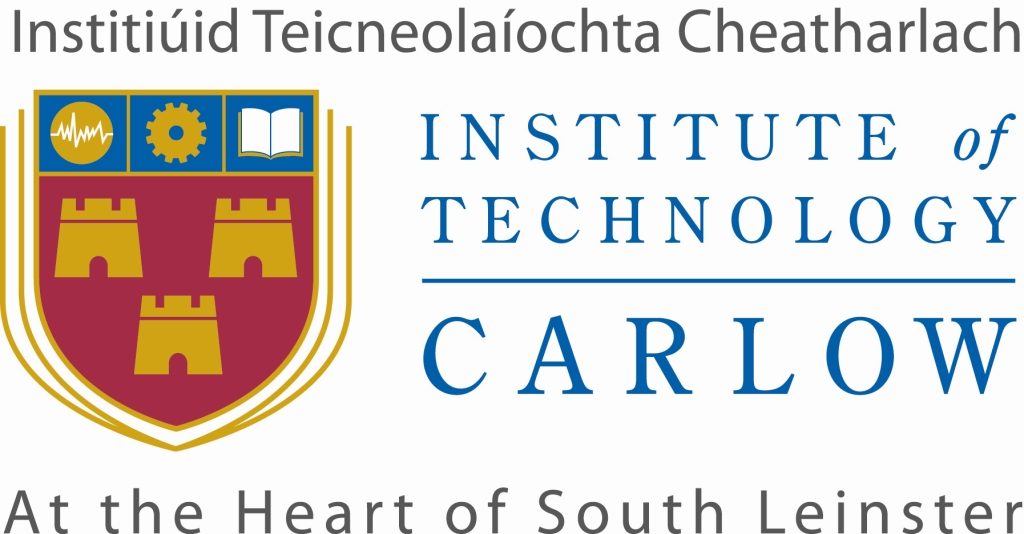
CAIRDE

**Car Sharing App**

Code Manual



18th April 2018

BSc (Hons) Software Development

**Name:** Dylan Lawlor

**Year:** 4th year

**Student ID:** C00197013

**Supervisor:** Lei Shi

# Abstract

The purpose of this document is to provide the source code written by the author for the project.

Contents

[Abstract 1](#_Toc511824980)

[AndroidManifest.xml 5](#_Toc511824981)

[AssemblyInfo.cs 5](#_Toc511824982)

[DBHelper.cs 6](#_Toc511824983)

[Journeys.cs 13](#_Toc511824984)

[LocationHelper.cs 14](#_Toc511824985)

[Search.cs 17](#_Toc511824986)

[UserProfiles.cs 18](#_Toc511824987)

[Users.cs 18](#_Toc511824988)

[ApplicantsMenu.xml 19](#_Toc511824989)

[CreateJourney.axml 20](#_Toc511824990)

[DriverJourneys.axml 23](#_Toc511824991)

[DriverJourneys.axml 24](#_Toc511824992)

[DriverOldJourneys.axml 24](#_Toc511824993)

[JourneyMenu.xml 25](#_Toc511824994)

[Main.axml 26](#_Toc511824995)

[MainProfile.axml 28](#_Toc511824996)

[ProfileSetUp.axml 30](#_Toc511824997)

[Register.axml 32](#_Toc511824998)

[SearchJourney.axml 34](#_Toc511824999)

[SearchJourneyMenu.axml 37](#_Toc511825000)

[SearchJourneyResults.axml 38](#_Toc511825001)

[ShowJourney.axml 39](#_Toc511825002)

[ViewOldPassengerJourneys.axml 41](#_Toc511825003)

[ViewUpcomingPassengerJourneys.axml 42](#_Toc511825004)

[CreateJourneyActivity.cs 43](#_Toc511825005)

[DriverJourneysActivity.cs 50](#_Toc511825006)

[DriverOldJourneysActivity.cs 52](#_Toc511825007)

[MainActivity.cs 54](#_Toc511825008)

[MainProfileActivity.cs 57](#_Toc511825009)

[packages.config 59](#_Toc511825010)

[PasswordStorage.cs 61](#_Toc511825011)

[RegisterActivity.cs 65](#_Toc511825012)

[SearchJourney.cs 67](#_Toc511825013)

[SearchJourneyResultsActivity.cs 74](#_Toc511825014)

[SetUpProfileActivity.cs 76](#_Toc511825015)

[ShowJourney.cs 79](#_Toc511825016)

[ViewOldPassengerJourneysActivity.cs 84](#_Toc511825017)

[ViewPassengerJourneysActivity.cs 86](#_Toc511825018)

# AndroidManifest.xml

<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="CarShare.CarShare" android:versionCode="1" android:versionName="1.0" android:installLocation="auto">

<uses-sdk android:minSdkVersion="19" />

<uses-feature android:glEsVersion="0x00020000" android:required="true" />

<uses-permission android:name="android.permission.INTERNET" />

<uses-permission android:name="com.google.android.providers.gsf.permission.READ\_GSERVICES" />

<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />

<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />

<uses-permission android:name="CarShare.CarShare.permission.MAPS\_RECEIVE" />

<uses-permission android:name="CarShare.CarShare.permission.MAPS\_RECEIVE" android:protectionLevel="signature" />

<uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION" />

<uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION" />

<application android:allowBackup="true" android:label="@string/app\_name">

       <meta-data android:name="com.google.android.geo.API\_KEY" android:value="AIzaSyAAetm6iDbYHp4q4XR5Cj2E7GHEK9a\_WGQ" />

       <meta-data android:name="com.google.android.gms.version" android:value="@integer/google\_play\_services\_version" />

</application>

</manifest>

# AssemblyInfo.cs

using System.Reflection;

using System.Runtime.CompilerServices;

using System.Runtime.InteropServices;

using Android.App;

// General Information about an assembly is controlled through the following

// set of attributes. Change these attribute values to modify the information

// associated with an assembly.

[assembly: AssemblyTitle("CarShare")]

[assembly: AssemblyDescription("")]

[assembly: AssemblyConfiguration("")]

[assembly: AssemblyCompany("")]

[assembly: AssemblyProduct("CarShare")]

[assembly: AssemblyCopyright("Copyright ©  2018")]

[assembly: AssemblyTrademark("")]

[assembly: AssemblyCulture("")]

[assembly: ComVisible(false)]

// Version information for an assembly consists of the following four values:

//

//   Major Version

//   Minor Version

//   Build Number

//   Revision

//

// You can specify all the values or you can default the Build and Revision Numbers

// by using the '\*' as shown below:

// [assembly: AssemblyVersion("1.0.\*")]

[assembly: AssemblyVersion("1.0.0.0")]

[assembly: AssemblyFileVersion("1.0.0.0")]

# DBHelper.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using Android.App;

using Android.Content;

using Android.OS;

using Android.Runtime;

using Android.Views;

using Android.Widget;

using Microsoft.WindowsAzure.MobileServices;

using System.Threading.Tasks;

using Android.Gms.Maps.Model;

using SendGrid;

using SendGrid.Helpers.Mail;

using Android.Content.Res;

namespace CarShare.Models

{

public static class DBHelper

{

    public static MobileServiceClient MobileService =

    new MobileServiceClient("https://c00197013.azurewebsites.net");

    public const string SendGridAPIKey = "SG.g9qKo2r3TaKdiXlyfnCqvQ.R\_EN81TkrVMXVrij\_7LUijQHNeKvToN5t57ItJqfxVM";

    public static async void InsertNewUser(Users u)

    {

           CurrentPlatform.Init();

           MobileService.GetTable<Users>().InsertAsync(u);

    }

    public static async void InsertNewUser(string userName, string password)

    {

           CurrentPlatform.Init();

       string hashedPassword = PasswordStorage.CreateHash(password);

           //CurrentPlatform.Init();

        Users u = new Users { Username = userName, Password = hashedPassword };

           MobileService.GetTable<Users>().InsertAsync(u);

    }

    public static async Task<Users> GetUser(string userName)

    {

           CurrentPlatform.Init();

        List<Users> ls = await MobileService.GetTable<Users>().ToListAsync();

        Users u = ls.FirstOrDefault(x => x.Username == userName);

        return u;

    }

    public static async Task<List<Users>> GetAllUsers()

    {

           CurrentPlatform.Init();

        List<Users> ls = await MobileService.GetTable<Users>().ToListAsync();

        return ls;

    }

    public static async Task<bool> DoesUserExist(string userName)

    {

           CurrentPlatform.Init();

        List<Users> ls = await MobileService.GetTable<Users>().ToListAsync();

        Users u = ls.FirstOrDefault(x => x.Username == userName);

        if (u != null)

        {

            return true;

        }

        else

        {

            return false;

        }

    }

    public static async void InsertUserProfile(UserProfiles up)

    {

           CurrentPlatform.Init();

           MobileService.GetTable<UserProfiles>().InsertAsync(up);

    }

    public static async void InsertUserProfile(string userID, string firstName, string lastName, string email, string phoneNo, string gender, string county)

    {

           CurrentPlatform.Init();

        UserProfiles up = new UserProfiles

        {

            UsersID = userID,

            Firstname = firstName,

            Lastname = lastName,

            Email = email,

            PhoneNo = phoneNo,

            Gender = gender,

            County = county

        };

           MobileService.GetTable<UserProfiles>().InsertAsync(up);

    }

    public static async Task<UserProfiles> GetUsersProfile(string userID)

    {

           CurrentPlatform.Init();

           List<UserProfiles> ls = await MobileService.GetTable<UserProfiles>().ToListAsync();

        UserProfiles u = ls.FirstOrDefault(x => x.UsersID == userID);

        return u;

    }

    public static async Task<bool> IsUserProfileCreated(string userID)

    {

           CurrentPlatform.Init();

           List<UserProfiles> ls = await MobileService.GetTable<UserProfiles>().ToListAsync();

        UserProfiles u = ls.FirstOrDefault(x => x.UsersID == userID);

        if (u != null)

        {

            return true;

        }

        else

        {

            return false;

        }

    }

    public static async void InsertJourney(Journeys j)

    {

           CurrentPlatform.Init();

           MobileService.GetTable<Journeys>().InsertAsync(j);

    }

    public static async void UpdateJourney(Journeys j)

    {

       CurrentPlatform.Init();

           MobileService.GetTable<Journeys>().UpdateAsync(j);

    }

    public static async Task<List<Journeys>> GetUsersJourneys(string userID)

    {

           CurrentPlatform.Init();

        IMobileServiceTable<Journeys> mTable = MobileService.GetTable<Journeys>();

           List<Journeys> items = await mTable

            .Where(x => x.CreatedBy == userID)

               .ToListAsync();

        if (items.Count > 0)

        {

            List<Journeys> result = new List<Journeys>();

            DateTime now = DateTime.Now;

            foreach (Journeys j in items)

            {

                DateTime dep = DateTime.Parse(j.DepartureDate);

                if (dep >= now)

                {

                       result.Add(j);

                }

            }

            return result;

        }

        else

        {

            return new List<Journeys>();

        }

    }

    public static async Task<List<Journeys>> GetUsersOldJourneys(string userID)

    {

           CurrentPlatform.Init();

           IMobileServiceTable<Journeys> mTable = MobileService.GetTable<Journeys>();

           List<Journeys> items = await mTable

            .Where(x => x.CreatedBy == userID)

               .ToListAsync();

        if (items.Count > 0)

        {

               List<Journeys> result = new List<Journeys>();

            DateTime now = DateTime.Now;

           foreach (Journeys j in items)

            {

                DateTime dep = DateTime.Parse(j.DepartureDate);

                if (dep < now)

                {

                       result.Add(j);

                }

           }

            return result;

        }

        else

        {

            return new List<Journeys>();

        }

    }

    public static async Task<List<Journeys>> GetUsersPassengerJourneys(string userID)

    {

           CurrentPlatform.Init();

           IMobileServiceTable<Journeys> mTable = MobileService.GetTable<Journeys>();

           List<Journeys> items = await mTable

            .Where(x => x.Passengers.Contains(userID))

            .ToListAsync();

        if (items.Count > 0)

        {

               List<Journeys> result = new List<Journeys>();

            DateTime now = DateTime.Now;

            foreach (Journeys j in items)

            {

                DateTime dep = DateTime.Parse(j.DepartureDate);

                if (dep >= now)

                {

                       result.Add(j);

                }

            }

            return result;

        }

        else

        {

            return new List<Journeys>();

        }

    }

    public static async Task<List<Journeys>> GetUsersOldPassengerJourneys(string userID)

    {

           CurrentPlatform.Init();

           IMobileServiceTable<Journeys> mTable = MobileService.GetTable<Journeys>();

           List<Journeys> items = await mTable

            .Where(x => x.Passengers.Contains(userID))

               .ToListAsync();

        if (items.Count > 0)

        {

               List<Journeys> result = new List<Journeys>();

            DateTime now = DateTime.Now;

            foreach (Journeys j in items)

            {

                DateTime dep = DateTime.Parse(j.DepartureDate);

                if (dep < now)

               {

                       result.Add(j);

                }

            }

            return result;

        }

        else

        {

            return new List<Journeys>();

        }

    }

    public static async Task<Journeys> GetJourneys(string ID)

    {

           CurrentPlatform.Init();

           IMobileServiceTable<Journeys> mTable = MobileService.GetTable<Journeys>();

           List<Journeys> items = await mTable

           .Where(x => x.ID == ID)

               .ToListAsync();

        if (items.Count > 0)

        {

            return items[0];

        }

        else

        {

            return new Journeys();

        }

    }

  public static async void InsertSearch(Search s)

    {

           CurrentPlatform.Init();

           MobileService.GetTable<Search>().InsertAsync(s);

    }

    public static async Task<List<Journeys>>SearchJourneys(Search search)

    {

        LatLng maxFrom = LocationHelper.getMaxLatLng(new LatLng(search.FromLat, search.FromLon), search.Range);

        LatLng minFrom = LocationHelper.getMinLatLng(new LatLng(search.FromLat, search.FromLon), search.Range);

        LatLng maxTo = LocationHelper.getMaxLatLng(new LatLng(search.ToLat, search.ToLon), search.Range);

        LatLng minTo = LocationHelper.getMinLatLng(new LatLng(search.ToLat, search.ToLon), search.Range);

           CurrentPlatform.Init();

        IMobileServiceTable<Journeys> mTable = MobileService.GetTable<Journeys>();

           List<Journeys> items = await mTable

            .Where(x => x.FromLat <= maxFrom.Latitude && x.FromLon <= maxFrom.Longitude

            && x.FromLat >= minFrom.Latitude && x.FromLon >= minFrom.Longitude

            && x.ToLat <= maxTo.Latitude && x.ToLon <= maxTo.Longitude

            && x.ToLat >= minTo.Latitude && x.ToLon >= minTo.Longitude

            && x.Completed == false && x.Filled == false)

           .ToListAsync();

        if (items.Count > 0)

        {

               List<Journeys> result = new List<Journeys>();

            DateTime depDate = DateTime.Parse(search.DepartureDate);

            DateTime minTime = DateTime.Parse(search.MinDepartureDateTime);

            DateTime maxTime = DateTime.Parse(search.MaxDepartureDateTime);

            foreach (Journeys j in items)

            {

                DateTime dep = DateTime.Parse(j.DepartureDate);

                if (dep == depDate && minTime <= DateTime.Parse(j.DepartureDateTime) && maxTime >= DateTime.Parse(j.DepartureDateTime))

                {

                       result.Add(j);

                }

            }

            return result;

        }

        else

        {

            return new List<Journeys>();

        }

    }

    public static async void SendApplicationEmail(string recipient, Journeys j)

    {

        string message = string.Format("Hey your application to CarShare was accepted! \nDetails: \nFrom: {0}\nTo: {1}\nOn: {2}\nAt: {3}\n\nLog In to the app to view more details!", j.From, j.To, j.DepartureDate, j.DepartureDateTime);

        var client = new SendGridClient(SendGridAPIKey);

       var msg = new SendGridMessage()

        {

            From = new EmailAddress("do-not-reply@carsharing.com", "CAIRDE - Application"),

            Subject = "CarSharing Journey Application",

               PlainTextContent = message,

               HtmlContent = "<strong>"+ message.Replace("\n","<br />") + "</strong>"

        };

        msg.AddTo(new EmailAddress(recipient, "User"));

        var response = await client.SendEmailAsync(msg);

    }

    public static async void SendApplicationNoticeEmail(string recipient, Journeys j)

    {

        string message = string.Format("Hey your journey on the {2} has new applicants. \n\nDetails: \nFrom: {0}\nTo: {1}\nOn: {2}\nAt: {3}\n\nLog In to the app to review the applicants!", j.From, j.To, j.DepartureDate, j.DepartureDateTime);

        var client = new SendGridClient(SendGridAPIKey);

        var msg = new SendGridMessage()

        {

            From = new EmailAddress("do-not-reply@carsharing.com", "CAIRDE - New Applicants"),

            Subject = "CarSharing Journey Application",

               PlainTextContent = message,

            HtmlContent = "<strong>" + message.Replace("\n", "<br />") + "</strong>"

        };

        msg.AddTo(new EmailAddress(recipient, "User"));

        var response = await client.SendEmailAsync(msg);

    }

    public static async Task<bool> SetupDemoData() //This method will populate the database with test users, and test journeys, old and new etc

    {

        Random rnd = new Random();

        List<Users> demoUsers = new List<Users>();

           List<UserProfiles> demoProfiles = new List<UserProfiles>();

        #region setupUsers

        string password = PasswordStorage.CreateHash("Carshar1ng");

        string[] userNames = new string[30] {"Dylan","Paul","Evelin","Zoe","Tommy",

           "Carl","Derek","Carol","Dean","Erica",

           "Roisin","Laura","Ken","Ron","Ross",

           "Barbie","Sean","Dan","Rory","Rachael",

           "Phoebe","Dolores","Leo","Lucifer","Carlos",

           "Renee","Zed","Damien","Nicole","David"};

        foreach (string s in userNames)

        {

            Users user = new Users() { Username = s, Password = password };

               InsertNewUser(user);

        }

        foreach (string s in userNames)

        {

            Users user = await GetUser(s);

               demoUsers.Add(user);

        }

        #endregion

        #region setup Profiles

        string lastName = "TestUser";

        string email = "carsharing.user@gmail.com";

        string phone = "0857584241";

        string[] gender = new string[3] { "Male", "Female", "Prefer not to Disclose" };

        string[] county = new string[6] { "Carlow", "Dublin", "Cork", "Galway","Waterford","Limerick" };

        foreach (Users u in demoUsers)

        {

            UserProfiles userProfile = new UserProfiles()

            {

                UsersID = u.ID,

                Firstname = u.Username,

                Lastname = lastName,

                Email = email,

                PhoneNo = phone,

                Gender = gender[rnd.Next(0, gender.Length)],

                County = county[rnd.Next(0, county.Length)]

            };

               InsertUserProfile(userProfile);

        }

        foreach (Users u in demoUsers)

        {

            UserProfiles up = await GetUsersProfile(u.ID);

               demoProfiles.Add(up);

        }

        #endregion

        #region Create Journeys

        int i;

        LatLng[] locations = new LatLng[6]

        {

            new LatLng(52.836438, -6.932914),

            new LatLng(53.347548, -6.259539),

            new LatLng(51.891959, -8.483876),

            new LatLng(53.269885, -9.056660),

            new LatLng(52.261530, -7.111773),

            new LatLng(52.667267, -8.631729)

       };

        DateTime[] depDates = new DateTime[11]

        {

               DateTime.Now.AddDays(-1),

            DateTime.Now,

               DateTime.Now.AddDays(1),

               DateTime.Now.AddDays(2),

            DateTime.Now.AddDays(3),

               DateTime.Now.AddDays(4),

               DateTime.Now.AddDays(5),

               DateTime.Now.AddDays(6),

               DateTime.Now.AddDays(7),

               DateTime.Now.AddDays(8),

               DateTime.Now.AddDays(9)

        };

        DateTime baseTime = new DateTime(2018, 4, 4, 7, 0, 0);

        DateTime[] depTimes = new DateTime[10]

        {

            baseTime,

               baseTime.AddMinutes(30),

               baseTime.AddMinutes(60),

               baseTime.AddMinutes(90),

               baseTime.AddMinutes(120),

               baseTime.AddMinutes(150),

               baseTime.AddMinutes(180),

               baseTime.AddMinutes(600),

               baseTime.AddMinutes(660),

          baseTime.AddMinutes(720)

        };

        for(i = 0; i < 1000 ; i++)

        {

            LatLng loc = locations[rnd.Next(0, 6)];

            LatLng des = locations[rnd.Next(0, 6)];

            string user = demoUsers[rnd.Next(0, demoUsers.Count)].ID;

            while (des.Latitude == loc.Latitude)

            {

                des = locations[rnd.Next(0, 6)];

            }

            Journeys randJourney = new Journeys()

            {

               CreatedBy = user,

                From = LocationHelper.ReverseGeoLoc(loc),

                To = LocationHelper.ReverseGeoLoc(des),

                FromLat = loc.Latitude,

                ToLat = des.Latitude,

               FromLon = loc.Longitude,

                ToLon = des.Longitude,

                DriverID = user,

                   NoOfPassengers = rnd.Next(1, 5),

                   DepartureDate = depDates[rnd.Next(0, depDates.Count())].ToShortDateString(),

                   DepartureDateTime = depTimes[rnd.Next(0, depTimes.Count())].ToShortTimeString(),

                Filled = false,

                Completed = false,

                Applicants = "",

                Passengers = ""

           };

               InsertJourney(randJourney);

        }

        #endregion

        return true;

    }

} }

# Journeys.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using Android.App;

using Android.Content;

using Android.OS;

using Android.Runtime;

using Android.Views;

using Android.Widget;

using Android.Gms.Maps.Model;

namespace CarShare.Models

{

public class Journeys

{

    public string ID { get; set; }

    public string CreatedBy { get; set; }

    public string From { get; set; }

    public string To { get; set; }

    public double FromLat { get; set; }

    public double FromLon { get; set; }

    public double ToLat { get; set; }

    public double ToLon { get; set; }

    public string DriverID { get; set; }

    public int NoOfPassengers { get; set; }

    public string Passengers { get; set; }

    public string DepartureDateTime { get; set; }

    public string DepartureDate { get; set; }

    public string Applicants { get; set; }

    public bool Filled { get; set; }

    public bool Completed { get; set; }

}

}

# LocationHelper.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using Android.App;

using Android.Content;

using Android.OS;

using Android.Runtime;

using Android.Views;

using Android.Widget;

using System.Xml;

using Android.Gms.Maps.Model;

using Android.Locations;

namespace CarShare.Models

{

public class LocationHelper : Activity

{

    const double pi = Math.PI;

    const double R = 6371.0;

    const double Lat2KmRatio = 0.008993215;

    const double Lon2KmRatio = 0.014607411;

    public static string ReverseGeoLoc(double latitude, double longitude)

    {

        XmlDocument doc = new XmlDocument();

        try

        {

               doc.Load("http://maps.googleapis.com/maps/api/geocode/xml?latlng=" + latitude + "," + longitude + "&sensor=false");

            XmlNode element = doc.SelectSingleNode("//GeocodeResponse/status");

            if (element.InnerText == "ZERO\_RESULTS")

            {

                return ("No data available for the specified location");

            }

            else if (element.InnerText != "OK")

            {

                return ReverseGeoLoc(latitude, longitude);

            }

            else

            {

                element = doc.SelectSingleNode("//GeocodeResponse/result/formatted\_address");

            }

            return (element.InnerText);

        }

        catch (Exception ex)

        {

            return ("(Address lookup failed: ) " + ex.Message);

        }

    }

    public static string ReverseGeoLoc(LatLng location)

    {

        double latitude = location.Latitude;

        double longitude = location.Longitude;

        XmlDocument doc = new XmlDocument();

        try

        {

               doc.Load("http://maps.googleapis.com/maps/api/geocode/xml?latlng=" + latitude + "," + longitude + "&sensor=false");

            XmlNode element = doc.SelectSingleNode("//GeocodeResponse/status");

            if (element.InnerText == "ZERO\_RESULTS")

            {

                return ("No data available for the specified location");

            }

            else if (element.InnerText != "OK")

            {

                return ReverseGeoLoc(latitude, longitude);

            }

            else

            {

                element = doc.SelectSingleNode("//GeocodeResponse/result/formatted\_address");

            }

            return (element.InnerText);

        }

        catch (Exception ex)

        {

            return ("(Address lookup failed: ) " + ex.Message);

        }

    }

    public static LatLng GetLatLngFromString(string l)

    {

        string latlng = l.Substring(l.IndexOf('('));

        latlng = latlng.TrimEnd(')');

        latlng = latlng.TrimStart('(');

        string[] laln = latlng.Split(',');

        return new LatLng(Convert.ToDouble(laln[0]), Convert.ToDouble(laln[1]));

    }

    public static double GetDistance(LatLng from, LatLng to)

    {

        double dLat = degree2Rad(to.Latitude - from.Latitude);  // deg2rad below

        double dLon = degree2Rad(to.Longitude - from.Longitude);

        double a =

            Math.Sin(dLat / 2) \* Math.Sin(dLat / 2) +

               Math.Cos(degree2Rad(from.Latitude)) \* Math.Cos(degree2Rad(to.Latitude)) \*

            Math.Sin(dLon / 2) \* Math.Sin(dLon / 2)

            ;

        double c = 2 \* Math.Atan2(Math.Sqrt(a), Math.Sqrt(1 - a));

        double d = R \* c; // Distance in km

        return d;

    }

    public static double GetDistance(string from, string to)

    {

        LatLng from2 = GetLatLngFromString(from);

        LatLng to2 = GetLatLngFromString(to);

        double dLat = degree2Rad(to2.Latitude - from2.Latitude);  // deg2rad below

        double dLon = degree2Rad(to2.Longitude - from2.Longitude);

        double a =

            Math.Sin(dLat / 2) \* Math.Sin(dLat / 2) +

               Math.Cos(degree2Rad(from2.Latitude)) \* Math.Cos(degree2Rad(to2.Latitude)) \*

            Math.Sin(dLon / 2) \* Math.Sin(dLon / 2)

            ;

        double c = 2 \* Math.Atan2(Math.Sqrt(a), Math.Sqrt(1 - a));

        double d = R \* c; // Distance in km

        return d;

    }

    public static double degree2Rad(double d)

    {

        return d \* (pi / 180);

    }

  public static LatLng getMaxLatLng(LatLng source, double distance)

    {

        double newLatitude = source.Latitude + (distance \* Lat2KmRatio);

        double newLongitude = source.Longitude + (distance \* Lon2KmRatio);

        return new LatLng(newLatitude, newLongitude);

    }

    public static LatLng getMinLatLng(LatLng source, double distance)

    {

        double newLatitude = source.Latitude - (distance \* Lat2KmRatio);

        double newLongitude = source.Longitude - (distance \* Lon2KmRatio);

        return new LatLng(newLatitude, newLongitude);

    }

}

}

# Search.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using Android.App;

using Android.Content;

using Android.OS;

using Android.Runtime;

using Android.Views;

using Android.Widget;

using Android.Gms.Maps.Model;

namespace CarShare.Models

{

public class Search

{

    public string ID { get; set; }

    public string CreatedBy { get; set; }

    public string From { get; set; }

    public string To { get; set; }

    public double FromLat { get; set; }

    public double FromLon { get; set; }

    public double ToLat { get; set; }

    public double ToLon { get; set; }

    public double Range { get; set; }

    public string MaxDepartureDateTime { get; set; }

    public string MinDepartureDateTime { get; set; }

    public string DepartureDate { get; set; }

    public string Matched { get; set; }

}

}

# 

# UserProfiles.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using Android.App;

using Android.Content;

using Android.OS;

using Android.Runtime;

using Android.Views;

using Android.Widget;

using Microsoft.WindowsAzure.MobileServices;

using System.Threading.Tasks;

namespace CarShare.Models

{

public class UserProfiles

{

    public string ID { get; set; }

    public string UsersID { get; set; }

    public string Firstname { get; set; }

    public string Lastname { get; set; }

    public string Email { get; set; }

    public string PhoneNo { get; set; }

    public string Gender { get; set; }

    public string County { get; set; }

}

}

# Users.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using Android.App;

using Android.Content;

using Android.OS;

using Android.Runtime;

using Android.Views;

using Android.Widget;

using Microsoft.WindowsAzure.MobileServices;

using System.Threading.Tasks;

namespace CarShare.Models

{

public class Users

{

    public string ID { get; set; }

    public string Username { get; set; }

    public string Password { get; set; }

}

}

# ApplicantsMenu.xml

<?xml version="1.0" encoding="utf-8" ?>

<menu xmlns:android="http://schemas.android.com/apk/res/android" xmlns:app="http://schemas.android.com/apk/res-auto">

 <item android:id="@+id/ApplicantAccept" android:title="Accept Applicant" showAsAction="always" />

 <item android:id="@+id/ApplicantRefuse" android:title="Refuse Applicant" showAsAction="always" />

</menu>

<!-- Code to implement into Activity:

Android.Widget.SearchView searchView;

public override bool OnCreateOptionsMenu(IMenu menu)

{

 this.MenuInflater.Inflate(Resource.Menu.ApplicantsMenu, menu);

 var searchItem = menu.FindItem(Resource.Id.action\_search);

 searchView = searchItem.ActionProvider.JavaCast<Android.Widget.SearchView>();

 searchView.QueryTextSubmit += (sender, args) =>

 {

Toast.MakeText(this, "You searched: " + args.Query, ToastLength.Short).Show();

 };

 return base.OnCreateOptionsMenu(menu);

}

-->

# CreateJourney.axml

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

   android:orientation="vertical"

   android:layout\_width="match\_parent"

   android:layout\_height="match\_parent"

   android:minWidth="25px"

   android:minHeight="25px"

android:scrollbars="vertical"

   android:isScrollContainer="true">

<FrameLayout

       android:minWidth="25px"

       android:minHeight="25px"

       android:layout\_width="match\_parent"

       android:layout\_height="300dip"

       android:id="@+id/mapFrame" />

<LinearLayout

       android:orientation="horizontal"

       android:layout\_width="match\_parent"

       android:layout\_height="wrap\_content"

       android:id="@+id/linearLayout1"

       android:weightSum="100">

    <Button

           android:text="Find Me"

           android:layout\_height="match\_parent"

           android:layout\_width="0dp"

           android:layout\_weight="25"

           android:id="@+id/buttonFindMe"

           android:backgroundTint="#ff0000ff" />

    <Button

           android:text="Set From"

           android:layout\_weight="25"

           android:layout\_height="match\_parent"

           android:layout\_width="0dp"

           android:id="@+id/buttonSetFrom"

           android:backgroundTint="#ff8b0000" />

    <Button

           android:text="set to"

           android:layout\_weight="25"

           android:layout\_height="match\_parent"

           android:layout\_width="0dp"

           android:id="@+id/buttonSetTo"

        android:backgroundTint="#ff008000" />

    <Button

           android:text="Save"

           android:layout\_weight="25"

           android:layout\_height="match\_parent"

           android:layout\_width="0dp"

           android:id="@+id/buttonSave" />

</LinearLayout>

<ScrollView

       android:layout\_width="match\_parent"

       android:layout\_height="match\_parent">

    <LinearLayout

           android:orientation="vertical"

           android:minWidth="25px"

        android:minHeight="25px"

           android:id="@+id/linearLayout6"

           android:layout\_width="match\_parent"

           android:layout\_height="wrap\_content">

        <LinearLayout

               android:orientation="horizontal"

               android:layout\_width="match\_parent"

               android:layout\_height="51.0dp"

               android:id="@+id/linearLayout2">

            <TextView

                   android:text="Number of Passengers: "

                   android:textAppearance="?android:attr/textAppearanceMedium"

                   android:layout\_width="wrap\_content"

                   android:layout\_height="match\_parent"

                   android:id="@+id/textViewNumOfPassengers"

                   android:layout\_gravity="center\_vertical"

                   android:gravity="center\_vertical" />

            <Spinner

                   android:layout\_width="match\_parent"

                   android:layout\_height="match\_parent"

                   android:id="@+id/spinnerNumOfPassengers"

                   android:layout\_gravity="center\_vertical"

                   android:textAlignment="center" />

           </LinearLayout>

        <LinearLayout

            android:orientation="vertical"

               android:layout\_width="match\_parent"

               android:layout\_height="wrap\_content"

               android:id="@+id/linearLayout3">

            <TextView

                   android:text="Departure Date: "

                   android:textAppearance="?android:attr/textAppearanceMedium"

                   android:layout\_width="wrap\_content"

                   android:layout\_height="match\_parent"

                   android:id="@+id/textViewDepDate" />

           <Button

                   android:text="Select Date"

                   android:layout\_width="match\_parent"

                   android:layout\_height="wrap\_content"

                   android:id="@+id/selectDateButton" />

        </LinearLayout>

        <LinearLayout

               android:orientation="vertical"

               android:layout\_width="match\_parent"

               android:layout\_height="wrap\_content"

               android:id="@+id/linearLayout7">

            <TextView

                   android:text="Departure Time: "

                   android:textAppearance="?android:attr/textAppearanceMedium"

                   android:layout\_width="wrap\_content"

                   android:layout\_height="match\_parent"

                   android:id="@+id/textViewDepTime" />

            <Button

                   android:text="Select Time"

                   android:layout\_width="match\_parent"

                   android:layout\_height="wrap\_content"

                   android:id="@+id/timeSelectButton" />

           </LinearLayout>

        <LinearLayout

               android:orientation="horizontal"

               android:layout\_width="match\_parent"

               android:layout\_height="51.0dp"

               android:id="@+id/linearLayout5">

            <TextView

                   android:text="Recurring: "

                   android:textAppearance="?android:attr/textAppearanceMedium"

                   android:layout\_width="wrap\_content"

                   android:layout\_height="match\_parent"

                   android:id="@+id/textViewReccuring"

                   android:gravity="center\_vertical" />

               <RadioButton

                   android:text="No"

               android:layout\_width="60.0dp"

                   android:layout\_height="match\_parent"

                   android:id="@+id/radioNo"

                   android:checked="true" />

               <RadioButton

                   android:text="Yes"

                   android:layout\_width="63.0dp"

                   android:layout\_height="match\_parent"

                   android:id="@+id/radioYes" />

           </LinearLayout>

        <LinearLayout

               android:orientation="vertical"

               android:layout\_width="match\_parent"

               android:layout\_height="wrap\_content"

               android:id="@+id/linearLayout8">

            <Button

                   android:text="Add Date"

                   android:layout\_width="match\_parent"

                   android:layout\_height="wrap\_content"

                   android:id="@+id/buttonAddDates"

                   android:visibility="invisible" />

            <TextView

                   android:textAppearance="?android:attr/textAppearanceMedium"

                   android:layout\_width="match\_parent"

                   android:layout\_height="wrap\_content"

                   android:id="@+id/textViewExtraDates" />

           </LinearLayout>

    </LinearLayout>

</ScrollView>

</LinearLayout>

# DriverJourneys.axml

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

   android:orientation="vertical"

   android:layout\_width="match\_parent"

   android:layout\_height="match\_parent">

<TextView

       android:text="Your Upcoming Journeys"

       android:textAppearance="?android:attr/textAppearanceMedium"

       android:layout\_width="match\_parent"

       android:layout\_height="31.0dp"

       android:id="@+id/textViewDriverJourneyPageTitle" />

<ListView

       android:minWidth="25px"

       android:minHeight="25px"

       android:layout\_width="match\_parent"

       android:layout\_height="547.0dp"

       android:id="@+id/listViewUpcomingJourneys"

       android:layout\_marginBottom="0.0dp"

       android:divider="@android:drawable/divider\_horizontal\_dark" />

<Button

       android:text="Back"

       android:layout\_width="match\_parent"

       android:layout\_height="wrap\_content"

       android:id="@+id/buttonDriverJourneyBack" />

</LinearLayout>

# DriverJourneys.axml

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

   android:orientation="vertical"

   android:layout\_width="match\_parent"

   android:layout\_height="match\_parent">

<TextView

       android:text="Your Upcoming Journeys"

       android:textAppearance="?android:attr/textAppearanceMedium"

       android:layout\_width="match\_parent"

       android:layout\_height="31.0dp"

       android:id="@+id/textViewDriverJourneyPageTitle" />

<ListView

       android:minWidth="25px"

       android:minHeight="25px"

       android:layout\_width="match\_parent"

       android:layout\_height="547.0dp"

       android:id="@+id/listViewUpcomingJourneys"

       android:layout\_marginBottom="0.0dp"

       android:divider="@android:drawable/divider\_horizontal\_dark" />

<Button

       android:text="Back"

       android:layout\_width="match\_parent"

       android:layout\_height="wrap\_content"

       android:id="@+id/buttonDriverJourneyBack" />

</LinearLayout>

# DriverOldJourneys.axml

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

   android:orientation="vertical"

   android:layout\_width="match\_parent"

   android:layout\_height="match\_parent">

 <TextView

  android:text="Your Old Journeys"

     android:textAppearance="?android:attr/textAppearanceMedium"

  android:layout\_width="match\_parent"

     android:layout\_height="31.0dp"

     android:id="@+id/textViewDriverOldJourneyPageTitle" />

 <ListView

     android:minWidth="25px"

     android:minHeight="25px"

     android:layout\_width="match\_parent"

  android:layout\_height="547.0dp"

     android:id="@+id/listViewOldDriverJourneys"

     android:layout\_marginBottom="0.0dp"

     android:divider="@android:drawable/divider\_horizontal\_dark" />

 <Button

     android:text="Back"

     android:layout\_width="match\_parent"

     android:layout\_height="wrap\_content"

     android:id="@+id/buttonDriverOldJourneyBack" />

</LinearLayout>

# JourneyMenu.xml

<?xml version="1.0" encoding="utf-8" ?>

<menu xmlns:android="http://schemas.android.com/apk/res/android" xmlns:app="http://schemas.android.com/apk/res-auto">

 <item android:id="@+id/pop\_button1" android:title="View Journey Detail" showAsAction="ifRoom|withText"/>

 <item android:id="@+id/pop\_button2" android:title="Cancel Journey" showAsAction="ifRoom|withText" />

</menu>

<!-- Code to implement into Activity:

Android.Widget.SearchView searchView;

public override bool OnCreateOptionsMenu(IMenu menu)

{

 this.MenuInflater.Inflate(Resource.Menu.JourneyMenu, menu);

 var searchItem = menu.FindItem(Resource.Id.action\_search);

 searchView = searchItem.ActionProvider.JavaCast<Android.Widget.SearchView>();

 searchView.QueryTextSubmit += (sender, args) =>

 {

Toast.MakeText(this, "You searched: " + args.Query, ToastLength.Short).Show();

 };

 return base.OnCreateOptionsMenu(menu);

}

-->

# Main.axml

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

   android:orientation="vertical"

   android:layout\_width="match\_parent"

   android:layout\_height="match\_parent"

   android:background="@android:color/holo\_blue\_light"

   android:weightSum="100"

   android:minWidth="25px"

   android:minHeight="25px">

<TextView

       android:text="Login"

       android:textAppearance="?android:attr/textAppearanceMedium"

       android:layout\_width="match\_parent"

       android:layout\_weight="20"

       android:layout\_height="0dp"

       android:id="@+id/textLogin"

       android:textColor="@android:color/black"

       android:gravity="center"

    android:textStyle="bold"

       android:textSize="50dp" />

<EditText

       android:layout\_width="350dp"

       android:layout\_weight="10"

       android:layout\_height="0dp"

       android:id="@+id/username"

       android:background="@android:color/darker\_gray"

       android:layout\_gravity="center"

       android:paddingLeft="10dp"

       android:layout\_marginTop="25dp"

       android:hint="Username" />

<EditText

       android:layout\_width="350dp"

       android:layout\_weight="10"

       android:layout\_height="0dp"

       android:id="@+id/passwordInput"

       android:background="@android:color/darker\_gray"

       android:layout\_gravity="center"

       android:paddingLeft="10dp"

       android:layout\_marginTop="25dp"

    android:password="true"

       android:hint="Password" />

<Button

       android:text="Login"

       android:layout\_width="350dp"

       android:layout\_height="0dp"

       android:id="@+id/loginButton"

       android:layout\_marginTop="25dp"

   android:layout\_gravity="center"

       android:layout\_weight="10"

       android:background="@android:color/holo\_blue\_dark"

       android:clickable="true" />

<Button

       android:text="Register"

       android:layout\_width="350dp"

    android:layout\_height="0dp"

       android:id="@+id/registerButton"

       android:layout\_marginTop="25dp"

       android:layout\_gravity="center"

       android:layout\_weight="10"

       android:background="@android:color/holo\_blue\_dark"

       android:clickable="true" />

 <Button

       android:text="Debugger"

       android:layout\_width="350dp"

       android:layout\_height="0dp"

       android:id="@+id/testButton"

       android:layout\_marginTop="25dp"

    android:layout\_gravity="center"

       android:layout\_weight="10"

       android:background="@android:color/holo\_blue\_dark"

       android:clickable="true" />

</LinearLayout>

# MainProfile.axml

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

   android:orientation="vertical"

   android:layout\_width="match\_parent"

   android:layout\_height="match\_parent"

   android:weightSum="100"

   android:minWidth="25px"

   android:minHeight="25px">

<ScrollView

  android:layout\_width="match\_parent"

       android:layout\_height="match\_parent">

    <LinearLayout

           android:orientation="vertical"

           android:layout\_width="match\_parent"

           android:layout\_height="match\_parent"

           android:weightSum="100"

           android:minWidth="25px"

           android:minHeight="25px">

        <LinearLayout

               android:orientation="vertical"

               android:minWidth="25px"

               android:minHeight="25px"

               android:layout\_width="match\_parent"

               android:layout\_height="wrap\_content"

               android:id="@+id/linearLayout1">

            <TextView

                   android:text="Welcome Back"

                android:textAppearance="?android:attr/textAppearanceMedium"

                   android:layout\_width="match\_parent"

                   android:layout\_height="wrap\_content"

                   android:id="@+id/textViewWelcome"

                android:layout\_marginTop="21.0dp"

                   android:layout\_marginBottom="15.0dp" />

           </LinearLayout>

        <ImageView

               android:src="@android:drawable/ic\_menu\_gallery"

               android:layout\_width="324.5dp"

           android:layout\_height="146.0dp"

               android:id="@+id/imageViewProfilePic"

               android:layout\_marginLeft="35.5dp" />

        <TextView

               android:text="Driver Options"

               android:textAppearance="?android:attr/textAppearanceMedium"

               android:layout\_width="match\_parent"

               android:layout\_height="wrap\_content"

               android:id="@+id/textViewDriverOptions" />

        <LinearLayout

               android:orientation="horizontal"

               android:layout\_width="match\_parent"

               android:layout\_height="wrap\_content"

               android:id="@+id/linearLayout2"

               android:weightSum="100">

               <Button

                   android:text="View My Upcoming Journeys"

                   android:layout\_height="match\_parent"

                   android:layout\_width="0dp"

                   android:id="@+id/buttonDriverUpcomingJourney"

               android:layout\_weight="33" />

            <Button

                   android:text="Create New Journey"

                   android:layout\_height="match\_parent"

                   android:layout\_width="0dp"

                android:id="@+id/buttonDriverCreateJourney"

                   android:layout\_weight="33" />

            <Button

                   android:text="View Completed Journeys"

                   android:layout\_height="match\_parent"

                   android:layout\_width="0dp"

                   android:id="@+id/buttonDriverCompletedJourney"

                   android:layout\_weight="33" />

           </LinearLayout>

        <TextView

               android:text="Passenger Options"

               android:textAppearance="?android:attr/textAppearanceMedium"

               android:layout\_width="match\_parent"

               android:layout\_height="wrap\_content"

               android:id="@+id/textViewPassengerOptions"

           android:layout\_marginTop="15.5dp" />

        <LinearLayout

               android:orientation="horizontal"

               android:layout\_width="match\_parent"

               android:layout\_height="wrap\_content"

               android:id="@+id/linearLayout3"

               android:weightSum="100">

            <Button

                   android:text="View My Upcoming Journeys"

                   android:layout\_height="match\_parent"

                   android:layout\_width="0dp"

               android:id="@+id/buttonPassengerUpcomingJourneys"

                   android:layout\_weight="33" />

            <Button

                   android:text="Search for new Lift"

                   android:layout\_height="match\_parent"

               android:layout\_width="0dp"

                   android:id="@+id/buttonPassengerSearchJourney"

                   android:layout\_weight="33" />

            <Button

                   android:text="View Completed Journeys"

                   android:layout\_height="match\_parent"

                   android:layout\_width="0dp"

                   android:id="@+id/buttonPassengerCompletedJourney"

                   android:layout\_weight="33" />

           </LinearLayout>

       <TextView

               android:text="Other"

               android:textAppearance="?android:attr/textAppearanceMedium"

               android:layout\_width="match\_parent"

               android:layout\_height="wrap\_content"

            android:id="@+id/textViewOther"

               android:layout\_marginTop="15.5dp" />

        <LinearLayout

               android:orientation="horizontal"

               android:minWidth="25px"

               android:minHeight="25px"

            android:layout\_width="match\_parent"

               android:layout\_height="wrap\_content"

               android:id="@+id/linearLayout4"

               android:weightSum="100">

            <Button

                   android:text="Log Out"

               android:layout\_width="wrap\_content"

                   android:layout\_height="match\_parent"

                   android:id="@+id/buttonLogOut" />

           </LinearLayout>

    </LinearLayout>

</ScrollView>

</LinearLayout>

# ProfileSetUp.axml

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

   android:orientation="vertical"

   android:layout\_width="match\_parent"

   android:layout\_height="match\_parent"

android:background="@android:color/holo\_blue\_light"

   android:weightSum="100"

   android:minWidth="25px"

   android:minHeight="25px">

<TextView

    android:text="Set Up Profile"

       android:textAppearance="?android:attr/textAppearanceMedium"

   android:layout\_width="match\_parent"

       android:layout\_weight="20"

       android:layout\_height="0dp"

       android:id="@+id/textRegister"

       android:textColor="@android:color/black"

       android:gravity="center"

    android:textStyle="bold"

       android:textSize="50dp" />

<EditText

       android:layout\_width="350dp"

       android:layout\_weight="10"

       android:layout\_height="0dp"

       android:id="@+id/firstName"

       android:background="@android:color/darker\_gray"

       android:layout\_gravity="center"

       android:paddingLeft="10dp"

       android:layout\_marginTop="15dp"

       android:hint="First Name"

       android:inputType="textPersonName" />

<EditText

       android:layout\_width="350dp"

       android:layout\_weight="10"

       android:layout\_height="0dp"

       android:id="@+id/lastName"

       android:background="@android:color/darker\_gray"

       android:layout\_gravity="center"

    android:paddingLeft="10dp"

       android:layout\_marginTop="15dp"

       android:hint="Last Name"

       android:inputType="textPersonName" />

<EditText

       android:layout\_width="350dp"

       android:layout\_weight="10"

    android:layout\_height="0dp"

       android:id="@+id/email"

       android:background="@android:color/darker\_gray"

       android:layout\_gravity="center"

       android:paddingLeft="10dp"

       android:layout\_marginTop="15dp"

       android:hint="Email"

       android:inputType="textEmailAddress" />

<EditText

       android:layout\_width="350dp"

       android:layout\_weight="10"

       android:layout\_height="0dp"

       android:id="@+id/phoneNo"

       android:background="@android:color/darker\_gray"

       android:layout\_gravity="center"

       android:paddingLeft="10dp"

       android:layout\_marginTop="15dp"

       android:hint="Mobile No."

       android:inputType="phone" />

<TextView

       android:layout\_width="fill\_parent"

    android:layout\_height="wrap\_content"

       android:layout\_marginTop="15dp"

       android:paddingLeft="10dp"

       android:text="@string/gender\_prompt" />

<Spinner

       android:layout\_width="350dp"

       android:layout\_weight="10"

    android:layout\_height="0dp"

       android:background="@android:color/darker\_gray"

       android:layout\_gravity="center"

       android:paddingLeft="10dp"

       android:prompt="@string/gender\_prompt"

       android:id="@+id/gender" />

<TextView

       android:layout\_width="fill\_parent"

       android:layout\_height="wrap\_content"

       android:layout\_marginTop="15dp"

       android:paddingLeft="10dp"

       android:text="@string/county\_prompt" />

<Spinner

       android:layout\_width="350dp"

       android:layout\_weight="10"

       android:layout\_height="0dp"

       android:background="@android:color/darker\_gray"

       android:layout\_gravity="center"

       android:paddingLeft="10dp"

       android:prompt="@string/county\_prompt"

       android:id="@+id/county" />

<Button

       android:text="Save Details"

       android:layout\_width="350dp"

       android:layout\_height="0dp"

       android:id="@+id/trySaveButton"

       android:layout\_marginTop="15dp"

    android:layout\_gravity="center"

       android:layout\_weight="10"

       android:background="@android:color/holo\_blue\_dark"

       android:clickable="true" />

</LinearLayout>

# Register.axml

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

   android:orientation="vertical"

   android:layout\_width="match\_parent"

   android:layout\_height="match\_parent"

   android:background="@android:color/holo\_blue\_light"

android:weightSum="100"

   android:minWidth="25px"

   android:minHeight="25px">

<TextView

       android:text="Register"

       android:textAppearance="?android:attr/textAppearanceMedium"

       android:layout\_width="match\_parent"

       android:layout\_weight="20"

       android:layout\_height="0dp"

       android:id="@+id/textRegister"

       android:textColor="@android:color/black"

       android:gravity="center"

       android:textStyle="bold"

       android:textSize="50dp" />

<EditText

    android:layout\_width="350dp"

       android:layout\_weight="10"

       android:layout\_height="0dp"

       android:id="@+id/usernameReg"

       android:background="@android:color/darker\_gray"

       android:layout\_gravity="center"

       android:paddingLeft="10dp"

       android:layout\_marginTop="25dp"

       android:hint="Username" />

<EditText

       android:layout\_width="350dp"

       android:layout\_weight="10"

       android:layout\_height="0dp"

    android:id="@+id/passwordInput1"

       android:background="@android:color/darker\_gray"

       android:layout\_gravity="center"

       android:paddingLeft="10dp"

       android:layout\_marginTop="25dp"

       android:password="true"

       android:hint="Password" />

   <EditText

       android:layout\_width="350dp"

       android:layout\_weight="10"

       android:layout\_height="0dp"

       android:id="@+id/passwordInput2"

       android:background="@android:color/darker\_gray"

       android:layout\_gravity="center"

       android:paddingLeft="10dp"

       android:layout\_marginTop="25dp"

       android:password="true"

       android:hint="Confirm Password" />

<Button

       android:text="Register"

       android:layout\_width="350dp"

       android:layout\_height="0dp"

       android:id="@+id/tryRegisterButton"

       android:layout\_marginTop="25dp"

       android:layout\_gravity="center"

       android:layout\_weight="10"

       android:background="@android:color/holo\_blue\_dark"

       android:clickable="true" />

<!-- Progress bar overlay; shown while login is in progress

<include

       layout="@layout/include\_progress\_overlay" />-->

</LinearLayout>

# SearchJourney.axml

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

   android:orientation="vertical"

   android:layout\_width="match\_parent"

   android:layout\_height="match\_parent"

   android:minWidth="25px"

   android:minHeight="25px"

   android:scrollbars="vertical"

   android:isScrollContainer="true">

<FrameLayout

       android:minWidth="25px"

       android:minHeight="25px"

       android:layout\_width="match\_parent"

       android:layout\_height="300dip"

       android:id="@+id/mapFrameSearch" />

<LinearLayout

       android:orientation="horizontal"

       android:layout\_width="match\_parent"

       android:layout\_height="wrap\_content"

       android:id="@+id/linearLayout1"

       android:weightSum="100">

    <Button

        android:text="Find Me"

           android:layout\_height="match\_parent"

           android:layout\_width="0dp"

           android:layout\_weight="25"

           android:id="@+id/buttonSearchJourneyMyLocation"

           android:backgroundTint="#ff0000ff" />

       <Button

           android:text="Set From"

           android:layout\_weight="25"

           android:layout\_height="match\_parent"

           android:layout\_width="0dp"

           android:id="@+id/buttonSetSearchJourneySetFrom"

        android:backgroundTint="#ff8b0000" />

    <Button

           android:text="set to"

           android:layout\_weight="25"

           android:layout\_height="match\_parent"

           android:layout\_width="0dp"

           android:id="@+id/buttonSearchJourneySetTo"

           android:backgroundTint="#ff008000" />

    <Button

           android:text="Search"

           android:layout\_weight="25"

           android:layout\_height="match\_parent"

           android:layout\_width="0dp"

           android:id="@+id/buttonSearchJourneySearch" />

</LinearLayout>

<ScrollView

       android:layout\_width="match\_parent"

       android:layout\_height="match\_parent">

    <LinearLayout

           android:orientation="vertical"

       android:minWidth="25px"

           android:minHeight="25px"

           android:id="@+id/linearLayout6"

           android:layout\_width="match\_parent"

           android:layout\_height="wrap\_content">

        <LinearLayout

            android:orientation="vertical"

               android:layout\_width="match\_parent"

               android:layout\_height="wrap\_content"

               android:id="@+id/linearLayout3">

            <TextView

                   android:text="Departure Date: "

                   android:textAppearance="?android:attr/textAppearanceMedium"

                   android:layout\_width="wrap\_content"

                   android:layout\_height="match\_parent"

                   android:id="@+id/textViewSearchJourneyDepDate" />

            <Button

                   android:text="Select Date"

                   android:layout\_width="match\_parent"

                   android:layout\_height="wrap\_content"

                   android:id="@+id/buttonSearchJourneySelectDate" />

           </LinearLayout>

        <LinearLayout

               android:orientation="vertical"

               android:layout\_width="match\_parent"

               android:layout\_height="wrap\_content"

               android:id="@+id/linearLayout7">

            <TextView

                   android:text="Departure Time: "

                   android:textAppearance="?android:attr/textAppearanceMedium"

                android:layout\_width="wrap\_content"

                   android:layout\_height="match\_parent"

                   android:id="@+id/textViewSearchJourneyDepTime" />

            <Button

                   android:text="Select Time"

                android:layout\_width="match\_parent"

                   android:layout\_height="wrap\_content"

                   android:id="@+id/buttonSearchJourneySelectTime" />

           </LinearLayout>

        <LinearLayout

               android:orientation="horizontal"

               android:layout\_width="match\_parent"

               android:layout\_height="51.0dp"

               android:id="@+id/linearLayout5">

            <TextView

                   android:text="Leaving: "

                   android:textAppearance="?android:attr/textAppearanceMedium"

                   android:layout\_width="wrap\_content"

                   android:layout\_height="match\_parent"

                   android:id="@+id/textViewSearchJourneyToggleTime"

                   android:gravity="center\_vertical" />

               <RadioButton

                   android:text="Before"

                   android:layout\_width="84.0dp"

                   android:layout\_height="match\_parent"

               android:id="@+id/radioSearchJourneyBefore"

                   android:checked="true" />

               <RadioButton

                   android:text="After"

                   android:layout\_width="80.0dp"

                   android:layout\_height="match\_parent"

                   android:id="@+id/radioSearchJourneyAfter" />

               <RadioButton

                   android:text="+/- 30min"

                   android:layout\_width="wrap\_content"

                   android:layout\_height="match\_parent"

                   android:id="@+id/radioSearchJourneyAround" />

           </LinearLayout>

        <LinearLayout

               android:orientation="horizontal"

               android:layout\_width="match\_parent"

            android:layout\_height="wrap\_content"

               android:id="@+id/linearLayout8">

            <TextView

                   android:text="Range:"

                   android:textAppearance="?android:attr/textAppearanceMedium"

                   android:layout\_width="wrap\_content"

                   android:layout\_height="match\_parent"

                   android:id="@+id/textViewSearchLabelRange"

                   android:gravity="center\_vertical" />

            <LinearLayout

                   android:orientation="vertical"

                   android:layout\_width="329.0dp"

                   android:layout\_height="40.0dp"

                   android:id="@+id/linearLayout9">

                   <TextView

                   android:text="1"

                       android:textAppearance="?android:attr/textAppearanceSmall"

                       android:layout\_width="match\_parent"

                       android:layout\_height="wrap\_content"

                    android:id="@+id/textViewSearchRange"

                       android:gravity="center\_horizontal" />

                   <SeekBar

                       android:layout\_width="244.5dp"

                       android:layout\_height="wrap\_content"

                   android:id="@+id/seekBarSearchRange"

                       android:layout\_marginLeft="41.5dp"

                       android:max="40"

                       android:progress="1" />

               </LinearLayout>

           </LinearLayout>

    </LinearLayout>

</ScrollView>

</LinearLayout>

# SearchJourneyMenu.axml

<?xml version="1.0" encoding="utf-8" ?>

<!--For all properties see: http://developer.android.com/guide/topics/resources/menu-resource.html-->

<menu xmlns:android="http://schemas.android.com/apk/res/android" xmlns:app="http://schemas.android.com/apk/res-auto">

 <item android:id="@+id/pop\_buttonSearchApply" android:title="Apply for lift" showAsAction="always" />

 <item android:id="@+id/pop\_buttonSearchView" android:title="View Journey Details" showAsAction="always" />

</menu>

<!-- Code to implement into Activity:

Android.Widget.SearchView searchView;

public override bool OnCreateOptionsMenu(IMenu menu)

{

 this.MenuInflater.Inflate(Resource.Menu.SearchJourneyMenu, menu);

 var searchItem = menu.FindItem(Resource.Id.action\_search);

 searchView = searchItem.ActionProvider.JavaCast<Android.Widget.SearchView>();

 searchView.QueryTextSubmit += (sender, args) =>

 {

Toast.MakeText(this, "You searched: " + args.Query, ToastLength.Short).Show();

 };

 return base.OnCreateOptionsMenu(menu);

}

-->

# SearchJourneyResults.axml

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

   android:orientation="vertical"

   android:layout\_width="match\_parent"

   android:layout\_height="match\_parent">

<TextView

       android:text="Search Results"

       android:textAppearance="?android:attr/textAppearanceMedium"

    android:layout\_width="match\_parent"

       android:layout\_height="31.0dp"

       android:id="@+id/textViewSearchJourneyResultsPageTitle" />

<ListView

       android:minWidth="25px"

       android:minHeight="25px"

       android:layout\_width="match\_parent"

       android:layout\_height="547.0dp"

       android:id="@+id/listViewSearchJourneyResults"

       android:layout\_marginBottom="0.0dp"

       android:divider="@android:drawable/divider\_horizontal\_dark" />

<Button

       android:text="Back to Search"

       android:layout\_width="match\_parent"

       android:layout\_height="wrap\_content"

       android:id="@+id/buttonSearchJourneyResultsBack" />

</LinearLayout>

# ShowJourney.axml

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

   android:orientation="vertical"

   android:layout\_width="match\_parent"

   android:layout\_height="match\_parent"

   android:minWidth="25px"

   android:minHeight="25px"

android:scrollbars="vertical"

   android:isScrollContainer="true">

<FrameLayout

       android:minWidth="25px"

       android:minHeight="25px"

       android:layout\_width="match\_parent"

       android:layout\_height="356.0dp"

    android:id="@+id/mapFrameShowJourney" />

 <ScrollView

           android:layout\_width="match\_parent"

           android:layout\_height="match\_parent"

           android:fillViewport="true">

<LinearLayout

       android:orientation="vertical"

       android:layout\_width="match\_parent"

       android:layout\_height="match\_parent"

       android:minWidth="25px"

       android:minHeight="25px">

        <TextView

               android:text="From:"

               android:textAppearance="?android:attr/textAppearanceMedium"

               android:layout\_width="match\_parent"

               android:layout\_height="wrap\_content"

               android:id="@+id/textViewShowJourneyFrom" />

        <TextView

            android:text="To:"

               android:textAppearance="?android:attr/textAppearanceMedium"

               android:layout\_width="match\_parent"

               android:layout\_height="wrap\_content"

               android:id="@+id/textViewShowJourneyTo" />

       <TextView

               android:text="Date:"

               android:textAppearance="?android:attr/textAppearanceMedium"

               android:layout\_width="match\_parent"

               android:layout\_height="wrap\_content"

            android:id="@+id/textViewShowJourneyDate" />

        <TextView

               android:text="Time:"

               android:textAppearance="?android:attr/textAppearanceMedium"

               android:layout\_width="match\_parent"

               android:layout\_height="wrap\_content"

               android:id="@+id/textViewShowJourneyTime" />

        <TextView

               android:text="Passengers: x/x"

               android:textAppearance="?android:attr/textAppearanceMedium"

               android:layout\_width="match\_parent"

               android:layout\_height="wrap\_content"

               android:id="@+id/textViewShowPassengerNumbers" />

        <ListView

               android:minWidth="25px"

            android:minHeight="25px"

               android:layout\_width="match\_parent"

               android:layout\_height="100.5dp"

               android:id="@+id/listViewShowJourneyPassengers" />

        <Button

               android:text="Check Applicants"

               android:layout\_width="match\_parent"

               android:layout\_height="wrap\_content"

               android:id="@+id/buttonShowJourneyApplicants" />

        <ListView

               android:minWidth="25px"

            android:minHeight="25px"

               android:layout\_width="match\_parent"

               android:layout\_height="400dp"

               android:id="@+id/listViewShowJourneyApplicants"

               android:visibility="invisible"

            android:divider="@android:drawable/divider\_horizontal\_dark" />

</LinearLayout>

 </ScrollView>

</LinearLayout>

# ViewOldPassengerJourneys.axml

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

   android:orientation="vertical"

   android:layout\_width="match\_parent"

   android:layout\_height="match\_parent">

 <TextView

  android:text="Your Upcoming Journeys"

     android:textAppearance="?android:attr/textAppearanceMedium"

  android:layout\_width="match\_parent"

     android:layout\_height="31.0dp"

     android:id="@+id/textViewPassengerOldJourneyPageTitle" />

 <ListView

     android:minWidth="25px"

     android:minHeight="25px"

     android:layout\_width="match\_parent"

  android:layout\_height="547.0dp"

     android:id="@+id/listViewPassengerOldJourney"

     android:layout\_marginBottom="0.0dp"

     android:divider="@android:drawable/divider\_horizontal\_dark" />

 <Button

     android:text="Back"

     android:layout\_width="match\_parent"

     android:layout\_height="wrap\_content"

     android:id="@+id/buttonPassengerOldJourneyBack" />

</LinearLayout>

# ViewUpcomingPassengerJourneys.axml

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

   android:orientation="vertical"

   android:layout\_width="match\_parent"

   android:layout\_height="match\_parent">

 <TextView

  android:text="Your Upcoming Journeys"

  android:textAppearance="?android:attr/textAppearanceMedium"

     android:layout\_width="match\_parent"

     android:layout\_height="31.0dp"

     android:id="@+id/textViewPassengerJourneyPageTitle" />

 <ListView

     android:minWidth="25px"

     android:minHeight="25px"

     android:layout\_width="match\_parent"

     android:layout\_height="547.0dp"

     android:id="@+id/listViewPassengerJourney"

     android:layout\_marginBottom="0.0dp"

     android:divider="@android:drawable/divider\_horizontal\_dark" />

 <Button

  android:text="Back"

     android:layout\_width="match\_parent"

     android:layout\_height="wrap\_content"

     android:id="@+id/buttonPassengerJourneyBack" />

</LinearLayout>

# CreateJourneyActivity.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using Android.App;

using Android.Content;

using Android.OS;

using Android.Runtime;

using Android.Views;

using Android.Widget;

using Android.Gms.Maps;

using Android.Locations;

using Android.Gms.Maps.Model;

using Android.Util;

using Android.Text.Format;

using CarShare.Models;

using Microsoft.WindowsAzure.MobileServices;

namespace CarShare

{

[Activity(Label = "CreateJourneyActivity")]

public class CreateJourneyActivity : Activity, IOnMapReadyCallback

{

    public static MobileServiceClient MobileService =

    new MobileServiceClient("https://c00197013.azurewebsites.net");

    static ISharedPreferences pref = Application.Context.GetSharedPreferences("UserInfo", FileCreationMode.Private);

       ISharedPreferencesEditor edit = pref.Edit();

    string UserID = pref.GetString("UserID", "");

    private GoogleMap \_map;

    private MapFragment \_mapFragment;

    LocationManager locMgr;

    MarkerOptions currentLocation;

       MarkerOptions fromLocation;

    MarkerOptions toLocation;

    Marker toMarker;

    Marker fromMarker;

    Spinner numPassengers;

    Button \_dateSelectButton;

    Button \_timeSelectButton;

    Button addDate;

    RadioButton notReccuring;

    RadioButton isRecurring;

    TextView depDate;

    TextView depTime;

    TextView additionalDates;

    List<DateTime> additionalDatesList;

    DateTime initialDepDate;

    DateTime initialDepTime;

    protected async override void OnCreate(Bundle savedInstanceState)

    {

           RequestWindowFeature(WindowFeatures.NoTitle);

           base.OnCreate(savedInstanceState);

           SetContentView(Resource.Layout.CreateJourney);

       #region setup map markers

        currentLocation = new MarkerOptions();

           currentLocation.SetTitle("Current Position");

           currentLocation.SetIcon(BitmapDescriptorFactory.DefaultMarker(BitmapDescriptorFactory.HueBlue));

       currentLocation.SetPosition(getCurrentPosition());

           currentLocation.Visible(false);

        fromLocation = new MarkerOptions();

           fromLocation.SetTitle("From Here");

           fromLocation.SetIcon(BitmapDescriptorFactory.DefaultMarker(BitmapDescriptorFactory.HueRed));

           fromLocation.SetPosition(getCurrentPosition());

           fromLocation.Visible(false);

        toLocation = new MarkerOptions();

           toLocation.SetTitle("To Here");

        toLocation.SetIcon(BitmapDescriptorFactory.DefaultMarker(BitmapDescriptorFactory.HueGreen));

           toLocation.SetPosition(getCurrentPosition());

           toLocation.Visible(false);

        #endregion

        // Create your application here

        InitMapFragment();

           SetupCurrentLocationButton();

           SetupSetLocationButton();

           SetupSetFromButton();

           SetupSetToButton();

        SetupForm();

    }

    protected override void OnResume()

    {

        base.OnResume();

           SetupMapIfNeeded();

    }

    private void SetupForm()

    {

        numPassengers = FindViewById<Spinner>(Resource.Id.spinnerNumOfPassengers);

        ArrayAdapter adapter = ArrayAdapter.CreateFromResource(

                this, Resource.Array.passengersNumber, Android.Resource.Layout.SimpleSpinnerItem);

           adapter.SetDropDownViewResource(Android.Resource.Layout.SimpleSpinnerDropDownItem);

           numPassengers.Adapter = adapter;

        depDate = FindViewById<TextView>(Resource.Id.textViewDepDate);

        \_dateSelectButton = FindViewById<Button>(Resource.Id.selectDateButton);

           \_dateSelectButton.Click += DateSelect\_OnClick;

        depTime = FindViewById<TextView>(Resource.Id.textViewDepTime);

        \_timeSelectButton = FindViewById<Button>(Resource.Id.timeSelectButton);

           \_timeSelectButton.Click += TimeSelect\_OnClick;

        notReccuring = FindViewById<RadioButton>(Resource.Id.radioNo);

        notReccuring.Click += notRecClicked;

        isRecurring = FindViewById<RadioButton>(Resource.Id.radioYes);

        isRecurring.Click += isRecClicked;

        addDate = FindViewById<Button>(Resource.Id.buttonAddDates);

        addDate.Click += addDateClicked;

        additionalDates = FindViewById<TextView>(Resource.Id.textViewExtraDates);

           additionalDatesList = new List<DateTime>();

    }

    private void addDateClicked(object sender, EventArgs e)

    {

        DatePickerFragment frag = DatePickerFragment.NewInstance(delegate (DateTime time)

        {

               additionalDates.Text += time.DayOfWeek.ToString() + " " + time.ToLongDateString() + "\n";

            additionalDatesList.Add(time);

        });

           frag.Show(FragmentManager, DatePickerFragment.TAG);

    }

    private void notRecClicked(object sender, EventArgs e)

    {

           notReccuring.Checked = true;

        isRecurring.Checked = false;

        addDate.Visibility = ViewStates.Invisible;

           additionalDates.Text = "";

           additionalDatesList.Clear();

    }

    private void isRecClicked(object sender, EventArgs e)

    {

           notReccuring.Checked = false;

           isRecurring.Checked = true;

        addDate.Visibility = ViewStates.Visible;

    }

    private void HandleDateChange(object sender, EventArgs e)

    {

        DatePicker dp = (DatePicker)sender;

        depDate = FindViewById<TextView>(Resource.Id.textViewDepDate);

        string date = dp.DateTime.DayOfWeek.ToString();

           depDate.SetText("Departure Date: " + date, TextView.BufferType.Normal);

    }

   private void SetupMapMarkers()

    {

           \_map.AddMarker(currentLocation);

           \_map.AddMarker(fromLocation);

           \_map.AddMarker(toLocation);

    }

    private void InitMapFragment()

    {

        \_mapFragment = FragmentManager.FindFragmentByTag("mapFrame") as MapFragment;

        if (\_mapFragment == null)

        {

               GoogleMapOptions mapOptions = new GoogleMapOptions()

                   .InvokeMapType(GoogleMap.MapTypeNormal)

               .InvokeZoomControlsEnabled(false)

                   .InvokeCompassEnabled(true)

                   .InvokeCamera(new CameraPosition(getCurrentPosition(), 18, 65, 0))

                ;

               FragmentTransaction fragTx = FragmentManager.BeginTransaction();

            \_mapFragment = MapFragment.NewInstance(mapOptions);

               fragTx.Add(Resource.Id.mapFrame, \_mapFragment, "mapFrame");

               fragTx.Commit();

        }

           \_mapFragment.GetMapAsync(this);

    }

    private void SetupCurrentLocationButton()

    {

        Button currentLocationButton = FindViewById<Button>(Resource.Id.buttonFindMe);

           currentLocationButton.Click += (sender, e) => {

               CameraPosition.Builder builder = CameraPosition.InvokeBuilder();

               builder.Target(getCurrentPosition());

               builder.Zoom(18);

               builder.Bearing(155);

            CameraPosition cameraPosition = builder.Build();

            // AnimateCamera provides a smooth, animation effect while moving

            // the camera to the the position.

               currentLocation.SetPosition(getCurrentPosition());

           currentLocation.Visible(true);

               \_map.AddMarker(currentLocation);

               \_map.AnimateCamera(CameraUpdateFactory.NewCameraPosition(cameraPosition));

        };

    }

    private void SetupSetLocationButton()

    {

        Button saveButton = FindViewById<Button>(Resource.Id.buttonSave);

        saveButton.Click += (sender, e) => {

            if (fromMarker.Position != null &&

               toMarker.Position != null &&

            initialDepDate != null &&

            initialDepTime != null)

            {

                   CreateJourney();

                   StartActivity(typeof(MainProfileActivity));

            }

            else

            {

                string message = "Please ensure you have selected all required Information \n " +

                "From Location \nTo Location \nNumber of available passenger slots \nDeparture Date and Time \n" +

                "\n Plus any additional dates in case of a recurring journey.";

                   Toast.MakeText(ApplicationContext, message, ToastLength.Long).Show();

            }

        };

    }

    private async void CreateJourney()

    {

        Journeys j = new Journeys();

        j.CreatedBy = UserID;

        j.DepartureDate = initialDepDate.ToShortDateString();

           j.DepartureDateTime = initialDepTime.ToShortTimeString();

        j.DriverID = UserID;

        j.From = LocationHelper.ReverseGeoLoc(fromMarker.Position.Latitude, fromMarker.Position.Longitude);

        j.FromLat = fromMarker.Position.Latitude;

        j.FromLon = fromMarker.Position.Longitude;

        j.To = LocationHelper.ReverseGeoLoc(toMarker.Position.Latitude, toMarker.Position.Longitude);

        j.ToLat = toMarker.Position.Latitude;

        j.ToLon = toMarker.Position.Longitude;

        j.NoOfPassengers = Convert.ToInt32(numPassengers.SelectedItem.ToString());

        if(DateTime.Now > initialDepDate)

        {

            j.Completed = true;

        }

        else

        {

            j.Completed = false;

        }

        j.Filled = false;

        j.Passengers = "";

        j.Applicants = "";

        ProgressDialog progress;

        progress = new ProgressDialog(this);

           progress.Indeterminate = true;

           progress.SetProgressStyle(ProgressDialogStyle.Spinner);

        progress.SetMessage("Creating... Please wait...");

           progress.SetCancelable(false);

        progress.Show();

           CurrentPlatform.Init();

           DBHelper.InsertJourney(j);

        if (additionalDatesList != null)

        {

            foreach (DateTime d in additionalDatesList)

            {

                   j.DepartureDate = d.ToShortDateString();

                   DBHelper.InsertJourney(j);

            }

        }

        progress.Hide();

       Toast.MakeText(ApplicationContext, "Journey created!", ToastLength.Short).Show();

    }

    private void SetupSetFromButton()

    {

        Button animateButton = FindViewById<Button>(Resource.Id.buttonSetFrom);

        animateButton.Click += (sender, e) => {

               fromLocation.SetPosition(\_map.CameraPosition.Target);

               fromLocation.Visible(true);

            if (fromMarker != null)

            {

                   fromMarker.Remove();

               }

            fromMarker = \_map.AddMarker(fromLocation);

        };

    }

    private void SetupSetToButton()

    {

        Button animateButton = FindViewById<Button>(Resource.Id.buttonSetTo);

           animateButton.Click += (sender, e) => {

               toLocation.SetPosition(\_map.CameraPosition.Target);

               toLocation.Visible(true);

            if (toMarker != null)

            {

                   toMarker.Remove();

            }

            toMarker = \_map.AddMarker(toLocation);

        };

    }

    private LatLng getCurrentPosition()

    {

        locMgr = GetSystemService(Context.LocationService) as LocationManager;

        string p = LocationManager.GpsProvider;

        Location l = locMgr.GetLastKnownLocation(p);

        if (l == null)

        {

            return new LatLng(52.8365, -6.9341);

        }

        return new LatLng(l.Latitude, l.Longitude);

       }

    public void OnMapReady(GoogleMap map)

    {

        \_map = map;

        SetupMapMarkers();

    }

    private void SetupMapIfNeeded()

    {

        if (\_map == null)

        {

            if (\_map != null)

            {

                   MarkerOptions markerOpt1 = new MarkerOptions();

                   markerOpt1.SetPosition(getCurrentPosition());

                   markerOpt1.SetTitle("Current Position");

               markerOpt1.InvokeIcon(BitmapDescriptorFactory.DefaultMarker(BitmapDescriptorFactory.HueCyan));

                   \_map.AddMarker(markerOpt1);

                // We create an instance of CameraUpdate, and move the map to it.

               CameraUpdate cameraUpdate = CameraUpdateFactory.NewLatLngZoom(getCurrentPosition(), 15);

                   \_map.MoveCamera(cameraUpdate);

            }

        }

    }

    void DateSelect\_OnClick(object sender, EventArgs eventArgs)

    {

        DatePickerFragment frag = DatePickerFragment.NewInstance(delegate (DateTime time)

        {

            depDate.Text = "Departure Date: " + time.DayOfWeek.ToString() + " " + time.ToLongDateString();

            initialDepDate = time;

        });

           frag.Show(FragmentManager, DatePickerFragment.TAG);

    }

    void TimeSelect\_OnClick(object sender, EventArgs eventArgs)

    {

        TimePickerFragment frag = TimePickerFragment.NewInstance(

            delegate (DateTime time)

            {

                   depTime.Text = "Departure Time: " + time.ToShortTimeString();

                   initialDepTime = time;

            });

        frag.Show(FragmentManager, TimePickerFragment.TAG);

    }

}

public class DatePickerFragment : DialogFragment,

                                 DatePickerDialog.IOnDateSetListener

{

    // TAG can be any string of your choice.

    public static readonly string TAG = "X:" + typeof(DatePickerFragment).Name.ToUpper();

    // Initialize this value to prevent NullReferenceExceptions.

    Action<DateTime> \_dateSelectedHandler = delegate { };

    public static DatePickerFragment NewInstance(Action<DateTime> onDateSelected)

    {

        DatePickerFragment frag = new DatePickerFragment();

           frag.\_dateSelectedHandler = onDateSelected;

        return frag;

    }

    public override Dialog OnCreateDialog(Bundle savedInstanceState)

    {

        DateTime currently = DateTime.Now;

        DatePickerDialog dialog = new DatePickerDialog(Activity,

                                                          this,

                                                          currently.Year,

                                                          currently.Month - 1,

                                                          currently.Day);

        return dialog;

    }

    public void OnDateSet(DatePicker view, int year, int monthOfYear, int dayOfMonth)

    {

        // Note: monthOfYear is a value between 0 and 11, not 1 and 12!

        DateTime selectedDate = new DateTime(year, monthOfYear + 1, dayOfMonth);

        Log.Debug(TAG, selectedDate.ToLongDateString());

           \_dateSelectedHandler(selectedDate);

    }

}

public class TimePickerFragment : DialogFragment, TimePickerDialog.IOnTimeSetListener

{

    public static readonly string TAG = "MyTimePickerFragment";

    Action<DateTime> timeSelectedHandler = delegate { };

    public static TimePickerFragment NewInstance(Action<DateTime> onTimeSelected)

    {

        TimePickerFragment frag = new TimePickerFragment();

           frag.timeSelectedHandler = onTimeSelected;

        return frag;

    }

    public override Dialog OnCreateDialog(Bundle savedInstanceState)

    {

        DateTime currentTime = DateTime.Now;

        bool is24HourFormat = DateFormat.Is24HourFormat(Activity);

        TimePickerDialog dialog = new TimePickerDialog

            (Activity, this, currentTime.Hour, currentTime.Minute, is24HourFormat);

       return dialog;

    }

    public void OnTimeSet(TimePicker view, int hourOfDay, int minute)

    {

        DateTime currentTime = DateTime.Now;

        DateTime selectedTime = new DateTime(currentTime.Year, currentTime.Month, currentTime.Day, hourOfDay, minute, 0);

        Log.Debug(TAG, selectedTime.ToLongTimeString());

           timeSelectedHandler(selectedTime);

    }

}

}

# DriverJourneysActivity.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using Android.App;

using Android.Content;

using Android.OS;

using Android.Runtime;

using Android.Views;

using Android.Widget;

using CarShare.Models;

using Android.Gms.Maps.Model;

using Android.Gms.Maps;

namespace CarShare

{

[Activity(Label = "DriverJourneysActivity")]

public class DriverJourneysActivity : Activity

{

    static ISharedPreferences pref = Application.Context.GetSharedPreferences("UserInfo", FileCreationMode.Private);

       ISharedPreferencesEditor edit = pref.Edit(); //out to session helper? future problem.

    ListView lv;

    List<Journeys> userJourneys;

    protected async override void OnCreate(Bundle savedInstanceState)

    {

           RequestWindowFeature(WindowFeatures.NoTitle);

           base.OnCreate(savedInstanceState);

           SetContentView(Resource.Layout.DriverJourneys);

        userJourneys = await DBHelper.GetUsersJourneys(pref.GetString("UserID", ""));

        string[] items;

        if (userJourneys.Count > 0)

        {

            items = GetJourneyDetails(userJourneys);

        }

        else

        {

            items = new string[1] { "No Upcoming Journeys" };

        }

        IListAdapter adapter = new ArrayAdapter<String>(this, Android.Resource.Layout.SimpleListItem1, items);

        lv = FindViewById<ListView>(Resource.Id.listViewUpcomingJourneys);

        lv.Adapter = adapter;

        lv.ItemClick += ListView\_ItemClick;

        Button backBtn = FindViewById<Button>(Resource.Id.buttonDriverJourneyBack);

        backBtn.Click += (sender, e) => { Finish(); };

        // Create your application here

    }

    private string[] GetJourneyDetails(List<Journeys> j)

    {

        List<string> items = new List<string>();

        foreach (Journeys i in j)

        {

            string appl = "";

               if(i.Applicants.Length > 10)

            {

                appl = "New Applicants!!!";

            }

            string d = Convert.ToDateTime(i.DepartureDate).DayOfWeek.ToString();

            string dd = Convert.ToDateTime(i.DepartureDate).Day.ToString();

            string m = Convert.ToDateTime(i.DepartureDate).Month.ToString();

            string y = Convert.ToDateTime(i.DepartureDate).Year.ToString();

            string from = i.From;

            string to = i.To;

            string details = "";

            if (appl == "")

           {

                details = String.Format("{0} {6}/{1}/{2} {3} \n\nFrom: {4} \nTo: {5}", d, m, y, i.DepartureDateTime, from, to, dd);

            }

            else

            {

                details = String.Format("{7}\n{0} {6}/{1}/{2} {3} \n\nFrom: {4} \nTo: {5}", d, m, y, i.DepartureDateTime, from, to, dd,appl);

            }

               items.Add(details);

        }

        return items.ToArray();

    }

    private void ListView\_ItemClick(object sender, AdapterView.ItemClickEventArgs e)

    {

        var menu = new PopupMenu(this, lv.GetChildAt(e.Position - lv.FirstVisiblePosition)); //lv.GetChildAt(e.Position)

           menu.Inflate(Resource.Layout.JourneyMenu);

        menu.MenuItemClick += (s, a) =>

        {

            switch (a.Item.ItemId)

            {

                case Resource.Id.pop\_button1:

                       Journeys j = userJourneys[e.Position];

                    edit.PutString("SelectedJourneyID", j.ID);

                       edit.Commit();

                       StartActivity(typeof(ShowJourney));

                    // goto details for selected journey

                    break;

                case Resource.Id.pop\_button2:

                    // cancel/delete this journey, if it has no passengers.

                    break;

            }

        };

        try

        {

            menu.Show();

        }

        catch (Exception ex)

        {

        }

    }

}

}

# DriverOldJourneysActivity.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using Android.App;

using Android.Content;

using Android.OS;

using Android.Runtime;

using Android.Views;

using Android.Widget;

using CarShare.Models;

using Android.Gms.Maps.Model;

using Android.Gms.Maps;

namespace CarShare

{

[Activity(Label = "DriverOldJourneysActivity")]

public class DriverOldJourneysActivity : Activity

{

    static ISharedPreferences pref = Application.Context.GetSharedPreferences("UserInfo", FileCreationMode.Private);

       ISharedPreferencesEditor edit = pref.Edit(); //out to session helper? future problem.

    ListView lv;

    List<Journeys> userJourneys;

    protected async override void OnCreate(Bundle savedInstanceState)

    {

           RequestWindowFeature(WindowFeatures.NoTitle);

           base.OnCreate(savedInstanceState);

           SetContentView(Resource.Layout.DriversOldJourneys);

        userJourneys = await DBHelper.GetUsersOldJourneys(pref.GetString("UserID", ""));

        string[] items;

        if (userJourneys.Count > 0)

        {

            items = GetJourneyDetails(userJourneys);

        }

        else

        {

            items = new string[1] { "No Old Journeys" };

        }

        IListAdapter adapter = new ArrayAdapter<String>(this, Android.Resource.Layout.SimpleListItem1, items);

        lv = FindViewById<ListView>(Resource.Id.listViewOldDriverJourneys);

        lv.Adapter = adapter;

        lv.ItemClick += ListView\_ItemClick;

        Button backBtn = FindViewById<Button>(Resource.Id.buttonDriverOldJourneyBack);

        backBtn.Click += (sender, e) => { Finish(); };

        // Create your application here

    }

    private string[] GetJourneyDetails(List<Journeys> j)

    {

        List<string> items = new List<string>();

        foreach (Journeys i in j)

        {

            string d = Convert.ToDateTime(i.DepartureDate).DayOfWeek.ToString();

            string dd = Convert.ToDateTime(i.DepartureDate).Day.ToString();

            string m = Convert.ToDateTime(i.DepartureDate).Month.ToString();

            string y = Convert.ToDateTime(i.DepartureDate).Year.ToString();

            string from = i.From;

            string to = i.To;

            string details = String.Format("{0} {6}/{1}/{2} {3} \n\nFrom: {4} \nTo: {5}\n\n", d, m, y, i.DepartureDateTime, from, to, dd);

               items.Add(details);

        }

        return items.ToArray();

    }

    private void ListView\_ItemClick(object sender, AdapterView.ItemClickEventArgs e)

    {

        var menu = new PopupMenu(this, lv.GetChildAt(e.Position));

           menu.Inflate(Resource.Layout.JourneyMenu);

        menu.MenuItemClick += (s, a) =>

        {

            switch (a.Item.ItemId)

            {

                case Resource.Id.pop\_button1:

                       Journeys j = userJourneys[e.Position];

                       edit.PutString("SelectedJourneyID", j.ID);

                       edit.Commit();

                   StartActivity(typeof(ShowJourney));

                    // goto details for selected journey

                    break;

                case Resource.Id.pop\_button2:

                    // cancel/delete this journey, if it has no passengers.

                    break;

            }

        };

        menu.Show();

    }

}

}

# MainActivity.cs

using Android.App;

using Android.Widget;

using Android.OS;

using Android.Views;

using System;

using System.Data.SqlClient;

using Microsoft.WindowsAzure.MobileServices;

using CarShare.Models;

using System.Collections.Generic;

using System.Threading.Tasks;

using System.Linq;

using Android.Content;

using Android.Gms.Maps;

using Android.Util;

namespace CarShare

{

[Activity(Label = "CarShare", MainLauncher = true)]

public class MainActivity : Activity

{

    static ISharedPreferences pref = Application.Context.GetSharedPreferences("UserInfo", FileCreationMode.Private);

       ISharedPreferencesEditor edit = pref.Edit(); //out to session helper? future problem.

    public const string TAG = "MainActivity";

    protected async override void OnCreate(Bundle savedInstanceState)

    {

           base.OnCreate(savedInstanceState);

           MapsInitializer.Initialize(ApplicationContext);

        // Set our view from the "main" layout resource

           SetContentView(Resource.Layout.Main);

        EditText username = (EditText)FindViewById(Resource.Id.username);

        EditText password = (EditText)FindViewById(Resource.Id.passwordInput);

        username.Text = pref.GetString("UserName", "");

        password.Text = pref.GetString("Password", "");

        var login = FindViewById(Resource.Id.loginButton);

        login.Click += TryLoginAsync;

           //if(pref.GetString("LoggedIn","false") == "true")

        //{

        //    StartActivity(typeof(SetUpProfileActivity));

        //}

        var register = FindViewById(Resource.Id.registerButton);

        register.Click += GoToRegister;

        var tester = FindViewById(Resource.Id.testButton);

        tester.Click += GoToTest;

    }

    private async void TryLoginAsync(object sender, EventArgs e)

    {

        ProgressDialog progress;

        progress = new Android.App.ProgressDialog(this);

           progress.Indeterminate = true;

           progress.SetProgressStyle(Android.App.ProgressDialogStyle.Spinner);

           progress.SetMessage("Logging In... Please wait...");

           progress.SetCancelable(false);

        progress.Show();

        EditText username = (EditText)FindViewById(Resource.Id.username);

        EditText password = (EditText)FindViewById(Resource.Id.passwordInput);

        string user = username.Text.Trim();

       string passwordInput = password.Text;

        if (!await DBHelper.DoesUserExist(user))

        {

               progress.Hide();

               Toast.MakeText(ApplicationContext, "Invalid Login", ToastLength.Short).Show();

       }

        else

        {

            Users u = await DBHelper.GetUser(user);

               if(PasswordStorage.VerifyPassword(passwordInput, u.Password))

            {

                   edit.PutString("UserID", u.ID);

                   edit.PutString("UserName", u.Username);

                   edit.PutString("LoggedIn", "true");

                   edit.PutString("Password", passwordInput);

                edit.Commit();

                   Toast.MakeText(ApplicationContext, "Success", ToastLength.Short).Show();

                if (await DBHelper.IsUserProfileCreated(u.ID))

                {

                    //go to main menu

                       UserProfiles up = await DBHelper.GetUsersProfile(u.ID);

                       edit.PutString("FirstName", up.Firstname);

                       edit.PutString("LastName", up.Lastname);

                    edit.Commit();

                       progress.Hide();

                       StartActivity(typeof(MainProfileActivity));

                }

                else

                {

                       progress.Hide();

                    StartActivity(typeof(SetUpProfileActivity));

                }

            }

            else

            {

                   progress.Hide();

                   Toast.MakeText(ApplicationContext, "Invalid Login", ToastLength.Short).Show();

            }

        }

    }

    private void GoToRegister(object sender, EventArgs e)

    {

           StartActivity(typeof(RegisterActivity));

    }

    private async void GoToTest(object sender, EventArgs e)

    {

        //await DBHelper.GetUsersPassengerJourneys("");

        //ProgressDialog progress;

        //progress = new Android.App.ProgressDialog(this);

           //progress.Indeterminate = true;

           //progress.SetProgressStyle(Android.App.ProgressDialogStyle.Spinner);

           //progress.SetMessage("Creating Demo Data...");

           //progress.SetCancelable(false);

        //progress.Show();

        //bool success = await DBHelper.SetupDemoData();

        //progress.Hide();

        //if (success)

        //{

        // Toast.MakeText(ApplicationContext, "Data Created", ToastLength.Short).Show();

        //}

    }

}

}

# MainProfileActivity.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using Android.App;

using Android.Content;

using Android.OS;

using Android.Runtime;

using Android.Views;

using Android.Widget;

using Android.Gms.Maps;

using Android.Locations;

using Android.Gms.Maps.Model;

using System.IO;

using Android.Graphics.Drawables;

namespace CarShare

{

[Activity(Label = "MainProfileActivity")]

public class MainProfileActivity : Activity

{

    static ISharedPreferences pref = Application.Context.GetSharedPreferences("UserInfo", FileCreationMode.Private);

       ISharedPreferencesEditor edit = pref.Edit(); //out to session helper? future problem.

    string firstName = pref.GetString("FirstName", "User");

    string lastName = pref.GetString("LastName", "");

    protected override void OnCreate(Bundle savedInstanceState)

    {

           RequestWindowFeature(WindowFeatures.NoTitle);

           base.OnCreate(savedInstanceState);

           SetContentView(Resource.Layout.MainProfile);

           SetupProfileImage();

        SetupButtons();

        // Create your application here

    }

  protected override void OnResume()

    {

           base.OnResume();

    }

    private void SetupButtons()

    {

        Button driverUpcoming = FindViewById<Button>(Resource.Id.buttonDriverUpcomingJourney);

           driverUpcoming.Click += DriverUpcoming\_Click;

        Button driverNewJourney = FindViewById<Button>(Resource.Id.buttonDriverCreateJourney);

           driverNewJourney.Click += DriverNewJourney\_Click;

        Button driverOldJourneys = FindViewById<Button>(Resource.Id.buttonDriverCompletedJourney);

           driverOldJourneys.Click += DriverOldJourneys\_Click;

        Button passengerUpcoming = FindViewById<Button>(Resource.Id.buttonPassengerUpcomingJourneys);

        passengerUpcoming.Click += PassengerUpcoming\_Click;

        Button passengerSearch = FindViewById<Button>(Resource.Id.buttonPassengerSearchJourney);

           passengerSearch.Click += PassengerSearch\_Click;

        Button passengerOldJourneys = FindViewById<Button>(Resource.Id.buttonPassengerCompletedJourney);

           passengerOldJourneys.Click += PassengerOldJourneys\_Click;

        Button logOut = FindViewById<Button>(Resource.Id.buttonLogOut);

        logOut.Click += LogOut\_Click;

    }

    private void LogOut\_Click(object sender, EventArgs e)

    {

           StartActivity(typeof(MainActivity));

    }

    private void PassengerOldJourneys\_Click(object sender, EventArgs e)

    {

           StartActivity(typeof(ViewOldPassengerJourneysActivity));

    }

    private void PassengerSearch\_Click(object sender, EventArgs e)

    {

           StartActivity(typeof(SearchJourney));

    }

    private void PassengerUpcoming\_Click(object sender, EventArgs e)

    {

           StartActivity(typeof(ViewPassengerJourneysActivity));

    }

    private void DriverOldJourneys\_Click(object sender, EventArgs e)

    {

        StartActivity(typeof(DriverOldJourneysActivity));

    }

    private void DriverNewJourney\_Click(object sender, EventArgs e)

    {

           StartActivity(typeof(CreateJourneyActivity));

    }

    private void DriverUpcoming\_Click(object sender, EventArgs e)

    {

           StartActivity(typeof(DriverJourneysActivity));

    }

    private void SetupProfileImage()

    {

        string profileIcon = GetProfileIcon(firstName);

        ImageView profileImage = FindViewById<ImageView>(Resource.Id.imageViewProfilePic);

        Stream ims = Assets.Open(profileIcon);

        Drawable d = Drawable.CreateFromStream(ims, null);

           profileImage.SetImageDrawable(d);

    }

    private string GetProfileIcon(string username)

    {

        char c = username.ToUpper()[0];

        return "profileIcons/" + c + ".png";

    }

}

}

# packages.config

<?xml version="1.0" encoding="utf-8"?>

<packages>

 <package id="Microsoft.AspNetCore.Http.Abstractions" version="1.1.0" targetFramework="monoandroid80" />

 <package id="Microsoft.AspNetCore.Http.Features" version="1.1.0" targetFramework="monoandroid80" />

 <package id="Microsoft.Azure.Mobile.Client" version="4.0.2" targetFramework="monoandroid80" />

 <package id="Microsoft.CSharp" version="4.4.1" targetFramework="monoandroid80" />

 <package id="Microsoft.Extensions.Primitives" version="1.1.0" targetFramework="monoandroid80" />

 <package id="Microsoft.NETCore.Platforms" version="2.0.1" targetFramework="monoandroid80" />

 <package id="NETStandard.Library" version="2.0.1" targetFramework="monoandroid80" />

 <package id="Newtonsoft.Json" version="11.0.2" targetFramework="monoandroid80" />

 <package id="PCLCrypto" version="2.0.147" targetFramework="monoandroid80" />

 <package id="PInvoke.BCrypt" version="0.5.111" targetFramework="monoandroid80" />

 <package id="PInvoke.Kernel32" version="0.5.111" targetFramework="monoandroid80" />

 <package id="PInvoke.NCrypt" version="0.5.111" targetFramework="monoandroid80" />

 <package id="PInvoke.Windows.Core" version="0.5.111" targetFramework="monoandroid80" />

 <package id="Sendgrid" version="9.9.0" targetFramework="monoandroid80" />

 <package id="System.ComponentModel" version="4.3.0" targetFramework="monoandroid80" />

 <package id="System.ComponentModel.TypeConverter" version="4.3.0" targetFramework="monoandroid80" />

 <package id="System.Globalization.Extensions" version="4.3.0" targetFramework="monoandroid80" />

 <package id="System.Net.WebSockets" version="4.3.0" targetFramework="monoandroid80" />

 <package id="System.Reflection.TypeExtensions" version="4.3.0" targetFramework="monoandroid80" />

 <package id="System.Runtime.CompilerServices.Unsafe" version="4.3.0" targetFramework="monoandroid80" />

 <package id="System.Runtime.Serialization.Formatters" version="4.3.0" targetFramework="monoandroid80" />

 <package id="System.Runtime.Serialization.Primitives" version="4.3.0" targetFramework="monoandroid80" />

 <package id="System.Security.Claims" version="4.3.0" targetFramework="monoandroid80" />

 <package id="System.Security.Principal" version="4.3.0" targetFramework="monoandroid80" />

 <package id="System.Text.Encodings.Web" version="4.3.0" targetFramework="monoandroid80" />

 <package id="System.Xml.XmlDocument" version="4.3.0" targetFramework="monoandroid80" />

 <package id="Validation" version="2.4.18" targetFramework="monoandroid80" />

 <package id="Xamarin.Android.Arch.Core.Common" version="1.0.0" targetFramework="monoandroid80" />

 <package id="Xamarin.Android.Arch.Lifecycle.Common" version="1.0.1" targetFramework="monoandroid80" />

 <package id="Xamarin.Android.Arch.Lifecycle.Runtime" version="1.0.0" targetFramework="monoandroid80" />

 <package id="Xamarin.Android.Support.Annotations" version="26.1.0.1" targetFramework="monoandroid80" />

 <package id="Xamarin.Android.Support.Compat" version="26.1.0.1" targetFramework="monoandroid80" />

 <package id="Xamarin.Android.Support.Core.UI" version="26.1.0.1" targetFramework="monoandroid80" />

 <package id="Xamarin.Android.Support.Core.Utils" version="26.1.0.1" targetFramework="monoandroid80" />

 <package id="Xamarin.Android.Support.CustomTabs" version="26.1.0.1" targetFramework="monoandroid80" />

 <package id="Xamarin.Android.Support.Fragment" version="26.1.0.1" targetFramework="monoandroid80" />

 <package id="Xamarin.Android.Support.Media.Compat" version="26.1.0.1" targetFramework="monoandroid80" />

 <package id="Xamarin.Android.Support.v4" version="26.1.0.1" targetFramework="monoandroid80" />

 <package id="Xamarin.Build.Download" version="0.4.9" targetFramework="monoandroid80" />

 <package id="Xamarin.GooglePlayServices.Base" version="60.1142.1" targetFramework="monoandroid80" />

 <package id="Xamarin.GooglePlayServices.Basement" version="60.1142.1" targetFramework="monoandroid80" />

 <package id="Xamarin.GooglePlayServices.Location" version="60.1142.1" targetFramework="monoandroid80" />

 <package id="Xamarin.GooglePlayServices.Maps" version="60.1142.1" targetFramework="monoandroid80" />

 <package id="Xamarin.GooglePlayServices.Tasks" version="60.1142.1" targetFramework="monoandroid80" />

</packages>

# PasswordStorage.cs

using System;

using System.Text;

using System.Security.Cryptography;

using Android.App;

using Android.Content;

using Android.OS;

using Android.Runtime;

using Android.Views;

using Android.Widget;

namespace CarShare

{

class InvalidHashException : Exception

{

  public InvalidHashException() { }

    public InvalidHashException(string message)

        : base(message) { }

    public InvalidHashException(string message, Exception inner)

        : base(message, inner) { }

}

class CannotPerformOperationException : Exception

{

    public CannotPerformOperationException() { }

    public CannotPerformOperationException(string message)

        : base(message) { }

    public CannotPerformOperationException(string message, Exception inner)

        : base(message, inner) { }

}

class PasswordStorage

{

    // These constants may be changed without breaking existing hashes.

    public const int SALT\_BYTES = 24;

    public const int HASH\_BYTES = 18;

    public const int PBKDF2\_ITERATIONS = 64000;

    // These constants define the encoding and may not be changed.

    public const int HASH\_SECTIONS = 5;

    public const int HASH\_ALGORITHM\_INDEX = 0;

    public const int ITERATION\_INDEX = 1;

    public const int HASH\_SIZE\_INDEX = 2;

    public const int SALT\_INDEX = 3;

    public const int PBKDF2\_INDEX = 4;

    public static string CreateHash(string password)

    {

        // Generate a random salt

        byte[] salt = new byte[SALT\_BYTES];

        try

        {

            using (RNGCryptoServiceProvider csprng = new RNGCryptoServiceProvider())

            {

                   csprng.GetBytes(salt);

            }

        }

        catch (CryptographicException ex)

        {

            throw new CannotPerformOperationException(

                   "Random number generator not available.",

                ex

            );

        }

      catch (ArgumentNullException ex)

        {

            throw new CannotPerformOperationException(

                   "Invalid argument given to random number generator.",

                ex

            );

        }

        byte[] hash = PBKDF2(password, salt, PBKDF2\_ITERATIONS, HASH\_BYTES);

        // format: algorithm:iterations:hashSize:salt:hash

        String parts = "sha1:" +

               PBKDF2\_ITERATIONS +

            ":" +

            hash.Length +

            ":" +

               Convert.ToBase64String(salt) +

            ":" +

               Convert.ToBase64String(hash);

        return parts;

    }

    public static bool VerifyPassword(string password, string goodHash)

    {

        char[] delimiter = { ':' };

        string[] split = goodHash.Split(delimiter);

        if (split.Length != HASH\_SECTIONS)

        {

            throw new InvalidHashException(

               "Fields are missing from the password hash."

            );

        }

        // We only support SHA1 with C#.

        if (split[HASH\_ALGORITHM\_INDEX] != "sha1")

        {

            throw new CannotPerformOperationException(

                   "Unsupported hash type."

            );

        }

        int iterations = 0;

        try

        {

            iterations = Int32.Parse(split[ITERATION\_INDEX]);

        }

        catch (ArgumentNullException ex)

        {

            throw new CannotPerformOperationException(

                   "Invalid argument given to Int32.Parse",

                ex

            );

        }

        catch (FormatException ex)

        {

            throw new InvalidHashException(

                   "Could not parse the iteration count as an integer.",

                ex

            );

        }

        catch (OverflowException ex)

      {

            throw new InvalidHashException(

                "The iteration count is too large to be represented.",

                ex

            );

        }

        if (iterations < 1)

        {

            throw new InvalidHashException(

                   "Invalid number of iterations. Must be >= 1."

            );

        }

        byte[] salt = null;

        try

        {

            salt = Convert.FromBase64String(split[SALT\_INDEX]);

        }

        catch (ArgumentNullException ex)

        {

            throw new CannotPerformOperationException(

                   "Invalid argument given to Convert.FromBase64String",

                ex

          );

        }

        catch (FormatException ex)

        {

            throw new InvalidHashException(

                   "Base64 decoding of salt failed.",

                ex

            );

        }

        byte[] hash = null;

        try

        {

            hash = Convert.FromBase64String(split[PBKDF2\_INDEX]);

        }

        catch (ArgumentNullException ex)

        {

            throw new CannotPerformOperationException(

                   "Invalid argument given to Convert.FromBase64String",

                ex

            );

        }

        catch (FormatException ex)

        {

            throw new InvalidHashException(

                "Base64 decoding of pbkdf2 output failed.",

                ex

            );

        }

        int storedHashSize = 0;

        try

        {

            storedHashSize = Int32.Parse(split[HASH\_SIZE\_INDEX]);

        }

       catch (ArgumentNullException ex)

        {

            throw new CannotPerformOperationException(

                   "Invalid argument given to Int32.Parse",

                ex

            );

        }

        catch (FormatException ex)

        {

            throw new InvalidHashException(

                   "Could not parse the hash size as an integer.",

                ex

            );

        }

        catch (OverflowException ex)

        {

            throw new InvalidHashException(

                "The hash size is too large to be represented.",

                ex

            );

        }

        if (storedHashSize != hash.Length)

       {

            throw new InvalidHashException(

                "Hash length doesn't match stored hash length."

            );

        }

        byte[] testHash = PBKDF2(password, salt, iterations, hash.Length);

        return SlowEquals(hash, testHash);

    }

    private static bool SlowEquals(byte[] a, byte[] b)

    {

        uint diff = (uint)a.Length ^ (uint)b.Length;

        for (int i = 0; i < a.Length && i < b.Length; i++)

        {

            diff |= (uint)(a[i] ^ b[i]);

        }

        return diff == 0;

    }

    private static byte[] PBKDF2(string password, byte[] salt, int iterations, int outputBytes)

    {

        using (Rfc2898DeriveBytes pbkdf2 = new Rfc2898DeriveBytes(password, salt))

        {

               pbkdf2.IterationCount = iterations;

            return pbkdf2.GetBytes(outputBytes);

        }

    }

}

}

# RegisterActivity.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using Android.App;

using Android.Content;

using Android.OS;

using Android.Runtime;

using Android.Views;

using Android.Widget;

using System.Data.SqlClient;

using System.Text.RegularExpressions;

using Android.Text;

using Microsoft.WindowsAzure.MobileServices;

using CarShare.Models;

namespace CarShare

{

[Activity(Label = "RegisterActivity")]

public class RegisterActivity : Activity

{

    public static MobileServiceClient MobileService =

    new MobileServiceClient("https://c00197013.azurewebsites.net");

    protected async override void OnCreate(Bundle savedInstanceState)

    {

           base.OnCreate(savedInstanceState);

           SetContentView(Resource.Layout.Register);

       // Create your application here

        var register = FindViewById(Resource.Id.tryRegisterButton);

        register.Click += TryRegister;

        EditText username = (EditText)FindViewById(Resource.Id.usernameReg);

        username.TextChanged += ValidateUserInput;

        EditText password = (EditText)FindViewById(Resource.Id.passwordInput1);

           password.TextChanged += ValidateUserInput;

        EditText confirmPass = (EditText)FindViewById(Resource.Id.passwordInput2);

           confirmPass.TextChanged += ValidateUserInput;

    }

    private async void TryRegister(object sender, EventArgs e)

    {

        ProgressDialog progress;

        progress = new Android.App.ProgressDialog(this);

           progress.Indeterminate = true;

           progress.SetProgressStyle(Android.App.ProgressDialogStyle.Spinner);

           progress.SetMessage("Registering... Please wait...");

           progress.SetCancelable(false);

        progress.Show();

        EditText username = (EditText)FindViewById(Resource.Id.usernameReg);

        EditText password = (EditText)FindViewById(Resource.Id.passwordInput1);

        EditText confirmPassword = (EditText)FindViewById(Resource.Id.passwordInput2);

        if (password.Text == confirmPassword.Text)

        {

            string hashedPassword = PasswordStorage.CreateHash(password.Text);

            string userName = username.Text.Trim();

               //CurrentPlatform.Init();

            Users newUser = new Users { Username = userName, Password = hashedPassword };

               List<Users> allUsers = await MobileService.GetTable<Users>().ToListAsync();

            Users u = allUsers.FirstOrDefault(x => x.Username == newUser.Username);

            if (u == null)

            {

                   DBHelper.InsertNewUser(newUser);

                   //MobileService.GetTable<Users>().InsertAsync(newUser);

                   progress.Hide();

                   Toast.MakeText(ApplicationContext, "User " + newUser.Username + " created! You can now log in!", ToastLength.Short).Show();

                   StartActivity(typeof(MainActivity));

            }

            else

            {

                   progress.Hide();

                   Toast.MakeText(ApplicationContext, "User " + u.Username + " already exists!", ToastLength.Short).Show();

            }

        }

        else

        {

           progress.Hide();

            string message = "Passwords don't match.";

               Toast.MakeText(ApplicationContext, message, ToastLength.Short).Show();

        }

    }

    private void ValidateUserInput(object sender, TextChangedEventArgs e)

    {

        EditText input = (EditText)sender;

        string pattern = "[^a-zA-Z0-9#?]";

        if (Regex.IsMatch(input.Text, pattern))

        {

            string message = "Input must only contain alphabetic characters, numbers, ? and #";

               Toast.MakeText(ApplicationContext, message, ToastLength.Short).Show();

            input.Text = "";

        }

    }

}

}

# SearchJourney.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using Android.App;

using Android.Content;

using Android.OS;

using Android.Runtime;

using Android.Views;

using Android.Widget;

using Android.Gms.Maps;

using Android.Locations;

using Android.Gms.Maps.Model;

using Android.Util;

using Android.Text.Format;

using CarShare.Models;

using Microsoft.WindowsAzure.MobileServices;

using Android.Graphics;

using Newtonsoft.Json;

namespace CarShare

{

[Activity(Label = "SearchJourney")]

public class SearchJourney : Activity, IOnMapReadyCallback

{

    static ISharedPreferences pref = Application.Context.GetSharedPreferences("UserInfo", FileCreationMode.Private);

       ISharedPreferencesEditor edit = pref.Edit();

    string UserID = pref.GetString("UserID", "");

    private GoogleMap \_map;

    private MapFragment \_mapFragment;

    LocationManager locMgr;

    MarkerOptions currentLocation;

    MarkerOptions fromLocation;

    MarkerOptions toLocation;

    CircleOptions fromCircleOp;

    CircleOptions toCircleOp;

    Marker toMarker;

    Marker fromMarker;

    Circle fromCircle;

    Circle toCircle;

    Button \_dateSelectButton;

    Button \_timeSelectButton;

    RadioButton before;

    RadioButton after;

    RadioButton around;

    TextView depDate;

    TextView depTime;

    TextView range;

    SeekBar rangeFinder;

    DateTime initialDepDate;

    DateTime initialDepTime;

    DateTime minDepTime;

    DateTime maxDepTime;

    protected override void OnCreate(Bundle savedInstanceState)

    {

           RequestWindowFeature(WindowFeatures.NoTitle);

           base.OnCreate(savedInstanceState);

        SetContentView(Resource.Layout.SearchJourney);

        #region setup map markers

        currentLocation = new MarkerOptions();

           currentLocation.SetTitle("Current Position");

           currentLocation.SetIcon(BitmapDescriptorFactory.DefaultMarker(BitmapDescriptorFactory.HueBlue));

           currentLocation.SetPosition(getCurrentPosition());

           currentLocation.Visible(false);

        fromLocation = new MarkerOptions();

           fromLocation.SetTitle("From Here");

       fromLocation.SetIcon(BitmapDescriptorFactory.DefaultMarker(BitmapDescriptorFactory.HueRed));

           fromLocation.SetPosition(getCurrentPosition());

           fromLocation.Visible(false);

        toLocation = new MarkerOptions();

       toLocation.SetTitle("To Here");

           toLocation.SetIcon(BitmapDescriptorFactory.DefaultMarker(BitmapDescriptorFactory.HueGreen));

           toLocation.SetPosition(getCurrentPosition());

           toLocation.Visible(false);

        fromCircleOp = new CircleOptions();

           fromCircleOp.InvokeCenter(getCurrentPosition());

           fromCircleOp.InvokeRadius(1000);

           fromCircleOp.InvokeFillColor(0X66FF0000);

           fromCircleOp.InvokeStrokeColor(0X66FF0000);

           fromCircleOp.InvokeStrokeWidth(0);

           fromCircleOp.Visible(false);

        toCircleOp = new CircleOptions();

           toCircleOp.InvokeCenter(getCurrentPosition());

           toCircleOp.InvokeRadius(1000);

           toCircleOp.InvokeFillColor(Color.Green);

           toCircleOp.InvokeStrokeColor(Color.Green);

           toCircleOp.InvokeStrokeWidth(0);

           toCircleOp.Visible(false);

        #endregion

        // Create your application here

        InitMapFragment();

           SetupCurrentLocationButton();

           SetupSearchButton();

           SetupSetFromButton();

           SetupSetToButton();

        SetupForm();

    }

    protected override void OnResume()

    {

        base.OnResume();

           SetupMapIfNeeded();

    }

    private void SetupForm()

    {

        depDate = FindViewById<TextView>(Resource.Id.textViewSearchJourneyDepDate);

        \_dateSelectButton = FindViewById<Button>(Resource.Id.buttonSearchJourneySelectDate);

           \_dateSelectButton.Click += DateSelect\_OnClick;

        depTime = FindViewById<TextView>(Resource.Id.buttonSearchJourneySelectTime);

        \_timeSelectButton = FindViewById<Button>(Resource.Id.buttonSearchJourneySelectTime);

           \_timeSelectButton.Click += TimeSelect\_OnClick;

        before = FindViewById<RadioButton>(Resource.Id.radioSearchJourneyBefore);

        before.Click += beforeClicked;

        after = FindViewById<RadioButton>(Resource.Id.radioSearchJourneyAfter);

        after.Click += afterClicked;

        around = FindViewById<RadioButton>(Resource.Id.radioSearchJourneyAround);

        around.Click += aroundClicked;

        range = FindViewById<TextView>(Resource.Id.textViewSearchRange);

        rangeFinder = FindViewById<SeekBar>(Resource.Id.seekBarSearchRange);

        range.Text = string.Format("{0}km", rangeFinder.Progress);

        rangeFinder.ProgressChanged += (object sender, SeekBar.ProgressChangedEventArgs e) => {

            if (e.FromUser)

            {

                range.Text = string.Format("{0}km", (double)e.Progress/2);

                   toCircle.Radius = e.Progress \* 500;

                   fromCircle.Radius = e.Progress \* 500;

            }

        };

    }

    private void beforeClicked(object sender, EventArgs e)

    {

        after.Checked = false;

        around.Checked = false;

        maxDepTime = initialDepTime;

        minDepTime = initialDepTime.AddHours(-1);

    }

    private void afterClicked(object sender, EventArgs e)

    {

        before.Checked = false;

        around.Checked = false;

        maxDepTime = initialDepTime.AddHours(1); ;

        minDepTime = initialDepTime;

    }

    private void aroundClicked(object sender, EventArgs e)

    {

        after.Checked = false;

       before.Checked = false;

        maxDepTime = initialDepTime.AddMinutes(30);

        minDepTime = initialDepTime.AddMinutes(-30);

    }

    private void HandleDateChange(object sender, EventArgs e)

    {

        DatePicker dp = (DatePicker)sender;

        depDate = FindViewById<TextView>(Resource.Id.textViewSearchJourneyDepDate);

        string date = dp.DateTime.DayOfWeek.ToString();

           depDate.SetText("Departure Date: " + date, TextView.BufferType.Normal);

    }

    private void SetupMapMarkers()

    {

           \_map.AddMarker(currentLocation);

           \_map.AddMarker(fromLocation);

           \_map.AddMarker(toLocation);

        fromCircle = \_map.AddCircle(fromCircleOp);

        toCircle =\_map.AddCircle(toCircleOp);

    }

    private void InitMapFragment()

    {

        \_mapFragment = FragmentManager.FindFragmentByTag("mapFrameSearch") as MapFragment;

        if (\_mapFragment == null)

      {

               GoogleMapOptions mapOptions = new GoogleMapOptions()

                   .InvokeMapType(GoogleMap.MapTypeNormal)

                   .InvokeZoomControlsEnabled(false)

                   .InvokeCompassEnabled(true)

               .InvokeCamera(new CameraPosition(getCurrentPosition(), 18, 65, 0))

                ;

               FragmentTransaction fragTx = FragmentManager.BeginTransaction();

            \_mapFragment = MapFragment.NewInstance(mapOptions);

               fragTx.Add(Resource.Id.mapFrameSearch, \_mapFragment, "mapFrameSearch");

               fragTx.Commit();

        }

           \_mapFragment.GetMapAsync(this);

    }

    private void SetupCurrentLocationButton()

    {

       Button currentLocationButton = FindViewById<Button>(Resource.Id.buttonSearchJourneyMyLocation);

           currentLocationButton.Click += (sender, e) => {

               CameraPosition.Builder builder = CameraPosition.InvokeBuilder();

           builder.Target(getCurrentPosition());

               builder.Zoom(18);

               builder.Bearing(155);

            CameraPosition cameraPosition = builder.Build();

            // AnimateCamera provides a smooth, animation effect while moving

            // the camera to the the position.

               currentLocation.SetPosition(getCurrentPosition());

               currentLocation.Visible(true);

               \_map.AddMarker(currentLocation);

               \_map.AnimateCamera(CameraUpdateFactory.NewCameraPosition(cameraPosition));

        };

    }

    private void SetupSearchButton()

    {

        Button saveButton = FindViewById<Button>(Resource.Id.buttonSearchJourneySearch);

        saveButton.Click += async (sender, e) => {

            if (fromMarker.Position != null &&

               toMarker.Position != null &&

            initialDepDate != null &&

            initialDepTime != null)

            {

              Search userSearch = new Search()

                {

                       CreatedBy = pref.GetString("UserID", ""),

                       DepartureDate = initialDepDate.ToShortDateString(),

                       MaxDepartureDateTime = maxDepTime.ToShortTimeString(),

                       MinDepartureDateTime = minDepTime.ToShortTimeString(),

                    From = LocationHelper.ReverseGeoLoc(fromMarker.Position.Latitude, fromMarker.Position.Longitude),

                   To = LocationHelper.ReverseGeoLoc(toMarker.Position.Latitude, toMarker.Position.Longitude),

                       FromLat = fromMarker.Position.Latitude,

                       FromLon = fromMarker.Position.Longitude,

                    ToLat = toMarker.Position.Latitude,

                    ToLon = toMarker.Position.Longitude,

                    Range = 20

                };

                //Get a list of Journeys (not expired) in the right area (distance = 20km for now)

              List<Journeys> candidates = await DBHelper.SearchJourneys(userSearch);

                //handle search

                if (candidates.Count == 0)

                {

                    //ask if user wants to list this search

                }

                else

                {

                    //show results

                    string saveSearchResults = JsonConvert.SerializeObject(candidates);

                       edit.PutString("SearchResults",saveSearchResults);

                       edit.Commit();

                       StartActivity(typeof(SearchJourneyResultsActivity));

                }

                string message = "Number of hits: " + candidates.Count;

                   Toast.MakeText(ApplicationContext,message, ToastLength.Short).Show();

                   //StartActivity(typeof(MainProfileActivity));

            }

            else

            {

               string message = "Please ensure you have selected all required Information \n " +

                "From Location \nTo Location \nNumber of available passenger slots \nDeparture Date and Time \n" +

                "\n Plus any additional dates in case of a recurring journey.";

                   Toast.MakeText(ApplicationContext, message, ToastLength.Long).Show();

            }

        };

    }

    private async void CreateJourney()

    {

        Journeys j = new Journeys();

        j.CreatedBy = UserID;

        j.DepartureDate = initialDepDate.ToShortDateString();

           j.DepartureDateTime = initialDepTime.ToShortTimeString();

        j.DriverID = UserID;

        j.From = LocationHelper.ReverseGeoLoc(fromMarker.Position.Latitude, fromMarker.Position.Longitude);

        j.FromLat = fromMarker.Position.Latitude;

        j.FromLon = fromMarker.Position.Longitude;

        j.To = LocationHelper.ReverseGeoLoc(toMarker.Position.Latitude, toMarker.Position.Longitude);

        j.ToLat = toMarker.Position.Latitude;

        j.ToLon = toMarker.Position.Longitude;

        j.Passengers = "";

        ProgressDialog progress;

        progress = new ProgressDialog(this);

           progress.Indeterminate = true;

           progress.SetProgressStyle(ProgressDialogStyle.Spinner);

           progress.SetMessage("Creating... Please wait...");

           progress.SetCancelable(false);

       progress.Show();

           CurrentPlatform.Init();

           DBHelper.InsertJourney(j);

        progress.Hide();

           Toast.MakeText(ApplicationContext, "Journey created!", ToastLength.Short).Show();

    }

    private void SetupSetFromButton()

    {

        Button animateButton = FindViewById<Button>(Resource.Id.buttonSetSearchJourneySetFrom);

           animateButton.Click += (sender, e) => {

               fromLocation.SetPosition(\_map.CameraPosition.Target);

           fromLocation.Visible(true);

            if (fromMarker != null)

            {

                   fromMarker.Remove();

            }

            fromMarker = \_map.AddMarker(fromLocation);

               fromCircleOp.InvokeCenter(\_map.CameraPosition.Target);

               fromCircleOp.InvokeRadius(rangeFinder.Progress \* 500);

               fromCircleOp.Visible(true);

            if (fromCircle != null)

            {

               fromCircle.Remove();

            }

            fromCircle = \_map.AddCircle(fromCircleOp);

        };

    }

    private void SetupSetToButton()

    {

        Button animateButton = FindViewById<Button>(Resource.Id.buttonSearchJourneySetTo);

           animateButton.Click += (sender, e) => {

               toLocation.SetPosition(\_map.CameraPosition.Target);

               toLocation.Visible(true);

            if (toMarker != null)

            {

               toMarker.Remove();

            }

            toMarker = \_map.AddMarker(toLocation);

               toCircleOp.InvokeCenter(\_map.CameraPosition.Target);

               toCircleOp.InvokeRadius(rangeFinder.Progress \* 500);

               toCircleOp.Visible(true);

            if (toCircle != null)

            {

                   toCircle.Remove();

            }

            toCircle = \_map.AddCircle(toCircleOp);

        };

    }

    private LatLng getCurrentPosition()

    {

        locMgr = GetSystemService(Context.LocationService) as LocationManager;

        string p = LocationManager.GpsProvider;

        Location l = locMgr.GetLastKnownLocation(p);

        if (l == null)

           {

            return new LatLng(52.8365, -6.9341);

        }

        return new LatLng(l.Latitude, l.Longitude);

    }

    public void OnMapReady(GoogleMap map)

    {

        \_map = map;

        SetupMapMarkers();

    }

    private void SetupMapIfNeeded()

    {

        if (\_map == null)

        {

            if (\_map != null)

            {

                   MarkerOptions markerOpt1 = new MarkerOptions();

                   markerOpt1.SetPosition(getCurrentPosition());

                   markerOpt1.SetTitle("Current Position");

                   markerOpt1.InvokeIcon(BitmapDescriptorFactory.DefaultMarker(BitmapDescriptorFactory.HueCyan));

               \_map.AddMarker(markerOpt1);

                // We create an instance of CameraUpdate, and move the map to it.

                   CameraUpdate cameraUpdate = CameraUpdateFactory.NewLatLngZoom(getCurrentPosition(), 15);

                \_map.MoveCamera(cameraUpdate);

            }

        }

    }

    void DateSelect\_OnClick(object sender, EventArgs eventArgs)

    {

        DatePickerFragment frag = DatePickerFragment.NewInstance(delegate (DateTime time)

        {

            depDate.Text = "Departure Date: " + time.DayOfWeek.ToString() + " " + time.ToLongDateString();

            initialDepDate = time;

        });

           frag.Show(FragmentManager, DatePickerFragment.TAG);

    }

    void TimeSelect\_OnClick(object sender, EventArgs eventArgs)

    {

        TimePickerFragment frag = TimePickerFragment.NewInstance(

            delegate (DateTime time)

            {

                depTime.Text = "Departure Time: "+ time.ToShortTimeString();

                   initialDepTime = time;

            });

           frag.Show(FragmentManager, TimePickerFragment.TAG);

    }

}

}

# SearchJourneyResultsActivity.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using Android.App;

using Android.Content;

using Android.OS;

using Android.Runtime;

using Android.Views;

using Android.Widget;

using CarShare.Models;

using Android.Gms.Maps.Model;

using Android.Gms.Maps;

using Newtonsoft.Json;

namespace CarShare

{

[Activity(Label = "SearchJourneyResultsActivity", NoHistory = true)]

public class SearchJourneyResultsActivity : Activity

{

    static ISharedPreferences pref = Application.Context.GetSharedPreferences("UserInfo", FileCreationMode.Private);

       ISharedPreferencesEditor edit = pref.Edit(); //out to session helper? future problem.

    ListView lv;

    List<Journeys> userJourneys;

    protected override void OnCreate(Bundle savedInstanceState)

    {

           RequestWindowFeature(WindowFeatures.NoTitle);

           base.OnCreate(savedInstanceState);

           SetContentView(Resource.Layout.SearchJourneysResults);

        string jsonJourney = pref.GetString("SearchResults", "");

        userJourneys = JsonConvert.DeserializeObject<List<Journeys>>(jsonJourney);

        string[] items = new string[1];

        if (userJourneys.Count > 0)

        {

            items = GetJourneyDetails(userJourneys);

        }

        else

        {

            items[0] = "No Upcomin Journeys Found";

        }

        IListAdapter adapter = new ArrayAdapter<String>(this, Android.Resource.Layout.SimpleListItem1, items);

        lv = FindViewById<ListView>(Resource.Id.listViewSearchJourneyResults);

        lv.Adapter = adapter;

        lv.ItemClick += ListView\_ItemClick;

        Button backBtn = FindViewById<Button>(Resource.Id.buttonSearchJourneyResultsBack);

        backBtn.Click += (sender, e) => { Finish(); };

        // Create your application here

    }

    private string[] GetJourneyDetails(List<Journeys> j)

    {

        List<string> items = new List<string>();

        foreach (Journeys i in j)

        {

            if (DateTime.Parse(i.DepartureDate) >= DateTime.Now && i.Filled == false)

            {

                string d = Convert.ToDateTime(i.DepartureDate).DayOfWeek.ToString();

                string dd = Convert.ToDateTime(i.DepartureDate).Day.ToString();

                string m = Convert.ToDateTime(i.DepartureDate).Month.ToString();

                string y = Convert.ToDateTime(i.DepartureDate).Year.ToString();

                string from = i.From;

                string to = i.To;

                string details = String.Format("{0} {6}/{1}/{2} {3} \nFrom: {4} \nTo: {5}", d, m, y, i.DepartureDateTime, from, to, dd);

                   items.Add(details);

            }

        }

        if (items.Count > 0)

        {

            return items.ToArray();

        }

        else

        {

            return new string[1] { "No Upcomin Journeys Found" };

        }

    }

    private async void ListView\_ItemClick(object sender, AdapterView.ItemClickEventArgs e)

    {

        var menu = new PopupMenu(this, lv.GetChildAt(e.Position - lv.FirstVisiblePosition));

           menu.Inflate(Resource.Layout.SearchJourneyMenu);

        Journeys j = userJourneys[e.Position];

        menu.MenuItemClick += async (s, a) =>

        {

            switch (a.Item.ItemId)

            {

                case Resource.Id.pop\_buttonSearchApply:

                       // Add user to applicants for journey

                       j.Applicants += pref.GetString("UserID", "")+",";

                       DBHelper.UpdateJourney(j);

                       UserProfiles up = await DBHelper.GetUsersProfile(j.DriverID);

                       DBHelper.SendApplicationNoticeEmail(up.Email, j);

                       StartActivity(typeof(MainProfileActivity));

                       break;

                case Resource.Id.pop\_buttonSearchView:

                       edit.PutString("SelectedJourneyID", j.ID);

                       edit.Commit();

                       StartActivity(typeof(ShowJourney));

                       break;

            }

        };

        menu.Show();

    }

}

}

# SetUpProfileActivity.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using Android.App;

using Android.Content;

using Android.OS;

using Android.Runtime;

using Android.Views;

using Android.Widget;

using System.Data.SqlClient;

using System.Text.RegularExpressions;

using Android.Text;

using Microsoft.WindowsAzure.MobileServices;

using CarShare.Models;

namespace CarShare

{

[Activity(Label = "SetUpProfileActivity")]

public class SetUpProfileActivity : Activity

{

    public static MobileServiceClient MobileService =

    new MobileServiceClient("https://c00197013.azurewebsites.net");

    static ISharedPreferences pref = Application.Context.GetSharedPreferences("UserInfo", FileCreationMode.Private);

       ISharedPreferencesEditor edit = pref.Edit();

    protected async override void OnCreate(Bundle savedInstanceState)

    {

       base.OnCreate(savedInstanceState);

           SetContentView(Resource.Layout.ProfileSetUp);

        #region setUpDropdowns

        Spinner genderSelect = FindViewById<Spinner>(Resource.Id.gender);

           //genderSelect.ItemSelected += new EventHandler<AdapterView.ItemSelectedEventArgs>(spinner\_ItemSelected);

        var adapter = ArrayAdapter.CreateFromResource(

                this, Resource.Array.gender, Android.Resource.Layout.SimpleSpinnerItem);

           adapter.SetDropDownViewResource(Android.Resource.Layout.SimpleSpinnerDropDownItem);

           genderSelect.Adapter = adapter;

        Spinner countySelect = FindViewById<Spinner>(Resource.Id.county);

        //genderSelect.ItemSelected += new EventHandler<AdapterView.ItemSelectedEventArgs>(spinner\_ItemSelected);

        var countyAdapter = ArrayAdapter.CreateFromResource(

                this, Resource.Array.county, Android.Resource.Layout.SimpleSpinnerItem);

           countyAdapter.SetDropDownViewResource(Android.Resource.Layout.SimpleSpinnerDropDownItem);

           countySelect.Adapter = countyAdapter;

        #endregion

        var saveButton = FindViewById(Resource.Id.trySaveButton);

       saveButton.Click += ValidateForm;

        //Create your application here

    }

    private async void TrySave()

    {

        ProgressDialog progress;

        progress = new Android.App.ProgressDialog(this);

        progress.Indeterminate = true;

           progress.SetProgressStyle(Android.App.ProgressDialogStyle.Spinner);

           progress.SetMessage("Logging In... Please wait...");

           progress.SetCancelable(false);

        progress.Show();

           CurrentPlatform.Init();

        EditText firstName = (EditText)FindViewById(Resource.Id.firstName);

        EditText lastName = (EditText)FindViewById(Resource.Id.lastName);

        EditText email = (EditText)FindViewById(Resource.Id.email);

        EditText phoneNo = (EditText)FindViewById(Resource.Id.phoneNo);

        Spinner gender = (Spinner)FindViewById(Resource.Id.gender);

        Spinner county = (Spinner)FindViewById(Resource.Id.county);

        UserProfiles newUserInfo = new UserProfiles { UsersID = pref.GetString("UserID","NULL"), Firstname = firstName.Text,

        Lastname = lastName.Text, Email = email.Text, PhoneNo = phoneNo.Text, Gender = gender.SelectedItem.ToString(),

         County = county.SelectedItem.ToString()};

           MobileService.GetTable<UserProfiles>().InsertAsync(newUserInfo);

        progress.Hide();

           Toast.MakeText(ApplicationContext, "User " + pref.GetString("UserName", "NULL") + " info created!", ToastLength.Short).Show();

    }

    private void ValidateForm(object sender, EventArgs e)

    {

        EditText firstName = (EditText)FindViewById(Resource.Id.firstName);

        EditText lastName = (EditText)FindViewById(Resource.Id.lastName);

           EditText email = (EditText)FindViewById(Resource.Id.email);

        EditText phoneNo = (EditText)FindViewById(Resource.Id.phoneNo);

        Spinner gender = (Spinner)FindViewById(Resource.Id.gender);

        Spinner county = (Spinner)FindViewById(Resource.Id.county);

        string namePattern = "[^a-zA-Z]";

        string phonePattern = "[^0-9+]";

        bool validInput = false;

        #region fieldValidation

        if (firstName.Text == "")

        {

               firstName.Error = "Cannot be Empty";

               firstName.RequestFocus();

        }

        else if (lastName.Text == "")

        {

            lastName.Error = "Cannot be Empty";

               lastName.RequestFocus();

        }

        else if (phoneNo.Text == "")

        {

            phoneNo.Error = "Cannot be Empty";

               phoneNo.RequestFocus();

        }

        else if (email.Text == "")

        {

            email.Error = "Cannot be Empty";

               email.RequestFocus();

        }

        else if (Regex.IsMatch(firstName.Text, namePattern))

        {

               firstName.Error = "Invalid characters in name";

               firstName.RequestFocus();

        }

        else if (Regex.IsMatch(lastName.Text, namePattern))

        {

            lastName.Error = "Invalid characters in name";

               lastName.RequestFocus();

        }

        else if (Regex.IsMatch(phoneNo.Text, phonePattern))

        {

            phoneNo.Error = "Invalid phone number. Use numbers (0-9) only";

               phoneNo.RequestFocus();

        }

        else if (!isValidEmail(email.Text))

        {

            email.Error = "Invalid email address";

               email.RequestFocus();

        }

        else

        {

            validInput = true;

        }

        #endregion

        if(validInput)

        {

               edit.PutString("FirstName", firstName.Text);

               edit.PutString("LastName", lastName.Text);

            edit.Commit();

            TrySave();

               StartActivity(typeof(MainProfileActivity));

      }

    }

    public static bool isValidEmail(string inputEmail)

    {

        string strRegex = @"^([a-zA-Z0-9\_\-\.]+)@((\[[0-9]{1,3}" +

                 @"\.[0-9]{1,3}\.[0-9]{1,3}\.)|(([a-zA-Z0-9\-]+\" +

                 @".)+))([a-zA-Z]{2,4}|[0-9]{1,3})(\]?)$";

        Regex re = new Regex(strRegex);

        if (re.IsMatch(inputEmail))

            return (true);

        else

            return (false);

    }

}

}

# ShowJourney.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using Android.App;

using Android.Content;

using Android.OS;

using Android.Runtime;

using Android.Views;

using Android.Widget;

using Microsoft.WindowsAzure.MobileServices;

using Android.Gms.Maps;

using Android.Locations;

using Android.Gms.Maps.Model;

using CarShare.Models;

using Android.Database;

using static Android.Views.View;

namespace CarShare

{

[Activity(Label = "ShowJourney")]

public class ShowJourney : Activity, IOnMapReadyCallback

{

    public static MobileServiceClient MobileService =

   new MobileServiceClient("https://c00197013.azurewebsites.net");

    static ISharedPreferences pref = Application.Context.GetSharedPreferences("UserInfo", FileCreationMode.Private);

       ISharedPreferencesEditor edit = pref.Edit();

    string UserID = pref.GetString("UserID", "");

    Journeys thisJourney;

    private GoogleMap \_map;

    private MapFragment \_mapFragment;

    LocationManager locMgr;

    MarkerOptions fromLocation;

    MarkerOptions toLocation;

    TextView tvFrom;

    TextView tvTo;

    TextView tvDate;

    TextView tvTime;

    TextView tvPassengerNumbers;

    ListView lvPassengers;

    ListView lstApplicants;

    Button btnApplicants;

    string[] passengers;

       List<UserProfiles> applicantsUserProfiles = new List<UserProfiles>();

       List<UserProfiles> passengersUserProfiles = new List<UserProfiles>();

    ArrayAdapter passengerAdapter;

    ArrayAdapter applicantsAdapter;

    List<string> passengerNames = new List<string>();

    List<string> applicantsNames = new List<string>();

    protected async override void OnCreate(Bundle savedInstanceState)

    {

           RequestWindowFeature(WindowFeatures.NoTitle);

           base.OnCreate(savedInstanceState);

           SetContentView(Resource.Layout.ShowJourney);

        string jID = pref.GetString("SelectedJourneyID", "");

        thisJourney = await DBHelper.GetJourneys(pref.GetString("SelectedJourneyID", ""));

           SetTextViewMessages();

        fromLocation = new MarkerOptions();

           fromLocation.SetTitle("From Here");

           fromLocation.SetIcon(BitmapDescriptorFactory.DefaultMarker(BitmapDescriptorFactory.HueRed));

           fromLocation.SetPosition(new LatLng(thisJourney.FromLat,thisJourney.FromLon));

           toLocation = new MarkerOptions();

           toLocation.SetTitle("To Here");

           toLocation.SetIcon(BitmapDescriptorFactory.DefaultMarker(BitmapDescriptorFactory.HueGreen));

           toLocation.SetPosition(new LatLng(thisJourney.ToLat, thisJourney.ToLon));

        InitMapFragment();

        // Create your application here

    }

    private async void SetTextViewMessages()

    {

        tvFrom = FindViewById<TextView>(Resource.Id.textViewShowJourneyFrom);

        tvTo = FindViewById<TextView>(Resource.Id.textViewShowJourneyTo);

        tvDate = FindViewById<TextView>(Resource.Id.textViewShowJourneyDate);

        tvTime = FindViewById<TextView>(Resource.Id.textViewShowJourneyTime);

        tvPassengerNumbers = FindViewById<TextView>(Resource.Id.textViewShowPassengerNumbers);

        tvFrom.Text = "From: " + thisJourney.From;

        tvTo.Text = "To: " + thisJourney.To;

        tvDate.Text = thisJourney.DepartureDate;

        tvTime.Text = thisJourney.DepartureDateTime;

        passengers = thisJourney.Passengers.Split(new char[1] { ',' }, StringSplitOptions.RemoveEmptyEntries);

        foreach (string s in passengers)

        {

            if (s.Length > 10)

           {

                   passengersUserProfiles.Add(await DBHelper.GetUsersProfile(s));

            }

        }

        foreach (UserProfiles up in passengersUserProfiles)

        {

               passengerNames.Add(up.Firstname + " " + up.Lastname);

        }

        lvPassengers = FindViewById<ListView>(Resource.Id.listViewShowJourneyPassengers);

        passengerAdapter = new ArrayAdapter<String>(this, Android.Resource.Layout.SimpleListItem1, passengerNames);

           lvPassengers.Adapter = passengerAdapter;

           tvPassengerNumbers.Text = "No of Passengers: " + passengerNames.Count() + "/" + thisJourney.NoOfPassengers;

        if (UserID == thisJourney.DriverID)

        {

            //show passengers

            //button checks applicants

            btnApplicants = FindViewById<Button>(Resource.Id.buttonShowJourneyApplicants);

               btnApplicants.Text = "Check Applications";

               btnApplicants.Click += ShowApplicants;

        }

    }

    private async void ShowApplicants(Object sender, EventArgs e)

    {

        if (thisJourney.Passengers.Split(new char[1] { ',' }, StringSplitOptions.RemoveEmptyEntries).Count() >= thisJourney.NoOfPassengers)

        {

               Toast.MakeText(ApplicationContext, "Car is full, cannot accept any more Applicants", ToastLength.Long).Show();

        }

        else

        {

               applicantsUserProfiles.Clear();

            applicantsNames.Clear();

            string[] appl = thisJourney.Applicants.Split(new char[1] { ',' }, StringSplitOptions.RemoveEmptyEntries);

            foreach (string s in appl)

            {

                if (s.Length > 10)

               {

                       applicantsUserProfiles.Add(await DBHelper.GetUsersProfile(s));

                }

            }

            foreach (UserProfiles up in applicantsUserProfiles)

            {

                   applicantsNames.Add(up.Firstname + " " + up.Lastname);

            }

            lstApplicants = FindViewById<ListView>(Resource.Id.listViewShowJourneyApplicants);

               applicantsAdapter = new ArrayAdapter<String>(this, Android.Resource.Layout.SimpleListItem1, applicantsNames);

               lstApplicants.Adapter = applicantsAdapter;

               lstApplicants.Visibility = ViewStates.Visible;

               lstApplicants.ItemClick += ListView\_ItemClick;

       }

    }

    private void ListView\_ItemClick(object sender, AdapterView.ItemClickEventArgs e)

    {

        var menu = new PopupMenu(this, lstApplicants.GetChildAt(e.Position - lstApplicants.FirstVisiblePosition));

        UserProfiles up = applicantsUserProfiles[e.Position];

           menu.Inflate(Resource.Layout.ApplicantsMenu);

        menu.MenuItemClick += (s, a) =>

        {

            switch (a.Item.ItemId)

            {

                case Resource.Id.ApplicantAccept:

                    //Add Applicant to passengers for journey, increment num passengers by 1

                    if (passengerNames.Count() < thisJourney.NoOfPassengers)

                    {

                           thisJourney.Passengers += up.UsersID + ","; // userId

                           string[] appl = thisJourney.Applicants.Split(new char[1] { ',' }, StringSplitOptions.RemoveEmptyEntries);

                           thisJourney.Applicants = "";

                        if (thisJourney.Passengers.Split(new char[1] { ',' }, StringSplitOptions.RemoveEmptyEntries).Count() < thisJourney.NoOfPassengers)

                        {

                               foreach (string st in appl)

                               {

                                   if (st != up.UsersID)

                                   {

                                       thisJourney.Applicants += st + ",";

                                   }

                               }

                        }

                           applicantsAdapter.Clear();

                           applicantsAdapter.NotifyDataSetChanged();

                           passengerNames.Add(up.Firstname + " " + up.Lastname);

                           passengerAdapter.Add(up.Firstname + " " + up.Lastname);

                           passengerAdapter.NotifyDataSetChanged();

                           lstApplicants.Visibility = ViewStates.Invisible;

                           tvPassengerNumbers.Text = "No of Passengers: " + passengerNames.Count() + "/" + thisJourney.NoOfPassengers;

                           //get applicants email and send them message

                           DBHelper.UpdateJourney(thisJourney);

                           DBHelper.SendApplicationEmail(up.Email, thisJourney);

                    }

                    break;

                case Resource.Id.ApplicantRefuse:

                       string[] appl2 = thisJourney.Applicants.Split(new char[1] { ',' }, StringSplitOptions.RemoveEmptyEntries);

                       thisJourney.Applicants = "";

                       foreach (string st in appl2)

                   {

                        if (st != up.UsersID)

                        {

                               thisJourney.Applicants += st + ",";

                        }

                    }

                       applicantsAdapter.Clear();

                       applicantsAdapter.NotifyDataSetChanged();

                       DBHelper.UpdateJourney(thisJourney);

                    // cancel/delete this journey, if it has no passengers.

                    break;

           }

        };

        menu.Show();

    }

    private void SetupMapMarkers()

    {

           \_map.AddMarker(fromLocation);

           \_map.AddMarker(toLocation);

    }

    private void InitMapFragment()

    {

       \_mapFragment = FragmentManager.FindFragmentByTag("mapFrameShowJourney") as MapFragment;

        if (\_mapFragment == null)

        {

               GoogleMapOptions mapOptions = new GoogleMapOptions()

                   .InvokeMapType(GoogleMap.MapTypeNormal)

                   .InvokeZoomControlsEnabled(false)

                   .InvokeCompassEnabled(true)

                   .InvokeCamera(new CameraPosition(new LatLng(thisJourney.FromLat, thisJourney.FromLon), 11, 65, 0))

                ;

               FragmentTransaction fragTx = FragmentManager.BeginTransaction();

            \_mapFragment = MapFragment.NewInstance(mapOptions);

               fragTx.Add(Resource.Id.mapFrameShowJourney, \_mapFragment, "mapFrameShowJourney");

               fragTx.Commit();

        }

           \_mapFragment.GetMapAsync(this);

    }

    public void OnMapReady(GoogleMap map)

    {

        \_map = map;

        SetupMapMarkers();

  }

    protected override void OnResume()

    {

        base.OnResume();

           SetupMapIfNeeded();

    }

    private void SetupMapIfNeeded()

    {

        if (\_map == null)

        {

            if (\_map != null)

            {

                   MarkerOptions markerOpt1 = new MarkerOptions();

                   markerOpt1.SetPosition(getCurrentPosition());

                   markerOpt1.SetTitle("Current Position");

                   markerOpt1.InvokeIcon(BitmapDescriptorFactory.DefaultMarker(BitmapDescriptorFactory.HueCyan));

                   \_map.AddMarker(markerOpt1);

                // We create an instance of CameraUpdate, and move the map to it.

                   CameraUpdate cameraUpdate = CameraUpdateFactory.NewLatLngZoom(getCurrentPosition(), 15);

                   \_map.MoveCamera(cameraUpdate);

            }

        }

    }

    public LatLng getCurrentPosition()

    {

        locMgr = GetSystemService(Context.LocationService) as LocationManager;

        string p = LocationManager.GpsProvider;

        Location l = locMgr.GetLastKnownLocation(p);

        if (l == null)

        {

            return new LatLng(52.8365, -6.9341);

        }

           return new LatLng(l.Latitude, l.Longitude);

    }

}

}

# ViewOldPassengerJourneysActivity.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using Android.App;

using Android.Content;

using Android.OS;

using Android.Runtime;

using Android.Views;

using Android.Widget;

using CarShare.Models;

using Android.Gms.Maps.Model;

using Android.Gms.Maps;

namespace CarShare

{

[Activity(Label = "ViewOldPassengerJourneysActivity")]

public class ViewOldPassengerJourneysActivity : Activity

{

    static ISharedPreferences pref = Application.Context.GetSharedPreferences("UserInfo", FileCreationMode.Private);

       ISharedPreferencesEditor edit = pref.Edit(); //out to session helper? future problem.

    ListView lv;

    List<Journeys> userJourneys;

    protected async override void OnCreate(Bundle savedInstanceState)

    {

           RequestWindowFeature(WindowFeatures.NoTitle);

           base.OnCreate(savedInstanceState);

           SetContentView(Resource.Layout.ViewOldPassengerJourneys);

        userJourneys = await DBHelper.GetUsersOldPassengerJourneys(pref.GetString("UserID", ""));

        string[] items;

        if (userJourneys.Count > 0)

        {

               items = GetJourneyDetails(userJourneys);

        }

        else

        {

            items = new string[1] { "No Upcoming Journeys" };

        }

        IListAdapter adapter = new ArrayAdapter<String>(this, Android.Resource.Layout.SimpleListItem1, items);

        lv = FindViewById<ListView>(Resource.Id.listViewPassengerOldJourney);

        lv.Adapter = adapter;

        lv.ItemClick += ListView\_ItemClick;

        Button backBtn = FindViewById<Button>(Resource.Id.buttonPassengerOldJourneyBack);

        backBtn.Click += (sender, e) => { Finish(); };

        // Create your application here

    }

    private string[] GetJourneyDetails(List<Journeys> j)

    {

        List<string> items = new List<string>();

        foreach (Journeys i in j)

        {

            string d = Convert.ToDateTime(i.DepartureDate).DayOfWeek.ToString();

            string dd = Convert.ToDateTime(i.DepartureDate).Day.ToString();

            string m = Convert.ToDateTime(i.DepartureDate).Month.ToString();

            string y = Convert.ToDateTime(i.DepartureDate).Year.ToString();

            string from = i.From;

            string to = i.To;

           string details = String.Format("{0} {6}/{1}/{2} {3} \n\nFrom: {4} \nTo: {5}\n\n", d, m, y, i.DepartureDateTime, from, to, dd);

               items.Add(details);

        }

        return items.ToArray();

    }

    private void ListView\_ItemClick(object sender, AdapterView.ItemClickEventArgs e)

    {

        var menu = new PopupMenu(this, lv.GetChildAt(e.Position - lv.FirstVisiblePosition)); //lv.GetChildAt(e.Position)

           menu.Inflate(Resource.Layout.JourneyMenu);

           menu.MenuItemClick += (s, a) =>

        {

            switch (a.Item.ItemId)

            {

                case Resource.Id.pop\_button1:

                       Journeys j = userJourneys[e.Position];

                    edit.PutString("SelectedJourneyID", j.ID);

                       edit.Commit();

                       StartActivity(typeof(ShowJourney));

                    // goto details for selected journey

                    break;

                case Resource.Id.pop\_button2:

                    // cancel/delete this journey, if it has no passengers.

                    break;

            }

        };

        try

        {

            menu.Show();

           }

        catch (Exception ex)

        {

        }

    }

}

}

# ViewPassengerJourneysActivity.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using Android.App;

using Android.Content;

using Android.OS;

using Android.Runtime;

using Android.Views;

using Android.Widget;

using CarShare.Models;

using Android.Gms.Maps.Model;

using Android.Gms.Maps;

namespace CarShare

{

[Activity(Label = "ViewPassengerJourneysActivity")]

public class ViewPassengerJourneysActivity : Activity

{

    static ISharedPreferences pref = Application.Context.GetSharedPreferences("UserInfo", FileCreationMode.Private);

       ISharedPreferencesEditor edit = pref.Edit(); //out to session helper? future problem.

    ListView lv;

    List<Journeys> userJourneys;

    protected async override void OnCreate(Bundle savedInstanceState)

    {

           RequestWindowFeature(WindowFeatures.NoTitle);

           base.OnCreate(savedInstanceState);

           SetContentView(Resource.Layout.ViewUpcomingPassengerJourneys);

        userJourneys = await DBHelper.GetUsersPassengerJourneys(pref.GetString("UserID", ""));

        string[] items;

        if (userJourneys.Count > 0)

        {

            items = GetJourneyDetails(userJourneys);

        }

        else

        {

            items = new string[1] { "No Upcoming Journeys" };

        }

        IListAdapter adapter = new ArrayAdapter<String>(this, Android.Resource.Layout.SimpleListItem1, items);

        lv = FindViewById<ListView>(Resource.Id.listViewPassengerJourney);

        lv.Adapter = adapter;

        lv.ItemClick += ListView\_ItemClick;

        Button backBtn = FindViewById<Button>(Resource.Id.buttonPassengerJourneyBack);

        backBtn.Click += (sender, e) => { Finish(); };

        // Create your application here

    }

    private string[] GetJourneyDetails(List<Journeys> j)

    {

        List<string> items = new List<string>();

        foreach (Journeys i in j)

        {

            string d = Convert.ToDateTime(i.DepartureDate).DayOfWeek.ToString();

            string dd = Convert.ToDateTime(i.DepartureDate).Day.ToString();

            string m = Convert.ToDateTime(i.DepartureDate).Month.ToString();

            string y = Convert.ToDateTime(i.DepartureDate).Year.ToString();

            string from = i.From;

            string to = i.To;

            string details = String.Format("{0} {6}/{1}/{2} {3} \n\nFrom: {4} \nTo: {5}\n\n", d, m, y, i.DepartureDateTime, from, to, dd);

               items.Add(details);

        }

        return items.ToArray();

    }

    private void ListView\_ItemClick(object sender, AdapterView.ItemClickEventArgs e)

    {

        var menu = new PopupMenu(this, lv.GetChildAt(e.Position - lv.FirstVisiblePosition)); //lv.GetChildAt(e.Position)

           menu.Inflate(Resource.Layout.JourneyMenu);

        menu.MenuItemClick += (s, a) =>

        {

            switch (a.Item.ItemId)

            {

                case Resource.Id.pop\_button1:

                       Journeys j = userJourneys[e.Position];

                  edit.PutString("SelectedJourneyID", j.ID);

                       edit.Commit();

                       StartActivity(typeof(ShowJourney));

                    // goto details for selected journey

                    break;

                   case Resource.Id.pop\_button2:

                    // cancel/delete this journey, if it has no passengers.

                    break;

            }

        };

        try

        {

            menu.Show();

        }

        catch (Exception ex)

        {

        }

    }

}

}