Pavements

Technical Manual

20th April 2020

Institiúid Teicneolaíochta Cheatharlach

INSTITUTE *of* TECHNOLOGY CARLOW

At the Heart of South Leinster

Name: Michael Chambers

Year: 4th Year

Student ID: C00220585

Supervisor: Paul Barry

Abstract

The purpose of this document is to show all code that has been written to make the whole project work. The primary user of this application is going to be a current employee of McCurdy associates. This mobile application will provide McCurdy associates employees an application that will help with their surveys that they complete quite frequently.

Table of Contents

1.Introduction

The following document contains all the code written for the application. The Pavements application is a cross-platform application that works on both Android and iOS. To install the application in its current state would require the latest Visual Studio with all Xamarin extensions that can be installed when installing Visual Studio.

The C#/Xamarin code is broken down into the sections as the application was built like Custom Renderers, Helper, Model, and Views. Some code was not added to this document as it was auto generated when first setting up the application. These files include MainActivity.cs, Android Manifest, AppDelegate.cs and info.plist. To see the full extent of the code can be seen at the GitHub link linked below.

Finally, all XAML code written in this document is provided to show all the workings of how all the screens provided in the application are displayed.

All code can be found at the GitHub link below:
https://github.com/michaelchambers89/McurdyAssociates

2. C#/ Xamarin Code

2.1 Custom Renderers

2.1.1 CustomMapForClickEvent.cs

```
using System;
using Xamarin.Essentials;
using Xamarin.Forms.Maps;
using Map = Xamarin.Forms.Maps.Map;

namespace McurdyAssociates.CustomRenderers
{
    public class CustomMapForClickEvent : Map
    {
        public event EventHandler<TapEventArgs> Tap;

        public CustomMapForClickEvent()
        {

        }

        public CustomMapForClickEvent(MapSpan region) : base(region)
        {

        }

        public void OnTap(Position coordinate)
        {
            OnTap(new TapEventArgs { Position = coordinate });
        }

        protected virtual void OnTap(TapEventArgs e)
        {
            var handler = Tap;
            if (handler != null) handler(this, e);
        }
    }

    public class TapEventArgs : EventArgs
    {
        public Position Position { get; set; }
    }
}
```

2.2 Helper

2.2.1 FirebaseHelper.cs

```
using System;
using Firebase.Database;
using Firebase.Database.Query;
using System.Linq;
using System.Threading.Tasks;
using Newtonsoft.Json;
using McurdyAssociates.Model;
using System.Collections.Generic;
using LiteDB;
using Firebase.Storage;

namespace McurdyAssociates.Helper
{
    public class FireBaseHelper
    {
        FirebaseStorage firebaseStorage = new FirebaseStorage("mccurdyassociates-
9d006.appspot.com");
        FirebaseClient firebase = new FirebaseClient("https://mccurdyassociates-
9d006.firebaseio.com/");
         //public FireBaseHelper()
        //{
        //}

        public async Task AddSurvey(int newSurveyId, string surveyLocation, string date, string
employee, string weather, string surfaceType, string direction, string lane, double surveyLat,
double surveyLong)
        {

            await firebase
              .Child("NewSurvey")
              .PostAsync(new NewSurveyModel() { NewSurveyID = newSurveyId,
                                    SurveyLocation = surveyLocation,
                                    Date = date,
                                    CompletedBy = employee,
                                    Weather = weather,
                                    SurfaceType = surfaceType,
                                    Direction = direction,
                                    Lane = lane,
                                    SurveyLatitute = surveyLat,
                                    SurveyLongitute = surveyLong
              });
        }
        public async Task AddDefect(int surveyDefectId, int surveyId, string chainage, string
defectName, string measurement, string value, double defectLong, double defectLat, string
comments)
        {

            await firebase
              .Child("NewDefect")
              .PostAsync(new SurveyDefect()
              {
                  SurveyDefectId = surveyDefectId,
                  SurveyId = surveyId,
                  Chainage = chainage,
                  DefectName = defectName,
                  Measurment = measurement,
                  Value = value,
                  DefectLongitute = defectLong,
                  DefectLatitude = defectLat,
                  Comments = comments
              });
        }

        public async Task<List<NewSurveyModel>> GetAllSurveys()
        {
            return (await firebase
              .Child("NewSurvey")
              .OnceAsync<NewSurveyModel>()).Select(item => new NewSurveyModel
              {
                  NewSurveyID = item.Object.NewSurveyID,
                  SurveyLocation = item.Object.SurveyLocation,
                  Date = item.Object.Date,
                  CompletedBy = item.Object.CompletedBy,
                  Weather = item.Object.Weather,
                  SurfaceType = item.Object.SurfaceType,
                  Direction = item.Object.Direction,
                  Lane = item.Object.Lane
              }).ToList();
        }

        public async Task<List<SurveyDefect>> GetAllSurveyDefects()
        {
            return (await firebase
              .Child("NewDefect")
              .OnceAsync<SurveyDefect>()).Select(item => new SurveyDefect
              {
                  SurveyDefectId = item.Object.SurveyDefectId,
                  SurveyId = item.Object.SurveyId,
                  Chainage = item.Object.Chainage,
                  DefectName = item.Object.DefectName,
                Measurment = item.Object.Measurment,
                Value = item.Object.Value,
                Comments = item.Object.Comments

            }).ToList();
        }

        public async Task UpdateSurvey(int surveyId, string completedBy, string
surveyLocation, string date)
        {
            var toUpdateSurvey = (await firebase
              .Child("NewSurvey")
              .OnceAsync<NewSurveyModel>()).Where(a => a.Object.NewSurveyID ==
surveyId).FirstOrDefault();

            await firebase
              .Child("NewSurvey")
              .Child(toUpdateSurvey.Key)
              .PutAsync(new NewSurveyModel()
              {
                NewSurveyID = surveyId,
                SurveyLocation = surveyLocation,
                Date = date,
                CompletedBy = completedBy,
              });
        }

        public async Task UpdateDefect(int defectId, int surveyId, string defectName, string
chainage, string measurment, string value, string comments)
        {
            var toUpdateDefect = (await firebase
              .Child("NewDefect")
              .OnceAsync<SurveyDefect>()).Where(a => a.Object.SurveyDefectId ==
defectId).FirstOrDefault();

            await firebase
              .Child("NewDefect")
              .Child(toUpdateDefect.Key)
              .PutAsync(new SurveyDefect()
              {
                SurveyDefectId = defectId,
                SurveyId = surveyId,
                DefectName = defectName,
                Chainage = chainage,
                Measurment = measurment,
                Value = value,
                Comments = comments
              }) ;
        }

        public async Task DeleteDefect(int defectId)
        {
            var toDeleteDefect = (await firebase
              .Child("NewDefect")
              .OnceAsync<SurveyDefect>()).Where(a => a.Object.SurveyDefectId ==
defectId).FirstOrDefault();
            await firebase.Child("NewDefect").Child(toDeleteDefect.Key).DeleteAsync();

        }

        public async Task<string> GetFile(string fileName)
        {
            return await firebaseStorage
                .Child("McCurdyAssociates")
                .Child(fileName)
                .GetDownloadUrlAsync();
        }

        public async Task DeleteFile(string fileName)
        {
            await firebaseStorage
                 .Child("McCurdyAssociates")
                 .Child(fileName)
                 .DeleteAsync();

        }

        public async Task AddEmployee(int employeeId, string employeeName, string position)
        {

            await firebase
              .Child("Employee")
              .PostAsync(new Employee()
              {
                  EmployeeID = employeeId,
                  EmployeeName = employeeName,
                  Position = position
              });
        }

        public async Task<List<Employee>> GetAllEmployees()
        {
            return (await firebase
              .Child("Employee")
              .OnceAsync<Employee>()).Select(item => new Employee
```

```
           {
              EmployeeID = item.Object.EmployeeID,
              EmployeeName = item.Object.EmployeeName,
              Position = item.Object.Position

           }).ToList();
       }

       public async Task DeleteEmployee(int employeeId)
       {
          var toDeleteEmployee = (await firebase
            .Child("Employee")
            .OnceAsync<Employee>()).Where(a => a.Object.EmployeeID ==
employeeId).FirstOrDefault();
          await firebase.Child("Employee").Child(toDeleteEmployee.Key).DeleteAsync();

       }

       public async Task UpdateEmployee(int employeeId, string employeeName, string
position)
       {
          var toUpdateEmployee = (await firebase
            .Child("Employee")
            .OnceAsync<Employee>()).Where(a => a.Object.EmployeeID ==
employeeId).FirstOrDefault();

          await firebase
            .Child("Employee")
            .Child(toUpdateEmployee.Key)
            .PutAsync(new Employee()
            {
              EmployeeID = employeeId,
              EmployeeName = employeeName,
              Position = position
            });
       }


    }
}
```

2.3 Model

2.3.1 Employee.cs

```
using System;
using System.Collections.Generic;

namespace McurdyAssociates.Model
{
    public class Employee
    {
        private int employeeId;
        public int EmployeeID
        {
            get { return employeeId; }
            set { employeeId = value; }
        }

        private string employeeName;

        public string EmployeeName
        {
            get { return employeeName; }
            set { employeeName = value; }
        }

        private string position;

        public string Position
        {
            get { return position; }
            set { position = value; }
        }

    }
}
```

2.3.2 NewSurveyModel.cs

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Firebase.Database;

namespace McurdyAssociates.Model
{
    public class NewSurveyModel
    {
        public int NewSurveyID { get; set; }
        public string SurveyLocation { get; set; }
        public string Date { get; set; }
        public string CompletedBy { get; set; }
        public string Weather { get; set; }
        public string SurfaceType { get; set; }
        public string Direction { get; set; }
        public string Lane { get; set; }
        public double SurveyLatitute { get; set; }
        public double SurveyLongitute { get; set; }
    }
}
```

2.3.3 SurveyDefect.cs

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Firebase.Database;

namespace McurdyAssociates.Model
{
    public class SurveyDefect
    {
        public int SurveyDefectId { get; set; }
        public int SurveyId { get; set; }
        public string Chainage { get; set; }
        public string DefectName { get; set; }
        public string Measurment { get; set; }
        public string Value { get; set; }
        public double DefectLatitude { get; set; }
        public double DefectLongitute { get; set; }
        public string Comments { get; set; }
    }
}
```

2.4 Views

2.4.1 EmployeesPage.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using McurdyAssociates.Helper;
using McurdyAssociates.Model;
using Xamarin.Forms;

namespace McurdyAssociates.Views
{
    public partial class EmployeesPage : ContentPage
    {
        public int SelectedEmployeeID { get; set; }
        FireBaseHelper firebaseHelper = new FireBaseHelper();
        List<Model.Employee> Employees { get; set; }
        public string EmployeeNameToUpdate { get; set; }
        public string PositionToUpdate { get; set; }


        public EmployeesPage(List<Model.Employee> allEmployees)
        {
            InitializeComponent();

            allEmployees = allEmployees.OrderBy(a => a.EmployeeID).ToList();
            Employees = allEmployees;
            lstEmp.ItemsSource = allEmployees;
        }

        private async void Delete_Clicked(object sender, EventArgs e)
        {
            try
            {
                var selectedItem = (Employee)lstEmp.SelectedItem;
                SelectedEmployeeID = selectedItem.EmployeeID;

                var action = await DisplayAlert("Delete?", "Are you sure", "Yes", "No");
                if (action)
                {

                    await firebaseHelper.DeleteEmployee(Convert.ToInt32(SelectedEmployeeID));
                    Employees = await firebaseHelper.GetAllEmployees();
                    var vUpdatedPage = new EmployeesPage(Employees);
                    Navigation.InsertPageBefore(vUpdatedPage, this);
                    Navigation.PopAsync();
                }
            }
            catch (NullReferenceException)
            {
                await DisplayAlert("", "Item Not Selected from List", "Ok");
            }


        }

        private async void Update_Clicked(object sender, EventArgs e)
        {
            try
            {
                var selectedItem = (Employee)lstEmp.SelectedItem;
                SelectedEmployeeID= selectedItem.EmployeeID;

                lstEmp.ItemsSource = Employees;
                List<Model.Employee> GetSingleEmployee = Employees.Where(a =>
a.EmployeeID.Equals(SelectedEmployeeID)).ToList();
                EmployeeNameToUpdate = GetSingleEmployee.Select(a =>
a.EmployeeName).FirstOrDefault();
                PositionToUpdate = GetSingleEmployee.Select(a => a.Position).FirstOrDefault();

                await Navigation.PushAsync(new UpdateEmployee(SelectedEmployeeID,
EmployeeNameToUpdate, PositionToUpdate));
            }
            catch (NullReferenceException)
            {
                await DisplayAlert("", "Item Not Selected from List", "Ok");
            }
        }

        private async void NewEmployeePage(object sender, EventArgs e)
        {

            await Navigation.PushAsync(new NewEmployeePage());
        }
    }
}
```

2.4.2 ExtraSurveyDetail.xaml.cs

```
using System;
using System.Collections.Generic;
using McurdyAssociates.Helper;
using Xamarin.Forms;

namespace McurdyAssociates.Views
{
    public partial class ExtraSurveyDetails : ContentPage
    {
        public int SurveyId { get; set; }
        public string Employee { get; set; }
        public String Street { get; set; }
        public String Weather { get; set; }
        public String SurfaceTypeDb { get;set; }
        public String DirectionDb { get; set; }
        public String LaneDb { get; set; }
        public double Lat { get; set; }
        public double Long { get; set; }
        FireBaseHelper firebaseHelper = new FireBaseHelper();
        public ExtraSurveyDetails(int surveyId, string employee, string street, string weatherDb,
double lat, double @long)
        {
            InitializeComponent();
            SurveyId = surveyId;
            Employee = employee;
            Street = street;
            Weather = weatherDb;
            Lat = lat;
            Long = @long;

            SurfaceTypeOther1.IsVisible = false;
            DirectionOther1.IsVisible = false;
            LaneOther1.IsVisible = false;

            SurfaceType.SelectedIndexChanged += (sender, args) =>
            {
                if (SurfaceType.SelectedItem == "Other")
                {
                    SurfaceTypeOther1.IsVisible = true;
                }
                else
                {
                    SurfaceTypeOther1.IsVisible = false;
                }
            };
            Direction.SelectedIndexChanged += (sender, args) =>
            {
                if (Direction.SelectedItem == "Other")
                {
                    DirectionOther1.IsVisible = true;
                }
                else
                {
                    DirectionOther1.IsVisible = false;
                }
            };

            Lane.SelectedIndexChanged += (sender, args) =>
            {
                if (Lane.SelectedItem == "Other")
                {
                    LaneOther1.IsVisible = true;
                }
                else
                {
                    LaneOther1.IsVisible = false;
                }
            };

        }

        private async void NavigateButton_OnClicked(object sender, EventArgs e)
        {
            if (SurfaceType.SelectedIndex == -1 || Direction.SelectedIndex == -1 ||
Lane.SelectedIndex == -1)
            {
                await DisplayAlert("Error", "Not all fields are completed", "OK");
            }
            else if (SurfaceTypeOther1.IsVisible && SurfaceTypeOther.Text ==null)
            {
                await DisplayAlert("Error", "Not all fields are completed", "OK");
            }
            else if (DirectionOther1.IsVisible && DirectionOther.Text == null)
            {
                await DisplayAlert("Error", "Not all fields are completed", "OK");
            }
            else if (LaneOther1.IsVisible && LaneOther.Text == null)
            {
                await DisplayAlert("Error", "Not all fields are completed", "OK");
            }
            else
            {
```

```
            if (SurfaceType.SelectedItem != "Other")
            {
               SurfaceTypeDb = SurfaceType.SelectedItem.ToString();
            }
            else
            {
               SurfaceTypeDb = SurfaceTypeOther.Text;
            }
            if (Direction.SelectedItem != "Other")
            {
               DirectionDb = Direction.SelectedItem.ToString();
            }
            else
            {
               DirectionDb = DirectionOther.Text;
            }
            if (Lane.SelectedItem != "Other")
            {
               LaneDb = Lane.SelectedItem.ToString();
            }
            else
            {
               DirectionDb = LaneOther.Text;
            }
            var date = DateTime.Now.ToString();
            await firebaseHelper.AddSurvey(Convert.ToInt32(SurveyId), Street, date,
Employee, Weather, SurfaceTypeDb, DirectionDb, LaneDb, Lat, Long);
            await Navigation.PushAsync(new NewSurvey(SurveyId));
         }

      }
   }
}
```

2.4.3 ImageDisplay.xaml.cs

```
using System;
using System.Collections.Generic;
using McurdyAssociates.Views;
using Plugin.Media.Abstractions;
using Xamarin.Forms;

namespace McurdyAssociates.Views
{
   public partial class ImageDisplay : ContentPage
   {
      public string DefectName { get; set; }
      public string Measurment { get; set; }
      public int SurveyId { get; set; }
      public double Long { get; set; }
      public double Lat { get; set; }
      List<Model.SurveyDefect> ListDefect { get; set; }
      MediaFile photo;
      System.IO.Stream NewPhoto { get; set; }

      public ImageDisplay(System.IO.Stream stream, string defectName, string
measurement, int surveyId, List<Model.SurveyDefect> listDefect, double @long, double lat)
      {
         InitializeComponent();
         Image.Source = ImageSource.FromStream(() => stream);
         DefectName = defectName;
         Measurment = measurement;
         SurveyId = surveyId;
         ListDefect = listDefect;
         Long = @long;
         Lat = lat;
      }

      private async void RetakeButton(object sender, EventArgs e)
      {
         photo = await Plugin.Media.CrossMedia.Current.TakePhotoAsync(new
Plugin.Media.Abstractions.StoreCameraMediaOptions() { });

         if (photo != null)
         {
            Image.Source = ImageSource.FromStream(() => { return photo.GetStream(); });
            NewPhoto = photo.GetStream();
         }
         var vUpdatedPage = new SurveyInfo(DefectName, Measurment, SurveyId,
ListDefect, NewPhoto);
         Navigation.InsertPageBefore(vUpdatedPage, this);
         Navigation.PopAsync();
      }
   }
}
```

2.4.4 MainPage.xaml.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Firebase.Database;
using Firebase.Database.Query;
using McurdyAssociates.Helper;
using McurdyAssociates.Model;
using McurdyAssociates.Views;

using Xamarin.Essentials;
using Xamarin.Forms;
using Xamarin.Forms.Maps;

namespace McurdyAssociates
{
    // Learn more about making custom code visible in the Xamarin.Forms previewer
    // by visiting https://aka.ms/xamarinforms-previewer
    [DesignTimeVisible(false)]
    public partial class MainPage : ContentPage
    {
        public double Lat { get; set; }
        public double Long { get; set; }
        public string SubArea { get; set; }
        public string SubLocality { get; set; }
        public string Street { get; set; }
        public int CurrentSurveyID { get; set; }
        FireBaseHelper firebaseHelper = new FireBaseHelper();
        public MainPage()
        {
            InitializeComponent();
        }

        private async void NavigateButton_OnClicked(object sender, EventArgs e)
        {
            popupLoadingView.IsVisible = true;
            activityIndicator.IsRunning = true;
            var location = await Geolocation.GetLocationAsync();
            if (location != null)
            {
                Lat = location.Latitude;
                Long = location.Longitude;
            }
            var placemarks = await Geocoding.GetPlacemarksAsync(Lat, Long);
            var placemark = placemarks?.FirstOrDefault();
            if (placemark != null)
            {
                //SubArea = placemark.SubAdminArea;//null
                SubLocality = placemark.SubLocality;//Coneyboro
                Street = placemark.FeatureName;//16
                var loc = placemark.AdminArea;//County kildre
                var loc2 = placemark.SubThoroughfare;//16
                SubArea = placemark.Thoroughfare;// Coney Green
            }


            var date = DateTime.Now.ToString();
            var LastId = await firebaseHelper.GetAllSurveys();
            //CurrentSurveyID = LastId.Count + 1;
            CurrentSurveyID = LastId.Max(a => a.NewSurveyID)+1;
            await Navigation.PushAsync(new SelectLocation(Lat, Long, Street, SubArea,
SubLocality, CurrentSurveyID));
            popupLoadingView.IsVisible = false;
        }

        private async void PveviousSurvey_OnClicked(object sender, EventArgs e)
        {
            popupLoadingView.IsVisible = true;
            activityIndicator.IsRunning = true;
            var allSurveys = await firebaseHelper.GetAllSurveys();

            await Navigation.PushAsync(new PreviousSurveys(allSurveys));
            popupLoadingView.IsVisible = false;
        }

        private async void Employee_OnClicked(object sender, EventArgs e)
        {
            popupLoadingView.IsVisible = true;
            activityIndicator.IsRunning = true;
            var allEmployees = await firebaseHelper.GetAllEmployees();
            await Navigation.PushAsync(new EmployeesPage(allEmployees));
            popupLoadingView.IsVisible = false;

        }
    }
}
```

2.4.5 NewEmployeePage.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using McurdyAssociates.Helper;
using Xamarin.Forms;

namespace McurdyAssociates.Views
{
    public partial class NewEmployeePage : ContentPage
    {
        FireBaseHelper firebaseHelper = new FireBaseHelper();
        public int EmployeeId { get; set; }
        public string EmployeeName { get; set; }
        public string Position { get; set; }

        public NewEmployeePage()
        {
            InitializeComponent();
            EmployeeName = NameVal.Text;
            Position = PositionVal.Text;
        }

        private async void Save_Clicked(object sender, EventArgs e)
        {
            if (NameVal.Text == null && PositionVal.Text == null)
            {
                await DisplayAlert("Error", "Not all fields are completed", "OK");
            }
            else
            {
                var allEmp = await firebaseHelper.GetAllEmployees();
                EmployeeId = allEmp.Max(a => a.EmployeeID) + 1;
                await firebaseHelper.AddEmployee(Convert.ToInt32(EmployeeId), NameVal.Text,
PositionVal.Text);
                await DisplayAlert("Employee Added", "", "ok");
                await Navigation.PushAsync(new MainPage());
            }
        }
    }
}
```

2.4.6 NewSurvey.xaml.cs

```
using System;
using System.Collections.Generic;
using McurdyAssociates.ViewModels;
using Xamarin.Forms;
using Prism.Navigation;
using McurdyAssociates.Helper;
using Xamarin.Essentials;

namespace McurdyAssociates
{
    public partial class NewSurvey : ContentPage
    {
        public string SurveyTitle {get;set;}
        public string MeasurementType { get; set; }
        public int SurveyId { get; set; }
        public double Lat { get; set; }
        public double Long { get; set; }
        FireBaseHelper firebaseHelper = new FireBaseHelper();

        public NewSurvey(int surveyId)
        {
            InitializeComponent();
            BindingContext = new NewSurveyViewModel();
            SurveyId = surveyId;
            Title = ("Survey Defects    "+ "Survey Id: "+SurveyId);

        }

        private async void SaveAndExit_Clicked(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new MainPage());
        }

        private async void NavigateButton_OnClickedBtn2mm(object sender, EventArgs e)
        {
            SurveyTitle = "Fine Crack < 2mm";
            MeasurementType = "Area(Msq)";
            var allDefects = await firebaseHelper.GetAllSurveyDefects();
            await Navigation.PushAsync(new SurveyInfo(SurveyTitle, MeasurementType,
SurveyId, allDefects, null));
        }

        private async void NavigateButton_OnClickedBtnBc(object sender, EventArgs e)
        {
            SurveyTitle = "Block Crack";
            MeasurementType = "Area(Msq)";
            var allDefects = await firebaseHelper.GetAllSurveyDefects();
            await Navigation.PushAsync(new SurveyInfo(SurveyTitle, MeasurementType,
SurveyId, allDefects, null));
        }

        private async void NavigateButton_OnClickedBtnCc(object sender, EventArgs e)
        {
            SurveyTitle = "Crocodile";
            MeasurementType = "Area(Msq)";
            var allDefects = await firebaseHelper.GetAllSurveyDefects();
            await Navigation.PushAsync(new SurveyInfo(SurveyTitle, MeasurementType,
SurveyId, allDefects, null));
        }

        private async void NavigateButton_OnClickedBtnDc(object sender, EventArgs e)
        {
            SurveyTitle = "Diagonal";
            MeasurementType = "Length";
            var allDefects = await firebaseHelper.GetAllSurveyDefects();
            await Navigation.PushAsync(new SurveyInfo(SurveyTitle, MeasurementType,
SurveyId, allDefects, null));
        }

        private async void NavigateButton_OnClickedBtnLs(object sender, EventArgs e)
        {
            SurveyTitle = "Longitudinal";
            MeasurementType = "Length";
            var allDefects = await firebaseHelper.GetAllSurveyDefects();
            await Navigation.PushAsync(new SurveyInfo(SurveyTitle, MeasurementType,
SurveyId, allDefects, null));
        }

        private async void NavigateButton_OnClickedBtnSc(object sender, EventArgs e)
        {
            SurveyTitle = "Slippage";
            MeasurementType = "Area(Msq)";
            var allDefects = await firebaseHelper.GetAllSurveyDefects();
            await Navigation.PushAsync(new SurveyInfo(SurveyTitle, MeasurementType,
SurveyId, allDefects, null));
        }

        private async void NavigateButton_OnClickedBtnTc(object sender, EventArgs e)
        {
            SurveyTitle = "Transverse";
            MeasurementType = "Area(Msq)";
            var allDefects = await firebaseHelper.GetAllSurveyDefects();
            await Navigation.PushAsync(new SurveyInfo(SurveyTitle, MeasurementType,
SurveyId, allDefects, null));
        }

        private async void NavigateButton_OnClickedBtnCr(object sender, EventArgs e)
        {
            SurveyTitle = "Corrugation";
            MeasurementType = "Area(Msq)";
            var allDefects = await firebaseHelper.GetAllSurveyDefects();
            await Navigation.PushAsync(new SurveyInfo(SurveyTitle, MeasurementType,
SurveyId, allDefects, null));
        }

        private async void NavigateButton_OnClickedBtnDw(object sender, EventArgs e)
        {
            SurveyTitle = "With Cracks";
            MeasurementType = "Area(Msq)";
            var allDefects = await firebaseHelper.GetAllSurveyDefects();
            await Navigation.PushAsync(new SurveyInfo(SurveyTitle, MeasurementType,
SurveyId, allDefects, null));
        }

        private async void NavigateButton_OnClickedBtnDp(object sender, EventArgs e)
        {
            SurveyTitle = "Without Cracks";
            MeasurementType = "Area(Msq)";
            var allDefects = await firebaseHelper.GetAllSurveyDefects();
            await Navigation.PushAsync(new SurveyInfo(SurveyTitle, MeasurementType,
SurveyId, allDefects, null));
        }

        private async void NavigateButton_OnClickedBtnRu(object sender, EventArgs e)
        {
            SurveyTitle = "Rutting";
            MeasurementType = "Length Depth (m/mm)";
            var allDefects = await firebaseHelper.GetAllSurveyDefects();
            await Navigation.PushAsync(new SurveyInfo(SurveyTitle, MeasurementType,
SurveyId, allDefects, null));
        }

        private async void NavigateButton_OnClickedBtnSv(object sender, EventArgs e)
        {
            SurveyTitle = "Shoving";
            MeasurementType = "Length Depth (m/mm)";
            var allDefects = await firebaseHelper.GetAllSurveyDefects();
            await Navigation.PushAsync(new SurveyInfo(SurveyTitle, MeasurementType,
SurveyId, allDefects, null));
```

```
        }

        private async void NavigateButton_OnClickedBtnFl(object sender, EventArgs e)
        {
            SurveyTitle = "Flushing";
            MeasurementType = "Area(Msq)";
            var allDefects = await firebaseHelper.GetAllSurveyDefects();
            await Navigation.PushAsync(new SurveyInfo(SurveyTitle, MeasurementType,
SurveyId, allDefects, null));
        }

        private async void NavigateButton_OnClickedBtnPo(object sender, EventArgs e)
        {
            SurveyTitle = "Polishing";
            MeasurementType = "Area(Msq)";
            var allDefects = await firebaseHelper.GetAllSurveyDefects();
            await Navigation.PushAsync(new SurveyInfo(SurveyTitle, MeasurementType,
SurveyId, allDefects, null));
        }

        private async void NavigateButton_OnClickedBtnRw(object sender, EventArgs e)
        {
            SurveyTitle = "Ravelling";
            MeasurementType = "Area(Msq)";
            var allDefects = await firebaseHelper.GetAllSurveyDefects();
            await Navigation.PushAsync(new SurveyInfo(SurveyTitle, MeasurementType,
SurveyId, allDefects, null));
        }

        private async void NavigateButton_OnClickedBtnLst(object sender, EventArgs e)
        {
            SurveyTitle = "Loss of Stone Chippings";
            MeasurementType = "Area(Msq)";
            var allDefects = await firebaseHelper.GetAllSurveyDefects();
            await Navigation.PushAsync(new SurveyInfo(SurveyTitle, MeasurementType,
SurveyId, allDefects, null));
        }

        private async void NavigateButton_OnClickedBtnPc(object sender, EventArgs e)
        {
            SurveyTitle = "Patching";
            MeasurementType = "Area(Msq)";
            var allDefects = await firebaseHelper.GetAllSurveyDefects();
            await Navigation.PushAsync(new SurveyInfo(SurveyTitle, MeasurementType,
SurveyId, allDefects, null));
        }
        private async void NavigateButton_OnClickedBtnPh(object sender, EventArgs e)
        {
            SurveyTitle = "Potholes";
            MeasurementType = "Area(Msq)";
            var allDefects = await firebaseHelper.GetAllSurveyDefects();
            await Navigation.PushAsync(new SurveyInfo(SurveyTitle, MeasurementType,
SurveyId, allDefects, null));
        }

        private async void NavigateButton_OnClickedBtnHo(object sender, EventArgs e)
        {
            SurveyTitle = "Hazards";
            MeasurementType = "Area(Msq)";
            var allDefects = await firebaseHelper.GetAllSurveyDefects();
            await Navigation.PushAsync(new SurveyInfo(SurveyTitle, MeasurementType,
SurveyId, allDefects, null));
        }
    }
}
```

2.4.7 PreviousSurveys.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using McurdyAssociates.Helper;
using McurdyAssociates.Model;
using Xamarin.Forms;

namespace McurdyAssociates.Views
{
   public partial class PreviousSurveys : ContentPage
   {
      FireBaseHelper firebaseHelper = new FireBaseHelper();
      public int SelectedSurveyId { get; set; }
      public PreviousSurveys(List<Model.NewSurveyModel> allSurveys)
      {
         InitializeComponent();
         List<Model.NewSurveyModel> ListOfSurveys = new
List<Model.NewSurveyModel>();

         allSurveys = allSurveys.OrderByDescending(a => a.NewSurveyID).ToList();

         lstSurveys.ItemsSource = allSurveys;


      }
      protected async override void OnAppearing()
      {

         base.OnAppearing();
         var allPersons = await firebaseHelper.GetAllSurveys();
         lstSurveys.SelectedItem = allPersons;
      }



      private async void NavigateButton_OnClicked(object sender, EventArgs e)
      {

         try
         {
            var selectedItem = (NewSurveyModel)lstSurveys.SelectedItem;
            SelectedSurveyId = selectedItem.NewSurveyID;
            await Navigation.PushAsync(new NewSurvey(SelectedSurveyId));
         }
         catch (InvalidCastException)
```

```
            {

                await DisplayAlert("", "Item Not Selected from List", "Ok");
            }

        }

    }
}
```

2.4.8 SelectLocation.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Net.Http;
using System.Threading.Tasks;
using Newtonsoft.Json;
using Xamarin.Essentials;
using Xamarin.Forms;
using Xamarin.Forms.Maps;

namespace McurdyAssociates.Views
{
    public partial class SelectLocation : ContentPage
    {
        public string SubArea { get; set; }
        public string SubLocality { get; set; }
        public string Street { get; set; }
        public int CurreentSurveyId { get; set; }
        public double Lat { get; set; }
        public double Long { get; set; }

        public SelectLocation(double lat, double longitude, string street, string subArea, string
subLocality, int currentSurveyID)
        {
            InitializeComponent();
            map.MoveToRegion(MapSpan.FromCenterAndRadius(
                new Position(lat, longitude),
                Distance.FromMiles(.2)));
            Lat = lat;
            Long = longitude;

            map.Pins.Add(new Pin
            {
                Type = PinType.Generic,
                Label = street,
                Position = new Position(lat, longitude)
            });

            SubArea = subArea;
            Street = street;
            SubLocality = subLocality;
            CurreentSurveyId = currentSurveyID;

        }


        private async void NavigateButton_OnClicked(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new SurveyDetails(SubLocality, Street , SubArea, 
CurreentSurveyId, Lat, Long));
        }

        private async void NavigateWrong_Clicked(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new WrongLocation(CurreentSurveyId, SubLocality, 
Street, SubArea));
        }

    }
}
```

2.4.9 SurveyDetail.xaml.cs

```
using System;
using System.Collections.Generic;
using McurdyAssociates.Helper;
using Xamarin.Forms;
using System.Linq;
using McurdyAssociates.Model;
using System.Threading.Tasks;
using System.Collections.ObjectModel;

namespace McurdyAssociates.Views
{
    public partial class SurveyDetails : ContentPage
    {
        FireBaseHelper firebaseHelper = new FireBaseHelper();
        public int CurrentSurveyId { get; set; }
        public String Street { get; set; }
        public double Lat { get; set; }
        public double Long { get; set; }

        public List<string> list;
        public List<string> List
        {
            get { return list; }
            set { list = value; }
        }
        public SurveyDetails(string subLocality, string street, string subArea, int
curreentSurveyId, double lat, double @long)
        {
            InitializeComponent();
            latlab.Text = lat.ToString();
            longlab.Text = @long.ToString();
            CurrentSurveyId = curreentSurveyId;
            Street = subArea;
            GetNames();
            Lat = lat;
            Long = @long;


        }

        private async void GetNames()
        {
            var allEmp = await firebaseHelper.GetAllEmployees();
            List = new List<string>();
            foreach(var item in allEmp)
            {
                var selectedName = item.EmployeeName.ToString();
                var selectedPosition = item.Position.ToString();
                List.Add(selectedName+" ("+selectedPosition+")");

            }
            completedBy.ItemsSource = List;
        }

        private async void NavigateButton_OnClicked(object sender, EventArgs e)
        {


            if (completedBy.SelectedIndex == -1 || Weather.SelectedIndex == -1)
            {
                await DisplayAlert("Error", "Not all fields are completed", "OK");
            }
            else
            {
                var employeeDb = completedBy.SelectedItem.ToString();
                var weatherDb = Weather.SelectedItem.ToString();
                await Navigation.PushAsync(new ExtraSurveyDetails(CurrentSurveyId,
employeeDb, Street, weatherDb, Lat, Long));

            }
        }
    }
}
```

2.4.10 SurveyInfo.xaml.cs

```
using System;
using System.Collections.Generic;
using McurdyAssociates.ViewModels;
using Plugin.Media;
using Xamarin.Forms;
using System.Threading.Tasks;
using McurdyAssociates.Helper;
using System.Linq;
using McurdyAssociates.Views;
using McurdyAssociates.Model;
using System.IO;
using Firebase.Storage;
using Plugin.Media.Abstractions;
using Xamarin.Essentials;
using Plugin.AudioRecorder;
using System.Text;

namespace McurdyAssociates
{
    public partial class SurveyInfo : ContentPage
    {
        public int CurrentDefectID { get; set; }
        public int SurveyId { get; set; }
        public string DefectName { get; set; }
        public string Measurement { get; set; }
        public string Value { get; set; }
        public string Chainage { get; set; }
        public string ChainageToUpdate { get; set; }
        public string ValueToUpdate { get; set; }
        public string ImgUrl { get; set; }
        public double Long { get; set; }
        public double Lat { get; set; }
        public string Comments { get; set; }
        public string CommentsToUpdate { get; set; }

        AudioRecorderService recorder;
        AudioPlayer player;
        bool isTimerRunning = false;
        int seconds = 0, minutes = 0;
        List<Model.SurveyDefect> SortedListByChainage { get; set; }
        List<Model.SurveyDefect> ListDefect { get; set; }
        MediaFile photo;
        MediaFile newPhoto;
        System.IO.Stream SavedPhoto { get; set; }
        Stream Recording { get; set;}
        FireBaseHelper firebaseHelper = new FireBaseHelper();
        public SurveyInfo(string surveyTitle, string measurementType, int surveyId,
List<Model.SurveyDefect> allDefects, System.IO.Stream stream)
        {
            InitializeComponent();
            SavedPhoto = stream;
            Gridlay.IsVisible = false;
            prevDefectLabel.IsVisible = false;
            listOfDefects.IsVisible = true;
            PhotoImage.IsVisible = false;
            recordLayout.IsVisible = false;
            title.Text = surveyTitle;
            SurveyId = surveyId;
            DefectName = surveyTitle;
            Measurement = measurementType;
            measurmentEntry.Text = measurementType;
            BindingContext = new SurveyInfoViewModel();
            Title = surveyTitle;

            btnCamera.Clicked += CameraButtonClicked;
            Record.Clicked += DisplayRecordLayout;
            ListDefect = allDefects;
            List<Model.SurveyDefect> ListNameDefects = allDefects.Where(a =>
a.DefectName.Equals(surveyTitle)).ToList();
            List<Model.SurveyDefect> ListDefectIds = ListNameDefects.Where(a =>
a.SurveyId.Equals(surveyId)).ToList();
            SortedListByChainage = ListDefectIds.OrderByDescending(a =>
a.Chainage).ToList();
            lstDefects.ItemsSource = SortedListByChainage;
            if (SortedListByChainage.Count>0)
            {
                Gridlay.IsVisible = true;
                prevDefectLabel.IsVisible = true;
                listOfDefects.IsVisible = false;
            }
            List<Model.SurveyDefect> GetSingleDefect = SortedListByChainage.Where(a =>
a.SurveyDefectId.Equals(CurrentDefectID)).ToList();
            ChainageToUpdate = GetSingleDefect.Select(a => a.Chainage).FirstOrDefault();
            ValueToUpdate = GetSingleDefect.Select(a => a.Value).FirstOrDefault();
            if (stream != null)
            {
                PhotoImage.Source = ImageSource.FromStream(() => stream);
                PhotoImage.IsVisible = true;
                btnCamera.IsVisible = false;
            }
            recorder = new AudioRecorderService
            {
```

```
            StopRecordingAfterTimeout = true,
            TotalAudioTimeout = TimeSpan.FromSeconds(15),
            AudioSilenceTimeout = TimeSpan.FromSeconds(2)
        };

        player = new AudioPlayer();
        player.FinishedPlaying += Finish_Playing;

    }

    private async void DisplayRecordLayout(object sender, EventArgs e)
    {
        recordLayout.IsVisible = true;
        Record.IsVisible = false;
    }

    void Finish_Playing(object sender, EventArgs e)
    {
        bntRecord.IsEnabled = true;
        bntRecord.BackgroundColor = Color.FromHex("#7cbb45");
        bntPlay.IsEnabled = true;
        bntPlay.BackgroundColor = Color.FromHex("#7cbb45");
        bntStop.IsEnabled = false;
        bntStop.BackgroundColor = Color.Silver;
        lblSeconds.Text = "00";
        lblMinutes.Text = "00";
    }

    async void Record_Clicked(object sender, EventArgs e)
    {
        if (!recorder.IsRecording)
        {
            seconds = 0;
            minutes = 0;
            isTimerRunning = true;
            Device.StartTimer(TimeSpan.FromSeconds(1), () => {
                seconds++;

                if (seconds.ToString().Length == 1)
                {
                    lblSeconds.Text = "0" + seconds.ToString();
                }
                else
                {
                    lblSeconds.Text = seconds.ToString();
                }
                if (seconds == 60)
                {
                    minutes++;
                    seconds = 0;

                    if (minutes.ToString().Length == 1)
                    {
                        lblMinutes.Text = "0" + minutes.ToString();
                    }
                    else
                    {
                        lblMinutes.Text = minutes.ToString();
                    }

                    lblSeconds.Text = "00";
                }
                return isTimerRunning;
            });

            //
            recorder.StopRecordingOnSilence = IsSilence.IsToggled;
            var audioRecordTask = await recorder.StartRecording();

            bntRecord.IsEnabled = false;
            bntRecord.BackgroundColor = Color.Silver;
            bntPlay.IsEnabled = false;
            bntPlay.BackgroundColor = Color.Silver;
            bntStop.IsEnabled = true;
            bntStop.BackgroundColor = Color.FromHex("#7cbb45");

            await audioRecordTask;
        }
    }

    async void Stop_Clicked(object sender, EventArgs e)
    {
        StopRecording();
        await recorder.StopRecording();
        Recording = recorder.GetAudioFileStream();
    }

    void StopRecording()
    {
        isTimerRunning = false;
        bntRecord.IsEnabled = true;
        bntRecord.BackgroundColor = Color.FromHex("#7cbb45");
        bntPlay.IsEnabled = true;
        bntPlay.BackgroundColor = Color.FromHex("#7cbb45");
```

```
            bntStop.IsEnabled = false;
            bntStop.BackgroundColor = Color.Silver;
            lblSeconds.Text = "00";
            lblMinutes.Text = "00";
        }
        void Play_Clicked(object sender, EventArgs e)
        {
            try
            {
                var filePath = recorder.GetAudioFilePath();
                Recording = recorder.GetAudioFileStream();

                if (filePath != null)
                {
                    StopRecording();
                    player.Play(filePath);
                }
            }
            catch (Exception ex)
            {
                throw ex;
            }
        }

        private async void CameraButtonClicked(object sender, EventArgs e)
        {
            photo = await Plugin.Media.CrossMedia.Current.TakePhotoAsync(new
Plugin.Media.Abstractions.StoreCameraMediaOptions() { });

            if (photo != null)
            {
                PhotoImage.Source = ImageSource.FromStream(() => { return photo.GetStream();
});
                PhotoImage.IsVisible = true;
                btnCamera.IsVisible = false;
                SavedPhoto = photo.GetStream();
            }
        }

        private async void SaveAndContinue_Clicked(object sender, EventArgs e)
        {
            if (ChainVal.Text == null)
            {
                await DisplayAlert("Error", "Not all fields are completed", "OK");
            }
            else if (mesVal.Text == null)
            {
                await DisplayAlert("Error", "Not all fields are completed", "OK");
            }
            else
            {
                var action = await DisplayAlert("Save?", "Are you sure", "Yes", "No");
                if (action)
                {
                    var location = await Geolocation.GetLocationAsync();
                    if (location != null)
                    {
                        Lat = location.Latitude;
                        Long = location.Longitude;
                    }
                    DefectName = title.Text;
                    Measurement = measurmentEntry.Text;
                    Value = mesVal.Text;
                    Chainage = ChainVal.Text;
                    Comments = comentry.Text;
                    var LastId = await firebaseHelper.GetAllSurveyDefects();
                    CurrentDefectID = LastId.Max(a => a.SurveyDefectId) + 1;
                    await firebaseHelper.AddDefect(Convert.ToInt32(CurrentDefectID),
Convert.ToInt32(SurveyId), Chainage, DefectName, Measurement, Value, Long, Lat,
Comments);
                    if (SavedPhoto != null)
                    {
                        await StoreFirstImage(SavedPhoto);
                    }
                    if(Recording != null)
                    {
                        await StoreRecording(Recording);
                    }

                    await Navigation.PushAsync(new NewSurvey(SurveyId));
                }

            }


        }
        public async Task<string> StoreFirstImage(System.IO.Stream imageStream)
        {
            var stroageImage = await new FirebaseStorage("mccurdyassociates-
9d006.appspot.com")
                .Child("McCurdyAssociates")
                .Child(CurrentDefectID+".1"+".png")
                .PutAsync(imageStream);
            ImgUrl = stroageImage;
            return ImgUrl;
        }

        public async Task<string> StoreRecording(Stream imageStream)
        {
            var stroageImage = await new FirebaseStorage("mccurdyassociates-
9d006.appspot.com")
                .Child("McCurdyAssociates")
                .Child("Audio")
                .Child(CurrentDefectID + ".1" + ".wav")
                .PutAsync(imageStream);
            ImgUrl = stroageImage;
            return ImgUrl;
        }

        private async void ExpandImage(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new ImageDisplay(photo.GetStream(), DefectName,
Measurement, SurveyId, ListDefect, Long, Lat));
        }


        private async void Update_Clicked(object sender, EventArgs e)
        {
            try
            {

                var selectedItem = (SurveyDefect)lstDefects.SelectedItem;
                CurrentDefectID = selectedItem.SurveyDefectId;

                lstDefects.ItemsSource = SortedListByChainage;
                List<Model.SurveyDefect> GetSingleDefect = SortedListByChainage.Where(a =>
a.SurveyDefectId.Equals(CurrentDefectID)).ToList();
                ChainageToUpdate = GetSingleDefect.Select(a => a.Chainage).FirstOrDefault();
                ValueToUpdate = GetSingleDefect.Select(a => a.Value).FirstOrDefault();
                CommentsToUpdate = GetSingleDefect.Select(a =>
a.Comments).FirstOrDefault();


                await Navigation.PushAsync(new UpdatePage(DefectName, SurveyId,
Measurement, ChainageToUpdate, CurrentDefectID, ValueToUpdate,
CommentsToUpdate));
            }
            catch (NullReferenceException)
            {
                await DisplayAlert("", "Item Not Selected from List", "Ok");
            }
        }

        private async void Delete_Clicked(object sender, EventArgs e)
        {
            try
            {
                var selectedItem = (SurveyDefect)lstDefects.SelectedItem;
                CurrentDefectID = selectedItem.SurveyDefectId;

                var action = await DisplayAlert("Delete?", "Are you sure", "Yes", "No");
                if (action)
                {

                    await firebaseHelper.DeleteDefect(Convert.ToInt32(CurrentDefectID));
                    var allDefects = await firebaseHelper.GetAllSurveyDefects();
                    //await Navigation.PushAsync(new SurveyInfo(DefectName, Measurement,
SurveyId, allDefects));
                    var vUpdatedPage = new SurveyInfo(DefectName, Measurement, SurveyId,
allDefects, null);
                    Navigation.InsertPageBefore(vUpdatedPage, this);
                    Navigation.PopAsync();
                }
            }
            catch (NullReferenceException)
            {
                await DisplayAlert("", "Item Not Selected from List", "Ok");
            }


        }
    }
}
```

2.4.11 UpdateEmployee.xaml.cs

```
using System;
using System.Collections.Generic;
using McurdyAssociates.Helper;
using Xamarin.Forms;

namespace McurdyAssociates.Views
{
    public partial class UpdateEmployee : ContentPage
    {
        FireBaseHelper firebaseHelper = new FireBaseHelper();
        public int EmployeeId { get; set; }
        public string NewEmployeeName { get; set; }
        public string NewPosition;

        public UpdateEmployee(int selectedEmployeeID, string employeeNameToUpdate,
string positionToUpdate)
        {
            InitializeComponent();
            PositionVal.Text = positionToUpdate;
            NameVal.Text = employeeNameToUpdate;
            EmployeeId = selectedEmployeeID;

        }

        private async void Update_Clicked(object sender, EventArgs e)
        {
            NewEmployeeName = NameVal.Text;
            NewPosition = PositionVal.Text;
            var action = await DisplayAlert("Update?", "Are you sure", "Yes", "No");
            if (action)
            {
                await firebaseHelper.UpdateEmployee(Convert.ToInt32(EmployeeId),
NewEmployeeName, NewPosition);
                await DisplayAlert("Employee Updated", "", "ok");
                await Navigation.PushAsync(new MainPage());
            }
        }
    }
}
```

2.4.12 UpdatePage.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Net.Http;
using System.Threading.Tasks;
using Firebase.Storage;
using McurdyAssociates.Helper;
using Xamarin.Forms;

namespace McurdyAssociates.Views
{
    public partial class UpdatePage : ContentPage
    {
        public string DefectName { get; set; }
        public int SurveyId { get; set; }
        public string Measurement { get; set; }
        public string Chainage { get; set; }
        public int CurrentDefectId { get; set; }
        public string Value { get; set; }
        public string Comments { get; set; }
        FireBaseHelper firebaseHelper = new FireBaseHelper();

        public UpdatePage(string defectName, int surveyId, string measurement, string
chainage, int currentDefectID, string value, string commentsToUpdate)
        {
            InitializeComponent();
            title.Text = defectName;
            PhotoImages.IsVisible = false;
            measurmentEntry.Text = measurement;
            ChainVal.Text = chainage;
            mesVal.Text = value;
            comentry.Text = commentsToUpdate;
            DefectName = defectName;
            SurveyId = surveyId;
            Measurement = measurement;
            Chainage = chainage;
            CurrentDefectId = currentDefectID;
            Value = value;
            GetImage1();


        }

        private async void GetImage1()
        {
            HttpClient client = new HttpClient();
            var response =  await client.GetAsync("https://firebasestorage.googleapis.com/v0/b/
mccurdyassociates-9d006.appspot.com/o/McCurdyAssociates
%2F"+CurrentDefectId+".1.png?alt=media&token=4775e01a-5adc-4edd-a819-
a9c40b74fca8");
            var imageStrem = await response.Content.ReadAsStreamAsync();


            if (imageStrem != null)
            {
                PhotoImages.IsVisible = true;
            }
            else
            {
                PhotoImages.IsVisible = false;
            }

            PhotoImages.Source = ImageSource.FromStream(() => imageStrem);

        }


        private async void Update_Clicked(object sender, EventArgs e)
        {
            if (ChainVal.Text == "")
            {
                await DisplayAlert("Error", "Not all fields are completed", "OK");
            }
            else if (mesVal.Text == "")
            {
                await DisplayAlert("Error", "Not all fields are completed", "OK");
            }
            else
            {
                Chainage = ChainVal.Text;
                Value = mesVal.Text;
                Comments = comentry.Text;
                var action = await DisplayAlert("Update?", "Are you sure", "Yes", "No");
                if (action)
                {
                    await firebaseHelper.UpdateDefect(Convert.ToInt32(CurrentDefectId),
Convert.ToInt32(SurveyId), DefectName, Chainage, Measurement, Value, Comments);
                    await Navigation.PushAsync(new NewSurvey(SurveyId));
                }
            }

        }
    }
```

```
}
```

2.4.13 WrongLocation.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using McurdyAssociates.CustomRenderers;
using Xamarin.Essentials;
using Xamarin.Forms;
using Xamarin.Forms.Maps;
using Map = Xamarin.Forms.Maps.Map;

namespace McurdyAssociates.Views
{
    public partial class WrongLocation : ContentPage
    {
        public double Lat { get; set; }
        public double Long { get; set; }
        public int CurrentSurveyId { get; set; }
        public string SubLocality { get; set; }
        public string Street { get; set; }
        public string SubArea { get; set; }
        public WrongLocation(int curreentSurveyId, string subLocality, string street, string
subArea)
        {
            InitializeComponent();
            GoogleMapScreenLayout();
            CurrentSurveyId = curreentSurveyId;
            SubLocality = subLocality;
            Street = street;
            SubArea = subArea;


        }

        #region initialize
        Label lblLat = new Label();
        Label lblLong = new Label();

        #endregion
        Xamarin.Forms.Maps.Map map = new Map();
        CustomMapForClickEvent customMap = new CustomMapForClickEvent();


        public void GoogleMapScreenLayout()
        {
```

```
#region label latitude
lblLat = new Label
{
   Text = "Lat",
   FontSize = 15,
   FontAttributes = FontAttributes.Bold,
   TextColor = Color.Black
};
StackLayout slLblLat = new StackLayout
{
   Children = { lblLat },
   Orientation = StackOrientation.Horizontal,
   Padding = new Thickness(10, 10, 10, 10)
};
#endregion

#region label longitude
lblLong = new Label
{
   Text = "Long",
   FontSize = 15,
   FontAttributes = FontAttributes.Bold,
   TextColor = Color.Black
};
StackLayout slLblLong = new StackLayout
{
   Children = { lblLong },
   Orientation = StackOrientation.Horizontal,
   Padding = new Thickness(10, 10, 10, 10)
};
#endregion


#region custom map
customMap = new CustomMapForClickEvent
{
   WidthRequest = 300,
   HeightRequest = 500,
   IsShowingUser = true,
   MapType = MapType.Street

};
customMap.Tap += CustomMap_Tap;
customMap.MoveToRegion(MapSpan.FromCenterAndRadius(
   new Position(54.36, -7.55),
   Distance.FromMiles(50)));
StackLayout slCustomMap = new StackLayout
```

```
{
   Children = { customMap },
   Orientation = StackOrientation.Horizontal,
   Padding = new Thickness(10, 10, 10, 10)
};
#endregion


#region button for current location
Button btnLatLong = new Button
{
   Text = "Next",
   HorizontalOptions = LayoutOptions.Center,
   VerticalOptions = LayoutOptions.Center,
   BackgroundColor = Color.FromHex("4690FB"),
   TextColor = Color.White,
   BorderRadius = 10,
   WidthRequest = 300,
   HeightRequest = 75,
   FontAttributes = FontAttributes.Bold
};
btnLatLong.Clicked += BtnLatLong_Clicked;

StackLayout slBtnLatLong = new StackLayout
{
   Children = { btnLatLong },
   Orientation = StackOrientation.Horizontal,
   HorizontalOptions = LayoutOptions.Center,
   VerticalOptions = LayoutOptions.Center,

   Padding = new Thickness(20, 20, 20, 20)
};

#endregion

#region stack layouts and contents
StackLayout slGoogleMapScreen = new StackLayout
{
   Children = { slLblLat, slLblLong,slCustomMap, slBtnLatLong },
   HorizontalOptions = LayoutOptions.FillAndExpand,

   VerticalOptions = LayoutOptions.FillAndExpand,
   Padding = new Thickness(0, 0, 0, 0),
   BackgroundColor = Color.LemonChiffon
};

ScrollView svGoogleMapScreen = new ScrollView
```

```
            {
                Content = slGoogleMapScreen
            };

            Content = svGoogleMapScreen;
            #endregion
        }

        private async void BtnLatLong_Clicked(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new SurveyDetails(SubLocality, Street, SubArea,
CurrentSurveyId, Lat, Long));
        }


        private void CustomMap_Tap(object sender, TapEventArgs e)
        {
            Lat = e.Position.Latitude;
            Long = e.Position.Longitude;

            lblLat.Text = Lat.ToString();
            lblLong.Text = Long.ToString();

            Pin pin = new Pin
            {
                Type = PinType.SavedPin,
                Position = e.Position,
                Label = "Position"
            };

            customMap.Pins.Add(pin);
        }
    }
}
```

3.Xaml Code

3.1 Views

3.1.1 EmployeesPage.xaml

```
<?xml version="1.0" encoding="UTF-8"?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             Title="All Employee's"
             x:Class="McurdyAssociates.Views.EmployeesPage">
    <ScrollView>
    <StackLayout BackgroundColor="LemonChiffon">

       <StackLayout Padding="50,20,0,20">
           <Image Source="CompanyLogo"
                 HorizontalOptions="Center"
                 VerticalOptions="Start"
                 HeightRequest="80"/>
        </StackLayout>
        <StackLayout>
           <Button x:Name="btnAdd"
               Grid.Column="4"
               HorizontalOptions="Center"
               CornerRadius="15"
               BackgroundColor="Green"
               HeightRequest="40"
               WidthRequest="100"
               Text="Add Employee"
               TextColor="White"
               Clicked="NewEmployeePage"
               FontSize="Micro" />
        </StackLayout>

 <StackLayout x:Name="Gridlay" Padding="0,0,0,0">
           <Grid>
               <Grid.RowDefinitions>
                   <RowDefinition Height="Auto" />
               </Grid.RowDefinitions>
               <Grid.ColumnDefinitions>
                   <ColumnDefinition Width="20*" />
                   <ColumnDefinition Width="20*" />
                   <ColumnDefinition Width="20*" />
                   <ColumnDefinition Width="20*" />
                   <ColumnDefinition Width="20*" />
               </Grid.ColumnDefinitions>
```

```
            <Label  Grid.Column="1" Grid.Row="0" FontSize="Medium"
HorizontalOptions="Center" FontAttributes="Bold" Text="Employee Id"/>
            <Label Grid.Column="2" Grid.Row="0" FontSize="Title"
HorizontalOptions="Center" FontAttributes="Bold" Text="Name"/>
            <Label Grid.Column="3" Grid.Row="0" FontSize="Title"
HorizontalOptions="Center" FontAttributes="Bold" Text="Position"/>
         </Grid>


      <ListView x:Name="lstEmp">
         <ListView.ItemTemplate>
            <DataTemplate>
               <ViewCell>
                  <Grid>
                     <Grid.RowDefinitions>
                        <RowDefinition Height="Auto" />
                     </Grid.RowDefinitions>
                     <Grid.ColumnDefinitions>
                        <ColumnDefinition Width="20*" />
                        <ColumnDefinition Width="20*" />
                        <ColumnDefinition Width="20*" />
                        <ColumnDefinition Width="20*" />
                        <ColumnDefinition Width="20*" />
                     </Grid.ColumnDefinitions>

                     <Button x:Name="btnDelete"
                           Grid.Column="0"
                           CornerRadius="15"
                           HorizontalOptions="Center"
                           BackgroundColor="Red"
                           HeightRequest="35"
                           WidthRequest="75"
                           Text="Delete"
                           TextColor="White"
                           Clicked="Delete_Clicked"
                           FontSize="Micro" />
                     <Label  Grid.Column="1" HorizontalOptions="Center" Text="{Binding
EmployeeID}"/>
                     <Label  Grid.Column="2" HorizontalOptions="Center" Text="{Binding
EmployeeName}"/>
                     <Label  Grid.Column="3" HorizontalOptions="Center" Text="{Binding
Position}"/>
                     <Button x:Name="btnUpdate"
                           Grid.Column="4"
                           CornerRadius="15"
                           HorizontalOptions="Center"
                           BackgroundColor="Green"
```

```
                                    HeightRequest="35"
                                    WidthRequest="75"
                                    Text="Update"
                                    TextColor="White"
                                    Clicked="Update_Clicked"
                                    FontSize="Micro" />
                        </Grid>
                      </ViewCell>
                  </DataTemplate>
              </ListView.ItemTemplate>

          </ListView>
          </StackLayout>
      </StackLayout>
      </ScrollView>
</ContentPage>
```

3.1.2 ExtraSurveyDetails.xaml

```
<?xml version="1.0" encoding="UTF-8"?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="McurdyAssociates.Views.ExtraSurveyDetails"
             Title="Survey Details">
   <StackLayout BackgroundColor="LemonChiffon">

       <!-- Place new controls here -->
      <StackLayout Padding="50,20,0,0">
          <Image Source="CompanyLogo"
                 HorizontalOptions="Center"
                 VerticalOptions="Start"
                 HeightRequest="80"
                 />
       </StackLayout>

       <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="50,50,0,0">
          <Label Text="Surface Type"
                 FontSize="Medium"
                 FontAttributes="Bold"
                 TextColor="Black"
                 VerticalOptions="Start"/>
          <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="150,0,0,0">
              <Picker x:Name="SurfaceType" FontAttributes="Bold"
HorizontalOptions="FillAndExpand" WidthRequest="250" HeightRequest="35" Title="Please
Select Surface Type">
                 <Picker.Items>
                 <x:String>Hot Rolled Asphalt (HRA)</x:String>
                 <x:String>Stone Mastic Asphalt (SMA)</x:String>
                 <x:String>Asphalt Concrete (AC)</x:String>
                 <x:String>Surface Dressing (SD)</x:String>
                 <x:String>Porous Asphalt (PA)</x:String>
                 <x:String>High Friction Surfacing (HFS)</x:String>
                 <x:String>Bitumen Macadam (MB)</x:String>
                 <x:String>Other</x:String>
                 </Picker.Items>
              </Picker>


          </StackLayout>

       </StackLayout>
       <StackLayout x:Name="SurfaceTypeOther1" Orientation="Horizontal"
HorizontalOptions="Start" VerticalOptions="Start" Padding="310,20,0,0">
```

```
            <Entry x:Name="SurfaceTypeOther"
                 Placeholder="Type here for other"
                 WidthRequest="250"
                 HeightRequest="35"
                 />
        </StackLayout>

        <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="50,50,0,0">
          <Label Text="Direction"
               FontAttributes="Bold"
               TextColor="Black"
               FontSize="Medium"
               VerticalOptions="Start"/>
          <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="180,0,0,0">
            <Picker x:Name="Direction" FontAttributes="Bold"
HorizontalOptions="FillAndExpand" WidthRequest="250" HeightRequest="35" Title="Please
Select Section">
               <Picker.Items>
               <x:String>Northbound</x:String>
               <x:String>Southbound</x:String>
               <x:String>Eastbound</x:String>
               <x:String>Westbound</x:String>
               <x:String>Inbound</x:String>
               <x:String>Outbound</x:String>
               <x:String>Other</x:String>
               </Picker.Items>
            </Picker>
          </StackLayout>
        </StackLayout>
        <StackLayout x:Name="DirectionOther1" Orientation="Horizontal"
HorizontalOptions="Start" VerticalOptions="Start" Padding="310,20,0,0">
            <Entry x:Name="DirectionOther"
                 Placeholder="Type here for other"
                 WidthRequest="250"
                 HeightRequest="35"
                 />
        </StackLayout>

        <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="50,50,0,0">
          <Label Text="Lane"
               TextColor="Black"
               FontAttributes="Bold"
               FontSize="Medium"
               VerticalOptions="Start"/>
            <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="215,0,0,0">
                <Picker x:Name="Lane" FontAttributes="Bold" HorizontalOptions="FillAndExpand"
WidthRequest="250" HeightRequest="35" Title="Please Select Lane">
                    <Picker.Items>
                    <x:String>Lane 1 (Slow),</x:String>
                    <x:String>Lane 2</x:String>
                    <x:String>Lane 3</x:String>
                    <x:String>Lane 4,</x:String>
                    <x:String>Climbing Lane</x:String>
                    <x:String>Right-Turn Lane</x:String>
                    <x:String>Left-Turn Lane</x:String>
                    <x:String>Other</x:String>
                    </Picker.Items>
                </Picker>
            </StackLayout>
        </StackLayout>
        <StackLayout x:Name="LaneOther1" Orientation="Horizontal"
HorizontalOptions="Start" VerticalOptions="Start" Padding="310,20,0,0">
                <Entry x:Name="LaneOther"
                       Placeholder="Type here for other"
                       WidthRequest="250"
                       HeightRequest="35"
                       />
        </StackLayout>

        <StackLayout Orientation="Horizontal" HorizontalOptions="Center"
VerticalOptions="CenterAndExpand" >
            <Button x:Name="btnNew"
                 HorizontalOptions="End"
                 BackgroundColor="#03A0FA"
                 HeightRequest="75"
                 WidthRequest="300"
                 Text="Save And Start Survey"
                 TextColor="White"
                 FontAttributes="Bold"
                 CornerRadius="25"
                 Clicked="NavigateButton_OnClicked"
                 FontSize="Medium" />
        </StackLayout>
    </StackLayout>
</ContentPage>
```

3.1.3 ImageDisplay.xaml

```
<?xml version="1.0" encoding="UTF-8"?>
<ContentPage
    xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="McurdyAssociates.Views.ImageDisplay"
    Title="Your Image">
    <StackLayout BackgroundColor="LemonChiffon">

        <!-- Place new controls here -->
       <StackLayout Padding="50,20,0,0">
            <Image Source="CompanyLogo"
                   HorizontalOptions="Center"
                   VerticalOptions="Start"
                   HeightRequest="80"/>
        </StackLayout>
        <StackLayout HorizontalOptions="CenterAndExpand"
VerticalOptions="CenterAndExpand">
            <Image x:Name="Image"
                   HeightRequest="650"
                   WidthRequest="500"
                   />
            <Button x:Name="btn2mm"
                    HorizontalOptions="End"
                    BackgroundColor="#03A0FA"
                    HeightRequest="50"
                    WidthRequest="120"
                    Text="Retake"
                    TextColor="White"
                    CornerRadius="20"
                    FontAttributes="Bold"
                    Clicked="RetakeButton"
                    FontSize="Medium" />
        </StackLayout>
    </StackLayout>
</ContentPage>
```

3.1.4 MainPage.xaml

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

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:d="http://xamarin.com/schemas/2014/forms/design"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             mc:Ignorable="d"
             x:Class="McurdyAssociates.MainPage">
    <StackLayout BackgroundColor="LemonChiffon">


        <!-- Place new controls here -->
        <!--<Label Text="McCurdy Associates" FontSize="Title" HorizontalOptions="Center"
VerticalOptions="CenterAndExpand" />-->
        <StackLayout Padding="0,100,0,0">
        <Image Source="CompanyLogo"
               WidthRequest="{OnPlatform iOS=400, Android=350}"
               HorizontalOptions="Center"
               VerticalOptions="Center"/>
            </StackLayout>
                        <StackLayout Padding="0,50,0,150" HorizontalOptions="Center"
VerticalOptions="Center" BackgroundColor="LemonChiffon">
        <ContentView x:Name="popupLoadingView" BackgroundColor="LemonChiffon"
Padding="10, 0" IsVisible="false" AbsoluteLayout.LayoutBounds="0, 0, 1, 1"
AbsoluteLayout.LayoutFlags="All">
                <StackLayout VerticalOptions="Center" HorizontalOptions="Center">
                    <StackLayout Orientation="Vertical" HeightRequest="150" WidthRequest="150"
BackgroundColor="LemonChiffon">

                        <ActivityIndicator x:Name="activityIndicator" Margin="0,50,0,0"
VerticalOptions="Center" HorizontalOptions="Center" Color="Black" WidthRequest="30"
HeightRequest="30" ></ActivityIndicator>
                        <Label x:Name="lblLoadingText" TextColor="Black" VerticalOptions="Center"
HorizontalOptions="Center" VerticalTextAlignment="Center" Text="Loading..."></Label>
                    </StackLayout>
                </StackLayout>
            </ContentView>
         </StackLayout>
           <StackLayout Orientation="Horizontal" HorizontalOptions="Center"
VerticalOptions="Center">
                <Button x:Name="btnNew"
                    CornerRadius="25"
                    HorizontalOptions="End"
                    BackgroundColor = "#03A0FA"
                    HeightRequest="75"
                    WidthRequest="250"
                    Text="Start New Survey"
                    TextColor="White"
                    FontAttributes="Bold"
                    Clicked="NavigateButton_OnClicked"
                    FontSize="Medium" />
                <Button x:Name="btnPrev"
                    CornerRadius="25"
                    HorizontalOptions="End"
                    BackgroundColor="#03A0FA"
                    HeightRequest="75"
                    WidthRequest="250"
                    Text="View Previous Survey"
                    TextColor="White"
                    FontAttributes="Bold"
                    Clicked="PveviousSurvey_OnClicked"
                    FontSize="Medium" />
            </StackLayout>
         <StackLayout Padding="0" Orientation="Horizontal" HorizontalOptions="Center"
VerticalOptions="Start">
                    <Button x:Name="employeeList"
                          CornerRadius="25"
                          HorizontalOptions="End"
                          BackgroundColor="#03A0FA"
                          HeightRequest="75"
                          WidthRequest="250"
                          Text="All Employees"
                          TextColor="White"
                          FontAttributes="Bold"
                          Clicked="Employee_OnClicked"
                          FontSize="Medium" />
                </StackLayout>

    </StackLayout>

</ContentPage>
```

3.1.5 NewEmployeePage.xaml

```
<?xml version="1.0" encoding="UTF-8"?>
<ContentPage
   xmlns="http://xamarin.com/schemas/2014/forms"
   xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
   x:Class="McurdyAssociates.Views.NewEmployeePage"
   xmlns:prism="clr-namespace:Prism.Mvvm;assembly=Prism.Forms"
          Title="New Employee">
   <ScrollView>
   <StackLayout BackgroundColor="LemonChiffon">

      <!-- Place new controls here -->
      <StackLayout Padding="0,20,0,60">
         <Image Source="CompanyLogo"
               HorizontalOptions="Center"
               VerticalOptions="Start"
               HeightRequest="80"/>
      </StackLayout>

      <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="20,20,0,0">
         <Label Text="Name"
               FontSize="Medium"
               FontAttributes="Bold"
               TextColor="Black"
               VerticalOptions="Start"/>
         <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="169,0,0,0">
         <Entry WidthRequest="200"
               HeightRequest="35"

               x:Name="NameVal"/>
         </StackLayout>
      </StackLayout>
      <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="20,20,0,0">
         <Label Text="Position"
               FontSize="Medium"
               FontAttributes="Bold"
               TextColor="Black"
               VerticalOptions="Start"/>
         <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="150,0,0,0">
         <Entry WidthRequest="200"
               HeightRequest="35"
               x:Name="PositionVal"
               />
```

```
            </StackLayout>
        </StackLayout>

<!--Save-->
        <StackLayout Orientation="Horizontal" HorizontalOptions="CenterAndExpand"
Padding="80,60,75,0" >
            <Button x:Name="btnSave"
                    HorizontalOptions="End"
                    BackgroundColor="#03A0FA"
                    HeightRequest="60"
                    WidthRequest="300"
                    CornerRadius="25"
                    FontAttributes="Bold"
                    Text="Save Employee"
                    TextColor="White"
                    Clicked="Save_Clicked"
                    FontSize="Medium" />
        </StackLayout>


    </StackLayout>
    </ScrollView>
</ContentPage>
```

3.1.6 NewSurvey.xaml

```
<?xml version="1.0" encoding="UTF-8"?>
<ContentPage
   xmlns="http://xamarin.com/schemas/2014/forms"
   xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
   xmlns:prism="clr-namespace:Prism.Mvvm;assembly=Prism.Forms"
   x:Class="McurdyAssociates.NewSurvey"
   Title="Survey Defects">
   <StackLayout BackgroundColor="LemonChiffon">
     <StackLayout Padding="50,20,0,0">
         <Image Source="CompanyLogo"
                HorizontalOptions="Center"
                VerticalOptions="Start"
                HeightRequest="80"/>
      </StackLayout>
<!--Cracking-->
     <StackLayout Padding="50,10,0,0">
         <Label Text="Cracking"
           FontSize="Large"
           FontAttributes="Bold"
           TextColor="Black"
           HorizontalOptions="Start"
           VerticalOptions="Start"  />
      </StackLayout>
      <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="50,0,0,0" >
         <Button x:Name="btn2mm"
                HorizontalOptions="End"
                BackgroundColor="#DEFD00"
                CornerRadius="15"
                HeightRequest="50"
                WidthRequest="120"
                Text="FC &lt; 2mm"
                TextColor="Green"
                FontAttributes="Bold"
                Clicked="NavigateButton_OnClickedBtn2mm"
                FontSize="Small" />
         <Button x:Name="btnBC"
                HorizontalOptions="End"
                BackgroundColor="#DEFD00"
                HeightRequest="50"
                WidthRequest="120"
                Text="BC"
                TextColor="Green"
                Clicked="NavigateButton_OnClickedBtnBc"
                CornerRadius="15"
                FontAttributes="Bold"
```

```
            FontSize="Small" />
        <Button x:Name="btnCC"
            HorizontalOptions="End"
            BackgroundColor="#DEFD00"
            HeightRequest="50"
            WidthRequest="120"
            Text="CC"
            TextColor="Green"
            Clicked="NavigateButton_OnClickedBtnCc"
            CornerRadius="15"
            FontAttributes="Bold"
            FontSize="Small" />
        <Button x:Name="btnDC"
            HorizontalOptions="End"
            BackgroundColor="#DEFD00"
            HeightRequest="50"
            WidthRequest="120"
            Text="DC"
            TextColor="Green"
            Clicked="NavigateButton_OnClickedBtnDc"
            CornerRadius="15"
            FontAttributes="Bold"
            FontSize="Small" />

    </StackLayout>
    <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="50,0,0,0" >
        <Button x:Name="btnLs"
            HorizontalOptions="End"
            BackgroundColor="#DEFD00"
            HeightRequest="50"
            WidthRequest="120"
            Text="LS"
            TextColor="Green"
            Clicked="NavigateButton_OnClickedBtnLs"
            CornerRadius="15"
            FontAttributes="Bold"
            FontSize="Small" />
        <Button x:Name="btnSC"
            HorizontalOptions="End"
            BackgroundColor="#DEFD00"
            HeightRequest="50"
            WidthRequest="120"
            Text="SC"
            TextColor="Green"
            Clicked="NavigateButton_OnClickedBtnSc"
            CornerRadius="15"
```

```
                FontAttributes="Bold"
                FontSize="Small" />
           <Button x:Name="btnTC"
                HorizontalOptions="End"
                BackgroundColor="#DEFD00"
                HeightRequest="50"
                WidthRequest="120"
                Text="TC"
                TextColor="Green"
                Clicked="NavigateButton_OnClickedBtnTc"
                CornerRadius="15"
                FontAttributes="Bold"
                FontSize="Small" />

        </StackLayout>
<!--Corrugation-->
        <StackLayout Padding="50,10,0,0">
           <Label Text="Corrugation"
             FontSize="Large"
             FontAttributes="Bold"
             TextColor="Black"
             HorizontalOptions="Start"
             VerticalOptions="Start"  />
        </StackLayout>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="50,0,0,0" >
           <Button x:Name="btnCr"
                HorizontalOptions="End"
                BackgroundColor="#5055F9"
                HeightRequest="50"
                WidthRequest="120"
                Text="CR"
                TextColor="White"
                Clicked="NavigateButton_OnClickedBtnCr"
                CornerRadius="15"
                FontAttributes="Bold"
                FontSize="Small" />
        </StackLayout>
<!--Depressions-->
        <StackLayout Padding="50,10,0,0">
           <Label Text="Depressions"
             FontSize="Large"
             FontAttributes="Bold"
             TextColor="Black"
             HorizontalOptions="Start"
             VerticalOptions="Start"  />
        </StackLayout>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="50,0,0,0" >
           <Button x:Name="btnDw"
                HorizontalOptions="End"
                BackgroundColor="#FA03A0"
                HeightRequest="50"
                WidthRequest="120"
                Text="DW"
                TextColor="White"
                Clicked="NavigateButton_OnClickedBtnDw"
                CornerRadius="15"
                FontAttributes="Bold"
                FontSize="Small" />
           <Button x:Name="btnDp"
                HorizontalOptions="End"
                BackgroundColor="#FA03A0"
                HeightRequest="50"
                WidthRequest="120"
                Text="DP"
                TextColor="White"
                Clicked="NavigateButton_OnClickedBtnDp"
                CornerRadius="15"
                FontAttributes="Bold"
                FontSize="Small" />
        </StackLayout>
<!--Other-->
       <StackLayout Padding="50,10,0,0">
           <Label Text="Other"
             FontSize="Large"
             FontAttributes="Bold"
             TextColor="Black"
             HorizontalOptions="Start"
             VerticalOptions="Start"  />
        </StackLayout>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="50,0,0,0" >
           <Button x:Name="btnRu"
                HorizontalOptions="End"
                BackgroundColor="#119C19 "
                HeightRequest="50"
                WidthRequest="120"
                Text="RU"
                TextColor="White"
                Clicked="NavigateButton_OnClickedBtnRu"
                CornerRadius="15"
                FontAttributes="Bold"
                FontSize="Small" />
        <Button x:Name="btnSv"
                HorizontalOptions="End"
                BackgroundColor="#119C19 "
                HeightRequest="50"
                WidthRequest="120"
                Text="SV"
                TextColor="White"
                Clicked="NavigateButton_OnClickedBtnSv"
                CornerRadius="15"
                FontAttributes="Bold"
                FontSize="Small" />
        <Button x:Name="btnFl"
                HorizontalOptions="End"
                BackgroundColor="#119C19 "
                HeightRequest="50"
                WidthRequest="120"
                Text="FL"
                TextColor="White"
                Clicked="NavigateButton_OnClickedBtnFl"
                CornerRadius="15"
                FontAttributes="Bold"
                FontSize="Small" />
        <Button x:Name="btnPo"
                HorizontalOptions="End"
                BackgroundColor="#119C19 "
                HeightRequest="50"
                WidthRequest="120"
                Text="PO"
                TextColor="White"
                Clicked="NavigateButton_OnClickedBtnPo"
                CornerRadius="15"
                FontAttributes="Bold"
                FontSize="Small" />

    </StackLayout>
    <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="50,0,0,0" >
        <Button x:Name="btnRw"
                HorizontalOptions="End"
                BackgroundColor="#119C19 "
                HeightRequest="50"
                WidthRequest="120"
                Text="RW"
                TextColor="White"
                Clicked="NavigateButton_OnClickedBtnRw"
                CornerRadius="15"
                FontAttributes="Bold"
                FontSize="Small" />
            <Button x:Name="btnLs2"
                HorizontalOptions="End"
                BackgroundColor="#119C19 "
                HeightRequest="50"
                WidthRequest="120"
                Text="LS"
                TextColor="White"
                Clicked="NavigateButton_OnClickedBtnLst"
                CornerRadius="15"
                FontAttributes="Bold"
                FontSize="Small" />
            <Button x:Name="btnPc"
                HorizontalOptions="End"
                BackgroundColor="#119C19 "
                HeightRequest="50"
                WidthRequest="120"
                Text="PC"
                TextColor="White"
                Clicked="NavigateButton_OnClickedBtnPc"
                CornerRadius="15"
                FontAttributes="Bold"
                FontSize="Small"/>
            <Button x:Name="btnPh"
                HorizontalOptions="End"
                BackgroundColor="#119C19 "
                HeightRequest="50"
                WidthRequest="120"
                Text="PH"
                TextColor="White"
                Clicked="NavigateButton_OnClickedBtnPh"
                CornerRadius="15"
                FontAttributes="Bold"
                FontSize="Small" />

        </StackLayout>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="50,0,0,0" >
            <Button x:Name="btnHo"
                HorizontalOptions="End"
                BackgroundColor="#119C19 "
                HeightRequest="50"
                WidthRequest="120"
                Text="HO"
                TextColor="White"
                Clicked="NavigateButton_OnClickedBtnHo"
                CornerRadius="15"
                FontAttributes="Bold"
                FontSize="Small" />
      </StackLayout>
<!--Save-->
      <StackLayout Orientation="Horizontal" HorizontalOptions="CenterAndExpand"
VerticalOptions="Center" Padding="50,25,75,0" >
          <Button x:Name="btnSave"
                HorizontalOptions="End"
                BackgroundColor="#03A0FA"
                HeightRequest="60"
                CornerRadius="20"
                FontAttributes="Bold"
                WidthRequest="300"
                Text="Save And Exit"
                TextColor="White"
                Clicked="SaveAndExit_Clicked"
                FontSize="Medium" />
      </StackLayout>
   </StackLayout>

</ContentPage>
```

3.1.7 PreviousSurveys.xaml

```
<?xml version="1.0" encoding="UTF-8"?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             Title="Previous Surveys"
             x:Class="McurdyAssociates.Views.PreviousSurveys">
    <ScrollView>
    <StackLayout BackgroundColor="LemonChiffon" >

       <StackLayout Padding="50,20,0,0">
            <Image Source="CompanyLogo"
                   HorizontalOptions="Center"
                   VerticalOptions="Start"
                   HeightRequest="80"/>
        </StackLayout>

        <StackLayout Padding="0,20,0,0">
            <Grid x:Name="gridView">
                <Grid.RowDefinitions>
                    <RowDefinition Height="Auto" />
                </Grid.RowDefinitions>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="20*" />
                    <ColumnDefinition Width="20*" />
                    <ColumnDefinition Width="20*" />
                    <ColumnDefinition Width="20*" />
                    <ColumnDefinition Width="20*" />
                </Grid.ColumnDefinitions>
                <Label  Grid.Column="0" Grid.Row="0" FontSize="Title"
HorizontalOptions="Center" FontAttributes="Bold" Text="Survey ID"/>
                <Label  Grid.Column="1" Grid.Row="0" FontSize="Title"
HorizontalOptions="Center" FontAttributes="Bold" Text="Location"/>
                <Label Grid.Column="2" Grid.Row="0" FontSize="Title"
HorizontalOptions="Center" FontAttributes="Bold" Text="Date"/>
                <Label Grid.Column="3" Grid.Row="0" FontSize="Title"
HorizontalOptions="Center" FontAttributes="Bold" Text="Survey By"/>
            </Grid>


        <ListView x:Name="lstSurveys">
            <ListView.ItemTemplate >
                <DataTemplate>
                    <ViewCell>
                        <Grid >
                            <Grid.RowDefinitions>
                                <RowDefinition Height="Auto" />
                            </Grid.RowDefinitions>
                            <Grid.ColumnDefinitions>
                                <ColumnDefinition Width="20*" />
                                <ColumnDefinition Width="20*" />
                                <ColumnDefinition Width="20*" />
                                <ColumnDefinition Width="20*" />
                                <ColumnDefinition Width="20*" />
                            </Grid.ColumnDefinitions>

                            <Label  Grid.Column="0" HorizontalOptions="Center"
VerticalOptions="Center" Text="{Binding NewSurveyID}"/>
                            <Label  Grid.Column="1" HorizontalOptions="Center" Text="{Binding
SurveyLocation}"/>
                            <Label  Grid.Column="2" HorizontalOptions="Center" Text="{Binding
Date}"/>
                            <Label  Grid.Column="3" HorizontalOptions="Center" Text="{Binding
CompletedBy}"/>
                            <Button x:Name="btnResume"
                                    Grid.Column="4"
                                    CornerRadius="15"
                                    HorizontalOptions="Center"
                                    BackgroundColor="Green"
                                    HeightRequest="25"
                                    WidthRequest="75"
                                    Text="Resume"
                                    TextColor="White"
                                    Clicked="NavigateButton_OnClicked"
                                    FontSize="Micro" />
                        </Grid>



                    </ViewCell>
                </DataTemplate>
            </ListView.ItemTemplate>

        </ListView>
        </StackLayout>
    </StackLayout>
     </ScrollView>
</ContentPage>
```

3.1.8 SelectLocation.xaml

```
<?xml version="1.0" encoding="UTF-8"?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml" xmlns:maps="clr-
namespace:Xamarin.Forms.Maps;assembly=Xamarin.Forms.Maps"
             Title="Select Location"
             x:Class="McurdyAssociates.Views.SelectLocation">

    <StackLayout BackgroundColor="LemonChiffon" >

       <StackLayout Padding="50,20,0,50">
          <Image Source="CompanyLogo"
                 HorizontalOptions="Center"
                 VerticalOptions="Start"
                 HeightRequest="80"/>
        </StackLayout>

        <maps:Map MapType="Street" x:Name="map" HeightRequest="300"
WidthRequest="300"/>

        <StackLayout Padding="0,20,0,0">
            <Button Text="If this is not your location click here"
                    HorizontalOptions="CenterAndExpand"
                    CornerRadius="25"
                    WidthRequest="450"
                    BackgroundColor="Gold"
                    FontAttributes="Bold"
                    TextColor="Green"
                    Clicked="NavigateWrong_Clicked"
                    VerticalOptions="Start"/>
        </StackLayout>

        <StackLayout Orientation="Horizontal" HorizontalOptions="Center"
VerticalOptions="CenterAndExpand" >
            <Button x:Name="btnNew"
                HorizontalOptions="End"
                CornerRadius="25"
                BackgroundColor="#03A0FA"
                HeightRequest="75"
                WidthRequest="300"
                Text="Next"
                TextColor="White"
                FontAttributes="Bold"
                Clicked="NavigateButton_OnClicked"
                FontSize="Medium" />
        </StackLayout>
    </StackLayout>
```

</ContentPage>

3.1.9 SurveyDetails.xaml

```
<?xml version="1.0" encoding="UTF-8"?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="McurdyAssociates.Views.SurveyDetails"
             Title="Survey Details">
    <StackLayout BackgroundColor="LemonChiffon">

        <!-- Place new controls here -->
       <StackLayout Padding="50,20,0,0">
            <Image Source="CompanyLogo"
                   HorizontalOptions="Center"
                   VerticalOptions="Start"
                   HeightRequest="80"/>
        </StackLayout>

        <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="50,50,0,0">
            <Label Text="Survey Completed By"
                   TextColor="Black"
                   FontSize="Medium"
                   FontAttributes="Bold"
                   VerticalOptions="Start"/>
            <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="75,0,0,0">
                <Picker x:Name="completedBy"
                        FontAttributes="Bold"
                        HorizontalOptions="FillAndExpand"
                        WidthRequest="250" HeightRequest="35"
                        Title="Please Select Employee"
                        >
                </Picker>
            </StackLayout>
        </StackLayout>

        <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="50,50,0,0">
            <Label Text="Select Date"
                   TextColor="Black"
                   FontSize="Medium"
                   FontAttributes="Bold"
                   VerticalOptions="Start"/>
            <StackLayout HeightRequest="36" Orientation="Horizontal"
HorizontalOptions="Start" VerticalOptions="Start" Padding="150,0,0,0">
                <DatePicker
                    x:Name="startDatePicker"
                    FontAttributes="Bold"
                    Format="D"
                    WidthRequest="250"
                />
            </StackLayout>
        </StackLayout>

       <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="50,50,0,0">
            <Label Text="Weather"
                   TextColor="Black"
                   FontAttributes="Bold"
                   FontSize="Medium"
                   VerticalOptions="Start"
                   />
           <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="175,0,0,0">
                <Picker x:Name="Weather" HorizontalOptions="FillAndExpand"
FontAttributes="Bold" WidthRequest="250" HeightRequest="35" Title="Please Select
Weather">
                   <Picker.Items>
                       <x:String>Sunny</x:String>
                       <x:String>Raining</x:String>
                       <x:String>Drizzle</x:String>
                       <x:String>Overcast</x:String>
                       <x:String>Snow</x:String>
                   </Picker.Items>
                </Picker>
            </StackLayout>
       </StackLayout>

       <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="42,50,0,0">
            <Label Text=" Latitude"
                   TextColor="Black"
                   FontAttributes="Bold"
                   FontSize="Medium"
                   VerticalOptions="Start"/>
            <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="177,0,0,0">
            <Entry WidthRequest="250"
                   HeightRequest="35"
                   FontAttributes="Bold"
                   x:Name="latlab"/>
            </StackLayout>
        </StackLayout>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="42,50,0,0">
            <Label Text=" Longitude"
                   TextColor="Black"
                   FontAttributes="Bold"
                   FontSize="Medium"
                   VerticalOptions="Start"/>
            <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="160,0,0,0">
            <Entry WidthRequest="250"
                   HeightRequest="35"
                   FontAttributes="Bold"
                   x:Name="longlab"/>
            </StackLayout>
        </StackLayout>

        <StackLayout Orientation="Horizontal" HorizontalOptions="Center"
VerticalOptions="CenterAndExpand" >
            <Button x:Name="btnNew"
                 HorizontalOptions="End"

                 BackgroundColor="#03A0FA"
                 HeightRequest="75"
                 WidthRequest="300"
                 Text="Next"
                 TextColor="White"
                 CornerRadius="25"
                 FontAttributes="Bold"
                 Clicked="NavigateButton_OnClicked"
                 FontSize="Medium" />
        </StackLayout>
    </StackLayout>
</ContentPage>
```

3.1.10 SurveyInfo.xaml

```
<?xml version="1.0" encoding="UTF-8"?>
<ContentPage
   xmlns="http://xamarin.com/schemas/2014/forms"
   xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
   x:Class="McurdyAssociates.SurveyInfo"
   xmlns:prism="clr-namespace:Prism.Mvvm;assembly=Prism.Forms"
          Title="Defect Details">
   <ScrollView>
   <StackLayout BackgroundColor="LemonChiffon">

      <!-- Place new controls here -->
     <StackLayout Padding="0,20,0,0">
         <Image Source="CompanyLogo"
               HorizontalOptions="Center"
               VerticalOptions="Start"
               HeightRequest="80"/>
      </StackLayout>

      <StackLayout Orientation="Horizontal" HorizontalOptions="Center"
VerticalOptions="Start" Padding="0,20,0,0">
         <Label Text="{Binding Title}"
               x:Name="title"
               FontSize="Large"
               FontAttributes="Bold"
               TextColor="Black"
               VerticalOptions="Start"/>
      </StackLayout>
      <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="20,20,0,0">
         <Label Text="Chainage"
               FontSize="Medium"
               FontAttributes="Bold"
               TextColor="Black"
               VerticalOptions="Start"/>
         <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="150,0,0,0">
         <Entry WidthRequest="200"
               HeightRequest="35"
               Keyboard="Numeric"
               x:Name="ChainVal"/>
         </StackLayout>
      </StackLayout>
      <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="20,20,0,0">
         <Label Text=" Measurment Type"
               FontSize="Medium"
```

```
                FontAttributes="Bold"
                TextColor="Black"
                VerticalOptions="Start"/>
          <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="75,0,0,0">
          <Entry WidthRequest="200"
                HeightRequest="35"
                x:Name="measurmentEntry"
                IsReadOnly="True"/>
          </StackLayout>
       </StackLayout>
       <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="20,20,0,0">
          <Label Text=" Measurment Value"
                FontSize="Medium"
                FontAttributes="Bold"
                TextColor="Black"
                VerticalOptions="Start"/>
          <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="70,0,0,0">
          <Entry WidthRequest="200"
                HeightRequest="35"
                Keyboard="Numeric"
                x:Name="mesVal"/>
          </StackLayout>
          <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="20,0,0,0">

             <Button Text="Take a Photo"
                    x:Name="btnCamera"
                    CornerRadius="15"
                    BackgroundColor="DarkSalmon"
                    TextColor="White"
                    HeightRequest="40"
                    />
          </StackLayout>
       </StackLayout>
       <StackLayout Orientation="Horizontal" HorizontalOptions="Center"
VerticalOptions="Start">
          <ImageButton x:Name="PhotoImage"
                     HeightRequest="120"
                     WidthRequest="140"
                     Clicked="ExpandImage"
                     />
       </StackLayout>
       <StackLayout x:Name="recordLayout" Orientation="Horizontal"
HorizontalOptions="Center" VerticalOptions="Start">
```

```
                <StackLayout Orientation="Vertical">
              <Label Text="Recording..." FontSize = "21"/>
                <StackLayout Orientation = "Horizontal">
                   <Label x:Name="lblMinutes" Text="00" FontSize = "70"/>
                   <Label Text=":" FontSize = "70"/>
                   <Label x:Name="lblSeconds" Text="00" FontSize = "70"/>
                </StackLayout>
                 <StackLayout Orientation="Horizontal" Padding="0,0,0,20">
                   <Label Text="Detect silence:" FontSize = "21"/>
                   <Switch x:Name="IsSilence" IsToggled = "true" />
                </StackLayout>
            </StackLayout>
      <FlexLayout JustifyContent="SpaceAround">
         <Button x:Name="bntRecord"
                Text = "Record"
                BackgroundColor="#7cbb45"
                WidthRequest="100"
                CornerRadius="30"
                Clicked="Record_Clicked"/>
         <Button x:Name="bntStop"
                Text = "Stop"
                BackgroundColor="Silver"
                WidthRequest="100"
                Clicked="Stop_Clicked"
                CornerRadius="30"
                IsEnabled= "true"/>
         <Button x:Name="bntPlay"
                Text = "Play"
                BackgroundColor="Silver"
                WidthRequest="100"
                Clicked="Play_Clicked"
                CornerRadius="30"
                IsEnabled= "true"/>
       </FlexLayout>


         </StackLayout>
         <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="20,15,0,0">
             <Label Text=" Comments"
                    x:Name="comments"
                    FontAttributes="Bold"
                    TextColor="Black"
                    FontSize="Medium"
                    VerticalOptions="Start"/>
             <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="135,0,0,0">
```

```
            <Entry WidthRequest="200"
                   x:Name="comentry"
                   HeightRequest="35"/>
            </StackLayout>
            <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="20,0,0,0">
                <Button Text="Record"
                        x:Name="Record"
                        CornerRadius="15"
                        BackgroundColor="DarkSalmon"
                        TextColor="White"
                        HeightRequest="40"/>
                <Image x:Name="image"/>
            </StackLayout>
        </StackLayout>
<!--Save-->
        <StackLayout Orientation="Horizontal" HorizontalOptions="CenterAndExpand"
Padding="80,20,75,0" >
            <Button x:Name="btnSave"
                    HorizontalOptions="End"
                    BackgroundColor="#03A0FA"
                    HeightRequest="60"
                    WidthRequest="300"
                    CornerRadius="25"
                    FontAttributes="Bold"
                    Text="Save And Continue"
                    TextColor="White"
                    Clicked="SaveAndContinue_Clicked"
                    FontSize="Medium" />
        </StackLayout>

        <StackLayout x:Name="prevDefectLabel" Orientation="Horizontal"
HorizontalOptions="Center" VerticalOptions="Start" Padding="0,20,0,0">
            <Label Text="Previous Defects In This Survey"
                   FontSize="Large"
                   TextColor="Black"
                   FontAttributes="Bold"
                   VerticalOptions="Start"/>
        </StackLayout>

        <StackLayout x:Name="listOfDefects" Orientation="Horizontal"
HorizontalOptions="Center" VerticalOptions="Start" Padding="0,20,0,0">
            <Label Text="Defects will appear here as you enter them"
                   FontSize= "Medium"
                   FontAttributes="Italic"
                   VerticalOptions="Start"/>
        </StackLayout>
```

```
 <StackLayout x:Name="Gridlay" Padding="0,20,0,0">
           <Grid>
               <Grid.RowDefinitions>
                   <RowDefinition Height="Auto" />
               </Grid.RowDefinitions>
               <Grid.ColumnDefinitions>
                   <ColumnDefinition Width="20*" />
                   <ColumnDefinition Width="20*" />
                   <ColumnDefinition Width="20*" />
                   <ColumnDefinition Width="20*" />
                   <ColumnDefinition Width="20*" />
               </Grid.ColumnDefinitions>

               <Label  Grid.Column="1" Grid.Row="0" FontSize="Title" 
HorizontalOptions="Center" FontAttributes="Bold" Text="Survey Id"/>
               <Label Grid.Column="2" Grid.Row="0" FontSize="Title" 
HorizontalOptions="Center" FontAttributes="Bold" Text="Chainage"/>
               <Label Grid.Column="3" Grid.Row="0" FontSize="Title" 
HorizontalOptions="Center" FontAttributes="Bold" Text="Defect"/>
           </Grid>


       <ListView x:Name="lstDefects">
           <ListView.ItemTemplate>
               <DataTemplate>
                   <ViewCell>
                       <Grid>
                           <Grid.RowDefinitions>
                               <RowDefinition Height="Auto" />
                           </Grid.RowDefinitions>
                           <Grid.ColumnDefinitions>
                               <ColumnDefinition Width="20*" />
                               <ColumnDefinition Width="20*" />
                               <ColumnDefinition Width="20*" />
                               <ColumnDefinition Width="20*" />
                               <ColumnDefinition Width="20*" />
                           </Grid.ColumnDefinitions>

                           <Button x:Name="btnDelete"
                                   Grid.Column="0"
                                   HorizontalOptions="Center"
                                   BackgroundColor="Red"
                                   HeightRequest="35"
                                   CornerRadius="15"
                                   WidthRequest="75"
                                   Text="Delete"
```

```
                  TextColor="White"
                  Clicked="Delete_Clicked"
                  FontSize="Micro" />
              <Label  Grid.Column="1" HorizontalOptions="Center" Text="{Binding
SurveyId}"/>
              <Label  Grid.Column="2" HorizontalOptions="Center" Text="{Binding
Chainage}"/>
              <Label  Grid.Column="3" HorizontalOptions="Center" Text="{Binding
DefectName}"/>
              <Button x:Name="btnUpdate"
                  Grid.Column="4"
                  CornerRadius="15"
                  HorizontalOptions="Center"
                  BackgroundColor="Green"
                  HeightRequest="35"
                  WidthRequest="75"
                  Text="Update"
                  TextColor="White"
                  Clicked="Update_Clicked"
                  FontSize="Micro" />
            </Grid>



          </ViewCell>
        </DataTemplate>
      </ListView.ItemTemplate>

    </ListView>
    </StackLayout>
  </StackLayout>
  </ScrollView>
</ContentPage>
```

3.1.11 UpdateEmployee.xaml

```
<?xml version="1.0" encoding="UTF-8"?>
<ContentPage
   xmlns="http://xamarin.com/schemas/2014/forms"
   xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
   x:Class="McurdyAssociates.Views.UpdateEmployee"
   xmlns:prism="clr-namespace:Prism.Mvvm;assembly=Prism.Forms"
          Title="Update Employee">
   <ScrollView>
   <StackLayout BackgroundColor="LemonChiffon">

      <!-- Place new controls here -->
     <StackLayout Padding="0,20,0,60">
         <Image Source="CompanyLogo"
              HorizontalOptions="Center"
              VerticalOptions="Start"
              HeightRequest="80"/>
      </StackLayout>

      <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="20,20,0,0">
         <Label Text="Name"
              FontAttributes="Bold"
              TextColor="Black"
              FontSize="Medium"
              VerticalOptions="Start"/>
         <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="169,0,0,0">
         <Entry WidthRequest="200"
              HeightRequest="35"

              x:Name="NameVal"/>
         </StackLayout>
      </StackLayout>
      <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="20,20,0,0">
         <Label Text="Position"
              FontAttributes="Bold"
              TextColor="Black"
              FontSize="Medium"
              VerticalOptions="Start"/>
         <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="150,0,0,0">
         <Entry WidthRequest="200"
              HeightRequest="35"
              x:Name="PositionVal"
              />
```

```
            </StackLayout>
        </StackLayout>

<!--Save-->
        <StackLayout Orientation="Horizontal" HorizontalOptions="CenterAndExpand"
Padding="80,60,75,0" >
            <Button x:Name="btnUpdate"
                    HorizontalOptions="End"
                    BackgroundColor="#03A0FA"
                    HeightRequest="63"
                    WidthRequest="300"
                    CornerRadius="25"
                    FontAttributes="Bold"
                    Text="Update Employee"
                    TextColor="White"
                    Clicked="Update_Clicked"
                    FontSize="Medium" />
        </StackLayout>


    </StackLayout>
    </ScrollView>
</ContentPage>
```

3.1.12 UpdatePage.xaml

```
<?xml version="1.0" encoding="UTF-8"?>
<ContentPage
   xmlns="http://xamarin.com/schemas/2014/forms"
   xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
   xmlns:forms="clr-
namespace:FFImageLoading;assembly=FFImageLoading.Forms;assembly=FFImageLoadin
g.Forms"
   x:Class="McurdyAssociates.Views.UpdatePage"
   xmlns:prism="clr-namespace:Prism.Mvvm;assembly=Prism.Forms" xmlns:forms1="clr-
namespace:FFImageLoading.Forms;assembly=FFImageLoading.Forms"
             Title="Update Defect Details">

   <StackLayout BackgroundColor="LemonChiffon">

      <!-- Place new controls here -->
      <StackLayout Padding="0,20,0,0">
         <Image Source="CompanyLogo"
               HorizontalOptions="Center"
               VerticalOptions="Start"
               HeightRequest="80"/>
      </StackLayout>

      <StackLayout Orientation="Horizontal" HorizontalOptions="Center"
VerticalOptions="Start" Padding="0,20,0,0">
         <Label Text="{Binding Title}"
               x:Name="title"
               FontSize="Large"
               FontAttributes="Bold"
               TextColor="Black"
               VerticalOptions="Start"/>
      </StackLayout>
      <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="20,20,0,0">
         <Label Text="Chainage"
               FontSize="Medium"
               FontAttributes="Bold"
               TextColor="Black"
               VerticalOptions="Start"/>
         <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="150,0,0,0">
         <Entry WidthRequest="200"
               Keyboard="Numeric"
               HeightRequest="35"
               x:Name="ChainVal"/>
         </StackLayout>
      </StackLayout>
```

```
        <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="20,20,0,0">
            <Label Text=" Measurment Type"
                   FontSize="Medium"
                   FontAttributes="Bold"
                   TextColor="Black"
                   VerticalOptions="Start"/>
            <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="75,0,0,0">
            <Entry WidthRequest="200"
                   HeightRequest="35"
                   Keyboard="Numeric"
                   x:Name="measurmentEntry"
                   IsReadOnly="True"/>
            </StackLayout>
        </StackLayout>
        <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="20,20,0,0">
            <Label Text=" Measurment Value"
                   FontSize="Medium"
                   FontAttributes="Bold"
                   TextColor="Black"
                   VerticalOptions="Start"/>
            <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="70,0,0,0">
            <Entry WidthRequest="200"
                   HeightRequest="35"
                   Keyboard="Numeric"
                   x:Name="mesVal"/>
            </StackLayout>
            <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="20,0,0,0">
            <!--<Button x:Name="btnCamera"
                    ImageSource="CameraIcon"
                    WidthRequest="0"
                    HeightRequest="0"
                    Clicked="CameraButtonClicked"
                    />-->
            </StackLayout>
        </StackLayout>

        <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="20,15,0,0">
            <Label Text=" Comments"
                   x:Name="comments"
                   FontSize="Medium"
                   FontAttributes="Bold"
```

```
                TextColor="Black"
                VerticalOptions="Start"/>
          <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="135,0,0,0">
          <Entry WidthRequest="200"
                x:Name="comentry"
                HeightRequest="35"/>
          </StackLayout>
          <StackLayout Orientation="Horizontal" HorizontalOptions="Start"
VerticalOptions="Start" Padding="20,0,0,0">

              <Image x:Name="image"/>
          </StackLayout>
       </StackLayout>
              <StackLayout Orientation="Horizontal" HorizontalOptions="Center"
VerticalOptions="Start">
          <forms1:CachedImage x:Name="PhotoImages"
                          HeightRequest="120"
                          WidthRequest="140"
                          DownsampleToViewSize="True"
                          RetryCount="3"

                          />
       </StackLayout>
<!--Save-->
       <StackLayout Orientation="Horizontal" HorizontalOptions="CenterAndExpand"
Padding="80,20,75,0" >
          <Button x:Name="btnSave"
                 HorizontalOptions="End"
                 BackgroundColor="#03A0FA"
                 HeightRequest="60"
                 WidthRequest="300"
                 CornerRadius="25"
                 FontAttributes="Bold"
                 Text="Update And Continue"
                 TextColor="White"
                 Clicked="Update_Clicked"
                 FontSize="Medium" />
       </StackLayout>

   </StackLayout>

</ContentPage>
```

3.1.13 WrongLocation.xaml

```
<?xml version="1.0" encoding="UTF-8"?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:maps="clr-
namespace:Xamarin.Forms.Maps;assembly=Xamarin.Forms.Maps"
             xmlns:local="clr-namespace:McurdyAssociates;assembly=McurdyAssociates"
             Title="Select Location"
             x:Class="McurdyAssociates.Views.WrongLocation">

    <StackLayout >
      <StackLayout Padding="50,20,0,50">
          <Image Source="CompanyLogo"
                 HorizontalOptions="Center"
                 VerticalOptions="Start"
                 HeightRequest="80"/>
       </StackLayout>

    </StackLayout>
</ContentPage>
```

4.Cross-Platform Custom Renderer

4.1 Custom Renderer

4.1.1 CustomMapForClickEventRenderer

```
using System;
using System.Threading.Tasks;
using Android.Content;
using Android.Gms.Maps;
using Android.Gms.Maps.Model;
using McurdyAssociates.CustomRenderers;
using McurdyAssociates.Droid.CustomRenderers;
using Xamarin.Essentials;
using Xamarin.Forms;
using Xamarin.Forms.Maps;
using Xamarin.Forms.Maps.Android;
using Xamarin.Forms.Platform.Android;
using static Java.Util.ResourceBundle;

[assembly: ExportRenderer(typeof(CustomMapForClickEvent),
typeof(CustomMapForClickEventRenderer))]

namespace McurdyAssociates.Droid.CustomRenderers
{

    public class CustomMapForClickEventRenderer : MapRenderer, IOnMapReadyCallback
    {
        private GoogleMap _map;
        Marker marker;
        public double Long { get; set; }
        public double Lat { get; set; }
        double Latitud { get; set; }
        double Longitud { get; set; }

        public async void OnMapReady(GoogleMap googleMap)
        {

            await GetMyLocation();
            _map = googleMap;
            _map.MoveCamera(CameraUpdateFactory.NewLatLngZoom(new LatLng(Lat,
Long),
    14
    ));
            MarkerOptions markerOpt1 = new MarkerOptions();
            markerOpt1.SetPosition(new LatLng(Lat, Long));
            markerOpt1.SetTitle("Latitude: " + Lat + ",Longitude: " + Long);
            marker = googleMap.AddMarker(markerOpt1);
            if (_map != null)
            {
                //_map.GestureRecognizer.Add(new);
                _map.MapClick += googleMap_MapClick;
            }
        }

        protected override void
OnElementChanged(ElementChangedEventArgs<Xamarin.Forms.Maps.Map> e) //cambiar
a xamarin.forms.view
        {
            if (_map != null)
            {
                _map.MapClick -= googleMap_MapClick;
            }

            base.OnElementChanged(e);
            if (Control != null)
                ((MapView)Control).GetMapAsync(this);
        }

        private async Task GetMyLocation()
        {
            var location = await Geolocation.GetLocationAsync();
            if (location != null)
            {
                Lat = location.Latitude;
                Long = location.Longitude;

            }
        }



        private void googleMap_MapClick(object sender, GoogleMap.MapClickEventArgs e)
        {
            ((CustomMapForClickEvent)Element).OnTap(new Position(e.Point.Latitude,
e.Point.Longitude));

            using (var markerOption = new MarkerOptions())
            {
                _map.Clear();
                markerOption.SetPosition(e.Point);
                markerOption.SetTitle("Latitude: " + e.Point.Latitude + ", Longitude: " +
e.Point.Longitude);
                var marker = _map.AddMarker(markerOption);
            }
        }
    }
}
```