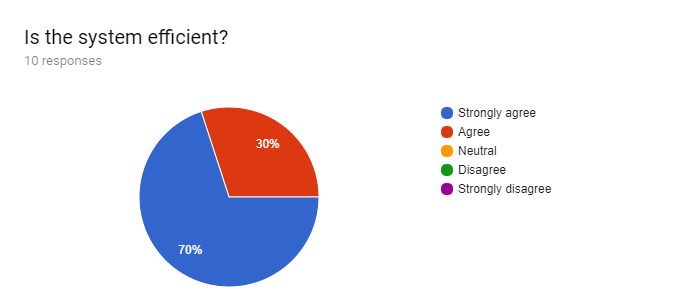
This website provides the API, from which application gets the data as well as latitudes and longitudes to start monitoring the locations of the events.

User feedback and Business value of the solution:

Project is hosted on our CSCLAB server and application is shared through a link by hosting the application on the same server website is hosted and a user’s feedback is collected through google form.







Conclusion and future directions

This Proposed solution can help users remember the events. It also helps user notify the events that are going to happen based on his location. Present calendar of the Murray State University could be improved by using this proposed solution, which help students notify the events based on their location on campus.

The future directions for this application would be notifying about the events based on user preferences. There are few limitations such as when we click on the notification alert that we received, it only opens the app and doesn't filter the events based on the location. If a new location or a new event of a new location is created, we need to open the application at least once to start monitoring the newly added location. If we create an event for the first time in a location, we need to open the application for at least once to start monitoring the location. These limitations could be solved in future.

Appendices:

1. Source code

**Events.cs:** (A class to get the JSON data)

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace MSU\_Events

{

public class Api

{

public string version { get; set; }

public string cal\_url { get; set; }

public string encoding { get; set; }

public DateTime generated { get; set; }

public string contents { get; set; }

}

public class Categories

{

public string \_\_invalid\_name\_\_1 { get; set; }

}

public class Data

{

public string id { get; set; }

public string title { get; set; }

public DateTime date\_start { get; set; }

public DateTime date\_end { get; set; }

public string venue\_id { get; set; }

public string venue\_name { get; set; }

public string venue\_add { get; set; }

public string venue\_add2 { get; set; }

public string venue\_city { get; set; }

public string venue\_region { get; set; }

public string venue\_postal { get; set; }

public string venue\_country { get; set; }

public string venue\_lat { get; set; }

public string venue\_lon { get; set; }

public string contact { get; set; }

public string contact\_email { get; set; }

public string contact\_phone { get; set; }

public string image { get; set; }

public string billboard { get; set; }

public string series\_id { get; set; }

public string registration { get; set; }

public string registration\_url { get; set; }

public Categories categories { get; set; }

}

public class RootObject

{

public Api api { get; set; }

public List<Data> Data { get; set; }

}

}

**SplashPage.cs:** (Splash Screen code)

using System;

using System.Collections.Generic;

using System.Linq;

using System.Diagnostics;

using System.Text;

using System.Threading.Tasks;

using Xamarin.Forms;

using System.Net.Http;

using Newtonsoft.Json;

using Geofence.Plugin;

using Geofence.Plugin.Abstractions;

namespace MSU\_Events

{

public class loaddata

{

public static string[] title

{

get;

set;

}

}

class SplashPage : ContentPage

{

Image splashImage;

Label title;

public SplashPage()

{

NavigationPage.SetHasNavigationBar(this, false);

var sub = new AbsoluteLayout();

title = new Label

{

TextColor = Color.Red

};

splashImage = new Image

{

Source = "logo.png",

WidthRequest = 350,

HeightRequest = 350

};

AbsoluteLayout.SetLayoutFlags(splashImage,

AbsoluteLayoutFlags.PositionProportional);

AbsoluteLayout.SetLayoutBounds(splashImage,

new Rectangle(0.5, 0.5, AbsoluteLayout.AutoSize, AbsoluteLayout.AutoSize));

//var label = new Label { TextColor = Color.FromHex("#77d065"), FontSize = 20 };

//label.Text = loaddata.title[0];

sub.Children.Add(splashImage);

sub.Children.Add(title);

this.BackgroundColor = Color.FromHex("#002144");

this.Content = sub;

}

protected override async void OnAppearing()

{

base.OnAppearing();

await splashImage.ScaleTo(1, 1000); //Time-consuming processes such as initialization

await splashImage.ScaleTo(0.9, 2000, Easing.Linear);

await splashImage.ScaleTo(0, 1000, Easing.Linear);

//Application.Current.MainPage = new MainPage(); //After loading MainPage it gets Navigated to our new Page

Application.Current.MainPage = new ListViewEvents();//new NavigationPage(new ListViewEvents());

//MainPage.Navigation.PushAsync(new interactiveListViewXaml());

//this.Navigation.PushAsync(new ListViewEvents(), true);

}

}

}

**ListView.cs:** (List view code to see the events)

using System;

using System.Diagnostics;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Net.Http;

using Newtonsoft.Json;

using System.Collections.ObjectModel;

using Geofence.Plugin;

using Geofence.Plugin.Abstractions;

using Xamarin.Forms;

namespace MSU\_Events

{

public class ListViewEvents : ContentPage

{

public static ObservableCollection<string> items { get; set; }

public ListViewEvents()

{

Image header = new Image

{

BackgroundColor = Color.FromHex("#002144"),

Source = "MSU12.jpg",

HorizontalOptions = LayoutOptions.Fill

};

//Label header = new Label

//{

// BackgroundColor = Color.FromHex("#0b1e36"),

// HorizontalOptions = LayoutOptions.Fill,

// VerticalOptions =LayoutOptions.Fill,

// Text = "Murray State University",

// FontSize = 24,//Device.GetNamedSize(NamedSize.Large, typeof(Label)),

// FontAttributes = FontAttributes.Bold,

// HorizontalTextAlignment = TextAlignment.Center,

// TextColor = Color.White

//};

items = new ObservableCollection<string>() { };

ListView lstView = new ListView()

{

RowHeight = 50,

};

var temp = new DataTemplate(typeof(textViewCell));

lstView.ItemTemplate = temp;

lstView.IsPullToRefreshEnabled = true;

lstView.Refreshing += OnRefresh;

lstView.ItemSelected += OnSelection;

lstView.ItemTapped += OnTap;

lstView.ItemsSource = items;

//Content = lstView;

this.Content = new StackLayout

{

Children =

{

header,

lstView

}

};

}

protected override async void OnAppearing()

{

items.Clear();

using (var client = new HttpClient())

{

HttpResponseMessage response = await client.GetAsync("http://csclab.murraystate.edu/mlekkala/api/?u=murray&k=racers&data=events\_c");

response.EnsureSuccessStatusCode();

using (HttpContent content = response.Content)

{

string responseBody = await response.Content.ReadAsStringAsync();

var articles = JsonConvert.DeserializeObject<RootObject>(responseBody);

foreach (var item in articles.Data)

{

var reg = new GeofenceCircularRegion(item.venue\_name, Convert.ToDouble(item.venue\_lat), Convert.ToDouble(item.venue\_lon), 100)

{

//To get notified if user stays in region for at least 5 minutes

NotifyOnStay = true,

StayedInThresholdDuration = TimeSpan.FromMinutes(1)

};

CrossGeofence.Current.StartMonitoring(reg);

items.Add(item.title.ToString());

}

}

}

}

async void OnTap(object sender, ItemTappedEventArgs e)

{

using (var client = new HttpClient())

{

HttpResponseMessage response = await client.GetAsync("http://csclab.murraystate.edu/mlekkala/api/?u=murray&k=racers&data=events\_c");

response.EnsureSuccessStatusCode();

using (HttpContent content = response.Content)

{

string responseBody = await response.Content.ReadAsStringAsync();

var articles = JsonConvert.DeserializeObject<RootObject>(responseBody);

foreach (var item in articles.Data)

{

if (item.title == e.Item.ToString())

{

await DisplayAlert(item.title, "Location: " +item.venue\_name +"\n Date: " + item.date\_start, "Ok");

}

}

}

}

}

void OnSelection(object sender, SelectedItemChangedEventArgs e)

{

if (e.SelectedItem == null)

{

return; //ItemSelected is called on deselection, which results in SelectedItem being set to null

}

//DisplayAlert("Item Selected", e.SelectedItem.ToString(), "Ok");

//comment out if you want to keep selections

ListView lst = (ListView)sender;

lst.SelectedItem = null;

}

async void OnRefresh(object sender, EventArgs e)

{

var list = (ListView)sender;

//put your refreshing logic here

items.Clear();

using (var client = new HttpClient())

{

HttpResponseMessage response = await client.GetAsync("http://csclab.murraystate.edu/mlekkala/api/?u=murray&k=racers&data=events\_c");

response.EnsureSuccessStatusCode();

using (HttpContent content = response.Content)

{

string responseBody = await response.Content.ReadAsStringAsync();

var articles = JsonConvert.DeserializeObject<RootObject>(responseBody);

foreach (var item in articles.Data)

{

items.Add(item.title.ToString());

//title.Text += item.title;

//Debug.WriteLine("Hello " + item.title);

}

}

}

//make sure to end the refresh state

list.IsRefreshing = false;

}

}

public class textViewCell : ViewCell

{

public textViewCell()

{

StackLayout layout = new StackLayout();

layout.Padding = new Thickness(15, 0);

Label label = new Label();

label.SetBinding(Label.TextProperty, ".");

layout.Children.Add(label);

//var moreAction = new MenuItem { Text = "More" };

//moreAction.SetBinding(MenuItem.CommandParameterProperty, new Binding("."));

//moreAction.Clicked += OnMore;

//var deleteAction = new MenuItem { Text = "Delete", IsDestructive = true }; // red background

//deleteAction.SetBinding(MenuItem.CommandParameterProperty, new Binding("."));

//deleteAction.Clicked += OnDelete;

//this.ContextActions.Add(moreAction);

//this.ContextActions.Add(deleteAction);

View = layout;

}

void OnMore(object sender, EventArgs e)

{

var item = (MenuItem)sender;

//Do something here... e.g. Navigation.pushAsync(new specialPage(item.commandParameter));

//page.DisplayAlert("More Context Action", item.CommandParameter + " more context action", "OK");

}

void OnDelete(object sender, EventArgs e)

{

var item = (MenuItem)sender;

ListViewEvents.items.Remove(item.CommandParameter.ToString());

}

}

}