BOOSTER - ARCHERY PERFORMANCE TRACKING APPLICATION

TECHNICAL MANUAL

Supervisor: Dr. Joseph Kehoe
Author: Brendan Browne
Submission Date: 20 April 2019

Table of Contents

1. Introduction

The purpose of this Technical Manual document is to outline the requirements, installation procedure and show all relevant code for the BOOSTER application. Due to the lack of access to an Apple MAC for development and testing, the currently working version of the application is for Android only. The installable APK file of the BOOSTER application is available on the GitLab link listed below. To install the application at present, simply download the APK file and install via an install wizard, agreeing to all permissions requested.

The Xamarin code for this document is displayed as it appears in Visual studio to provide the most accurate representation on the project. Some code has been omitted from this document as it is either code gained from an external source, which has been referenced, or has no implementation of the page functionality as of yet. To see the full extent of the code please see the GitLab account linked below.

All project BOOSTER code can be found at the GitLab link below: https://gitlab.com/Bren85/sportswearable

2. Project Code

2.1 Shared project code

The files Models/CameraOptions.cs and Models/CameraPreview.cs were the result of a combination of code taken from multiple tutorials on implementing camera views in Xamarin [Pala17], [Docs18], [Dell16]. Some screens have been omitted ad they are linker screens and provide no additional functionality.

2.1.1 Data/FirebaseHelper.cs

```
using BOOSTER.Models;
using System.Collections.Generic;
using System.Threading.Tasks;
using Firebase.Database;
using Firebase.Database.Query;
using System.Linq;
using System;
using System.Diagnostics;
using System.Globalization;

namespace BOOSTER.Data
{
    class FirebaseHelper
    {

        public static FirebaseClient firebase = new FirebaseClient("https://booster-e7b42.firebaseio.com/");

        #region Users
        public static async Task<List<Users>> GetAllUsers()
        {
            try
            {
                var userlist = (await firebase
                .Child("Users")
                .OnceAsync<Users>()).Select(item =>
                new Users
                {
                    Username = item.Object.Username,
                    Password = item.Object.Password,
                    Salt = item.Object.Salt,
                }).ToList();
                return userlist;
            }
            catch (Exception e)
            {
                Debug.WriteLine($"Error:{e}");
                return null;
            }
        }

        public static async Task<bool> AddUser(string username, string password, string salt)
        {
            try
            {
                await firebase
                .Child("Users")
                .PostAsync(new Users() { Username = username, Password = password, Salt = salt });
                return true;
            }
            catch (Exception e)
            {
                Debug.WriteLine($"Error:{e}");
                return false;
            }
        }

        public static async Task<Users> GetUser(string username)
        {
            try
            {
                var allUsers = await GetAllUsers();
                //await firebase
```

```
            //.Child("Users")
            //.OnceAsync<Users>();
            return allUsers.Where(a => a.Username == username).FirstOrDefault();
        }
        catch (Exception e)
        {
            Debug.WriteLine($"Error:{e}");
            return null;
        }
    }
    #endregion

    #region Password
    public static async Task<bool> UpdatePassword(string username, string newPass, string salt)
    {
        try
        {
            var toUpdateUser = (await firebase
            .Child("Users")
            .OnceAsync<Users>()).Where(a => a.Object.Username == username).FirstOrDefault();
            await firebase
            .Child("Users")
            .Child(toUpdateUser.Key)
            .PutAsync(new Users() { Username = username, Password = SecurePasswordHasher.Hash(newPass, salt), Salt = salt });
            return true;
        }
        catch (Exception e)
        {
            Debug.WriteLine($"Error:{e}");
            return false;
        }
    }
    #endregion

    #region Scores
    public static async Task<bool> AddScore(string Username, List<int> end1, List<int> end2, List<int> end3, List<int> end4, List<in
t> end5,
        List<int> end6, List<int> end7, List<int> end8, List<int> end9, List<int> end10)
    {
        try
        {
            DateTime DT = DateTime.Now;
            string timeDate = DT.ToString("g", CultureInfo.CreateSpecificCulture("fr-BE"));
            await firebase
            .Child("Scores")
            .PostAsync(new Scores() { DTime = timeDate, Username = Username, End1 = end1, End2 = end2, End3 = end3, End4 = end4
, End5 = end5,
                End6 = end6, End7 = end7, End8 = end8, End9 = end9, End10 = end10 });
            return true;
        }
        catch (Exception e)
        {
            Debug.WriteLine($"Error:{e}");
            return false;
        }
    }

    public static async Task<List<Scores>> GetAllScores(string username)
    {
        try
        {
            var scorelist = (await firebase
            .Child("Scores")
            .OnceAsync<Scores>()).Select(item =>
            new Scores
            {
                DTime = item.Object.DTime,
                Username = item.Object.Username,
                End1 = item.Object.End1,
                End2 = item.Object.End2,
                End3 = item.Object.End3,
                End4 = item.Object.End4,
                End5 = item.Object.End5,
                End6 = item.Object.End6,
                End7 = item.Object.End7,
                End8 = item.Object.End8,
                End9 = item.Object.End9,
                End10 = item.Object.End10,
            }).ToList();
            //return scorelist;
```

```
            List<Scores> user = scorelist.Where(a => a.Username.Equals(username)).ToList();
            return user;
        }
        catch (Exception e)
        {
            Debug.WriteLine($"Error:{e}");
            return null;
        }
    }

    public static async Task<List<Scores>> GetAllScoresFromDate(string username, string date)
    {
        try
        {
            var scorelist = (await firebase
            .Child("Scores")
            .OnceAsync<Scores>()).Select(item =>
            new Scores
            {
                DTime = item.Object.DTime,
                Username = item.Object.Username,
                End1 = item.Object.End1,
                End2 = item.Object.End2,
                End3 = item.Object.End3,
                End4 = item.Object.End4,
                End5 = item.Object.End5,
                End6 = item.Object.End6,
                End7 = item.Object.End7,
                End8 = item.Object.End8,
                End9 = item.Object.End9,
                End10 = item.Object.End10,
            }).ToList();
            //return scorelist;

            return scorelist.Where(a => a.Username.Equals(username) && a.DTime.Equals(date)).ToList();
            //return score;
        }
        catch (Exception e)
        {
            Debug.WriteLine($"Error:{e}");
            return null;
        }
    }

    public static async Task<Scores> GetScore(string username, string sessionDate)
    {
        try
        {
            var allScores = await GetAllScores(username);
            await firebase
            .Child("Scores")
            .OnceAsync<Scores>();
            return allScores.Where(a => a.Username == username && a.DTime == sessionDate).FirstOrDefault();
        }
        catch (Exception e)
        {
            Debug.WriteLine($"Error:{e}");
            return null;
        }
    }
    #endregion
  }
}
```

2.1.2 Data/PickerOptions.cs

```
using System.Collections.ObjectModel;

namespace BOOSTER.Data
{
  class PickerOptions
  {
    private ObservableCollection<string> _options;

    public ObservableCollection<string> Options
    {
      get { return _options; }
      set { _options = value; }
    }
```

```
        public PickerOptions()
        {
            Options = new ObservableCollection<string>();

            Options.Add("Home");

            Options.Add("Change Password");

            Options.Add("Logout");
        }
    }
}
```

2.1.3 Data/SecurePasswordHasher.cs

```
using System.Security.Cryptography;
using System.Text;

namespace BOOSTER.Data
{
    public static class SecurePasswordHasher
    {
        private static int saltLengthLimit = 32;
        public static string GetSalt()
        {
            return GetSalt(saltLengthLimit);
        }
        private static string GetSalt(int maximumSaltLength)
        {
            var salt = new byte[maximumSaltLength];
            using (var random = new RNGCryptoServiceProvider())
            {
                random.GetNonZeroBytes(salt);
            }

            return Encoding.Default.GetString(salt);
        }

        public static string Hash(string password, string salt)
        {
            string saltedPassString = salt + password;
            byte[] saltedPassByte = Encoding.ASCII.GetBytes(saltedPassString);
            var md5 = new MD5CryptoServiceProvider();
            var md5data = md5.ComputeHash(saltedPassByte);
            string hashedPassword = Encoding.Default.GetString(md5data);

            return hashedPassword;
        }

        public static bool Verify(string password, string dbpassword, string salt)
        {
            string hashedPassword = Hash(password, salt);

            if (hashedPassword == dbpassword) return true;
            else return false;

        }
    }
}
```

2.1.4 Data/SQLHelper.cs

```
using SQLite;
using System.Collections.Generic;
using System.Linq;
using PCLStorage;
using BOOSTER.Models;
using System.Threading.Tasks;
using System;
using System.Diagnostics;
using System.Globalization;
using Newtonsoft.Json;

namespace BOOSTER.Data
{
    public class SQLHelper
```

```
{
    static object locker = new object();
    public static SQLiteConnection database = GetConnection();
    public SQLHelper()
    {
        database = GetConnection();
    }
    public static SQLiteConnection GetConnection()
    {
        SQLiteConnection sqlitConnection;
        var sqliteFilename = "BOOSTER.db3";
        string localPath = Environment.GetFolderPath(Environment.SpecialFolder.Personal);
        //IFolder folder = PCLStorage.FileSystem.Current.LocalStorage;
        string path = PortablePath.Combine(localPath, sqliteFilename);
        sqlitConnection = new SQLiteConnection(path);
        return sqlitConnection;
    }

    #region Users

    public static async Task<List<Users>> GetAllUsers()
    {
        lock (locker)
        {

            try
            {
                var userlist = database.Table<Users>().Select(item =>
                new Users
                {
                    Username = item.Username,
                    Password = item.Password,
                    Salt = item.Salt,
                }).ToList();
                return userlist;
            }
            catch (Exception e)
            {
                Debug.WriteLine($"Error:{e}");
                return null;
            }
        }
    }

    public static async Task<bool> AddUser(string username, string password, string salt)
    {
        lock (locker)
        {

            try
            {
                Users newUser = new Users
                {
                    Username = username,
                    Password = password,
                    Salt = salt
                };
                database.Insert(newUser);

                return true;
            }
            catch (Exception e)
            {
                Debug.WriteLine($"Error:{e}");
                return false;
            }
        }
    }

    public static async Task<Users> GetUser(string username)
    {
        lock (locker)
        {

            try
            {
                return database.Table<Users>().FirstOrDefault(x => x.Username == username);
            }
            catch (Exception e)
            {
                Debug.WriteLine($"Error:{e}");
```

```
            return null;
        }
    }
}

#endregion

#region Password

public static async Task<bool> UpdatePassword(string username, string newPass, string salt)
{
    database = GetConnection();
    try
    {
        Users thisUser = database.Table<Users>().Where(x => x.Username == username).FirstOrDefault();
        thisUser.Username = username;
        thisUser.Password = SecurePasswordHasher.Hash(newPass, salt);
        thisUser.Salt = salt;
        database.Update(thisUser);
        return true;
    }
    catch (Exception e)
    {
        Debug.WriteLine($"Error:{e}");
        return false;
    }
}
#endregion

#region Scores
public static async Task<bool> AddScore(string Username, string end1, string end2, string end3, string end4, string end5,
  string end6, string end7, string end8, string end9, string end10)
{
    lock (locker)
    {

        try
        {
            DateTime DT = DateTime.Now;
            string timeDate = DT.ToString("g", CultureInfo.CreateSpecificCulture("fr-BE"));

            SQLScores newScores = new SQLScores
            {
                DTime = timeDate,
                Username = Username,
                end1 = end1,
                end2 = end2,
                end3 = end3,
                end4 = end4,
                end5 = end5,
                end6 = end6,
                end7 = end7,
                end8 = end8,
                end9 = end9,
                end10 = end10
            };
            database.Insert(newScores);
            return true;
        }
        catch (Exception e)
        {
            Debug.WriteLine($"Error:{e}");
            return false;
        }
    }
}

public static async Task<List<SQLScores>> GetAllScores(string username)
{
    lock (locker)
    {

        try
        {
            var scorelist = database.Table<SQLScores>().Select(item =>
            new SQLScores
            {
                DTime = item.DTime,
                Username = item.Username,
                end1 = item.end1,
                end2 = item.end2,
```

```
                end3 = item.end3,
                end4 = item.end4,
                end5 = item.end5,
                end6 = item.end6,
                end7 = item.end7,
                end8 = item.end8,
                end9 = item.end9,
                end10 = item.end10,
            }).ToList();

            List<SQLScores> user = scorelist.Where(a => a.Username.Equals(username)).ToList();
            return user;
        }
        catch (Exception e)
        {
            Debug.WriteLine($"Error:{e}");
            return null;
        }
    }
}

public static async Task<List<SQLScores>> GetAllScoresFromDate(string username, string date)
{
    lock (locker)
    {

        try
        {
            var scorelist = database.Table<SQLScores>().Select(item =>
            new SQLScores
            {
                DTime = item.DTime,
                Username = item.Username,
                end1 = item.end1,
                end2 = item.end2,
                end3 = item.end3,
                end4 = item.end4,
                end5 = item.end5,
                end6 = item.end6,
                end7 = item.end7,
                end8 = item.end8,
                end9 = item.end9,
                end10 = item.end10,
            }).ToList();

            return scorelist.Where(a => a.Username.Equals(username) && a.DTime.Equals(date)).ToList();
        }
        catch (Exception e)
        {
            Debug.WriteLine($"Error:{e}");
            return null;
        }
    }
}

public static async Task<SQLScores> GetScore(string username, string sessionDate)
{
    lock (locker)
    {

        try
        {
            return database.Table<SQLScores>().FirstOrDefault(a => a.Username == username && a.DTime == sessionDate);
        }
        catch (Exception e)
        {
            Debug.WriteLine($"Error:{e}");
            return null;
        }
    }
}

#endregion
}
}
```

2.1.5 Interfaces/IBluetoothReader.cs

```
using System.Threading.Tasks;

namespace BOOSTER.Interfaces
{
   public interface IBluetoothReader
   {
      Task<int> ReadDeviceAsync();
   }
}
```

2.1.6 Models/FormDrawCS.cs

```
using Xamarin.Forms;

namespace BOOSTER
{
		public class FormDrawCS : ContentPage
   {
      public FormDrawCS()
      {
                                        Title = "FormHor";
                                        Padding = new Thickness(0, 20, 0, 0);
                                        Content = new StackLayout
                                        {
                                                Children = {
                                                        new CameraPreview {
                                                                Camera = CameraOptions.Rear,

                HorizontalOptions = LayoutOptions.FillAndExpand,
                                                                VerticalOptions = LayoutOptions.FillAndExpand
                                                        },
                                                        new Button{ Text="Set" }
                                                }
                                        };
                        }
                }
}
```

2.1.7 Models/OxyPlotInfo.cs

```
using System.Collections.Generic;

namespace BOOSTER.Models
{
   public class OxyPlotInfo
   {

      public string info1 { get; set; }
      public string info2 { get; set; }
      public string info3 { get; set; }

      public ICollection<OxyPlotItem> Items;

   }
}
```

2.1.8 Models/OxyPlotItem.cs

```
using OxyPlot;

namespace BOOSTER.Models
{
   public class OxyPlotItem
   {
      public string Label { get; set; }
      //public double Value { get; set; }

      public OxyColor Color { get; set; }
   }
```

```
}
```

2.1.9 Models/Scores.cs

```
using System.Collections.Generic;

namespace BOOSTER.Models
{
    public class Scores
    {
        public string DTime { get; set; }
        public string Username { get; set; }
        public List<int> End1 { get; set; }
        public List<int> End2 { get; set; }
        public List<int> End3 { get; set; }
        public List<int> End4 { get; set; }
        public List<int> End5 { get; set; }
        public List<int> End6 { get; set; }
        public List<int> End7 { get; set; }
        public List<int> End8 { get; set; }
        public List<int> End9 { get; set; }
        public List<int> End10 { get; set; }
    }
}
```

2.1.10 Models/SQLScores.cs

```
namespace BOOSTER.Models
{
    public class SQLScores
    {
        public string DTime { get; set; }
        public string Username { get; set; }
        public string end1 { get; set; }
        public string end2 { get; set; }
        public string end3 { get; set; }
        public string end4 { get; set; }
        public string end5 { get; set; }
        public string end6 { get; set; }
        public string end7 { get; set; }
        public string end8 { get; set; }
        public string end9 { get; set; }
        public string end10 { get; set; }

    }
}
```

2.1.11 Models/Users.cs

```
namespace BOOSTER.Models
{
    public class Users
    {
        public string Username { get; set; }
        public string Password { get; set; }
        public string Salt { get; set; }
    }
}
```

2.1.12 Views/Breathing.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage
    x:Class="BOOSTER.Breathing"
    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"
    xmlns:opt="clr-namespace:BOOSTER.Data"
    Title="Breathing"
    mc:Ignorable="d">
    <ContentPage.BindingContext>
```

```
        <opt:PickerOptions />
    </ContentPage.BindingContext>

    <ContentPage.Content>
        <Grid>
            <Grid.RowDefinitions>
                <RowDefinition Height=".2*" />
                <RowDefinition Height="*" />
            </Grid.RowDefinitions>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="*" />
            </Grid.ColumnDefinitions>

            <Grid
                x:Name="LogoGrid"
                Grid.Row="0"
                BackgroundColor="ForestGreen">
                <Grid.RowDefinitions>
                    <RowDefinition Height=".1*" />
                    <RowDefinition Height=".5*" />
                    <RowDefinition Height=".1*" />
                </Grid.RowDefinitions>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="*" />
                    <ColumnDefinition Width="*" />
                    <ColumnDefinition Width="*" />
                </Grid.ColumnDefinitions>
                <Picker
                    x:Name="picker"
                    ItemsSource="{Binding Options}"
                    SelectedIndexChanged="OnPickerSelectedIndexChanged" />
                <Image
                    Grid.Row="1"
                    Grid.Column="1"
                    Source="boosterimg" />

                <ImageButton
                    Grid.Row="1"
                    Grid.Column="2"
                    BackgroundColor="Transparent"
                    Clicked="OptionButton_Clicked"
                    HeightRequest="25"
                    HorizontalOptions="Center"
                    Source="threegears"
                    VerticalOptions="Center"
                    WidthRequest="25" />
            </Grid>


            <ScrollView
                Grid.Row="1"
                Margin="5,5,5,5"
                Orientation="Vertical"
                VerticalOptions="FillAndExpand">
                <StackLayout>
                    <Label
                        x:Name="BreathingLabel"
                        Margin="5,0,0,0"
                        FontSize="Title"
                        HorizontalTextAlignment="Center"
                        Text="Breathing Techniques" />
                    <BoxView
                        HeightRequest="5"
                        HorizontalOptions="FillAndExpand"
                        Color="#22A558" />
                    <Label
                        x:Name="BreathingBody1"
                        FontAttributes="Bold"
                        FontSize="Large"
                        Text="Adrenaline Dump Technique" />
                    <BoxView
                        HeightRequest="5"
                        HorizontalOptions="FillAndExpand"
                        Color="#22A558" />
                    <Label
                        x:Name="BreathingBody2"
                        FontSize="Large"
                        Text="Breath in through the mouth for a count of 6 seconds, Hold for a count of 2 seconds, Breath out forcefully through
the mouth for a count of 4 seconds" />
                    <BoxView
                        HeightRequest="5"
```

```
                    HorizontalOptions="FillAndExpand"
                    Color="#22A558" />
                <Label
                    x:Name="BreathingBody3"
                    FontAttributes="Bold"
                    FontSize="Large"
                    Text="Shot Cycle Breathing" />
                <BoxView
                    HeightRequest="5"
                    HorizontalOptions="FillAndExpand"
                    Color="#22A558" />
                <Label
                    x:Name="BreathingBody4"
                    FontSize="Large"
                    Text="Breath in 50% on set, release to 30% on setup, breath in to 65% during load and anchor, release 10% during transfe
r then hold at 55% until followthrough has completed" />
            </StackLayout>
        </ScrollView>

    </Grid>
  </ContentPage.Content>
</ContentPage>
```

2.1.13 Views/Breathing.xaml.cs

```
using BOOSTER.Views;
using System;
using Xamarin.Essentials;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace BOOSTER
{
  [XamlCompilation(XamlCompilationOptions.Compile)]
  public partial class Breathing : ContentPage
  {
    public string User { get; set; }
    public NetworkAccess CurrentNetworkStatus { get; set; }
    public Breathing(string UserName, NetworkAccess currentNetworkStatus)
    {
      InitializeComponent();
      User = UserName;
      CurrentNetworkStatus = currentNetworkStatus;
    }

    #region Picker
    void OptionButton_Clicked(object sender, System.EventArgs e)
    {
      picker.IsEnabled = true;
      picker.IsVisible = true;
      picker.Focus();
    }

    void OnPickerSelectedIndexChanged(object sender, EventArgs e)
    {
      var picker = (Picker)sender;
      int selectedIndex = picker.SelectedIndex;

      if (selectedIndex == 0)
      {
        Navigation.PushModalAsync(new Home(User, CurrentNetworkStatus));
      }
      else if (selectedIndex == 1)
      {
        Navigation.PushModalAsync(new ChangePassword(User, CurrentNetworkStatus));
      }
      else if (selectedIndex == 2)
      {
        Navigation.PushModalAsync(new LoginView(CurrentNetworkStatus));
      }
    }
    #endregion
  }
}
```

2.1.14 Views/ChangePassword.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage
    x:Class="BOOSTER.Views.ChangePassword"
    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"
    xmlns:opt="clr-namespace:BOOSTER.Data"
    Title="Change Password"
    mc:Ignorable="d">
    <ContentPage.BindingContext>
        <opt:PickerOptions />
    </ContentPage.BindingContext>

    <ContentPage.Content>
        <Grid>
            <Grid.RowDefinitions>
                <RowDefinition Height=".2*" />
                <RowDefinition Height="*" />
            </Grid.RowDefinitions>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="*" />
            </Grid.ColumnDefinitions>

            <Grid
                x:Name="LogoGrid"
                Grid.Row="0"
                BackgroundColor="ForestGreen">
                <Grid.RowDefinitions>
                    <RowDefinition Height=".1*" />
                    <RowDefinition Height=".5*" />
                    <RowDefinition Height=".1*" />
                </Grid.RowDefinitions>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="*" />
                    <ColumnDefinition Width="*" />
                    <ColumnDefinition Width="*" />
                </Grid.ColumnDefinitions>
                <Picker
                    x:Name="picker"
                    ItemsSource="{Binding Options}"
                    SelectedIndexChanged="OnPickerSelectedIndexChanged" />
                <Image
                    Grid.Row="1"
                    Grid.Column="1"
                    Source="boosterimg" />

                <ImageButton
                    Grid.Row="1"
                    Grid.Column="2"
                    BackgroundColor="Transparent"
                    Clicked="OptionButton_Clicked"
                    HeightRequest="25"
                    HorizontalOptions="Center"
                    Source="threegears"
                    VerticalOptions="Center"
                    WidthRequest="25" />
            </Grid>

            <Grid
                x:Name="BodyGrid"
                Grid.Row="1"
                Margin="16">

                <Grid.RowDefinitions>
                    <RowDefinition Height="Auto" />
                    <RowDefinition Height="*" />
                    <RowDefinition Height="Auto" />
                </Grid.RowDefinitions>
                <StackLayout Grid.Row="0">
                    <Label Text="Please enter a Username" />
                    <Entry
                        x:Name="Username"
                        Placeholder="Username"
                        ReturnType="Next" />
                    <Label Text="Please enter old Password" />
```

```
            <Entry
              x:Name="Old_Password"
              IsPassword="True"
              Placeholder="Password"
              ReturnType="Next" />
            <Label Text="Please enter new Password" />
            <Entry
              x:Name="New_Password1"
              IsPassword="True"
              Placeholder="Password"
              ReturnType="Next" />
            <Label Text="Please confirm new Password" />
            <Entry
              x:Name="New_Password2"
              IsPassword="True"
              Placeholder="Password"
              ReturnType="Done" />
          </StackLayout>
          <StackLayout Grid.Row="1">
            <Button
              x:Name="btn_ChgPass"
              Grid.Row="2"
              Clicked="ChangePass_OnClicked"
              Style="{StaticResource buttonStyle}"
              Text="Change Password" />
          </StackLayout>
        </Grid>
      </Grid>
    </ContentPage.Content>
</ContentPage>
```

2.1.15 Views/ChangePassword.xaml.cs

```
using BOOSTER.Data;
using BOOSTER.Models;
using System;
using Xamarin.Essentials;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace BOOSTER.Views
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class ChangePassword : ContentPage
    {
        public string User { get; set; }
        public NetworkAccess CurrentNetworkStatus { get; set; }
        public ChangePassword(string UserName, NetworkAccess currentNetworkStatus)
        {
            InitializeComponent();
            User = UserName;
            Username.Text = User;
            CurrentNetworkStatus = currentNetworkStatus;
        }

        #region Password change verification
        private async void ChangePass_OnClicked(object sender, EventArgs e)
        {
            Users user;

            if (Username.Text != "")
            {
                if (CurrentNetworkStatus == NetworkAccess.Internet)
                {
                    user = await FirebaseHelper.GetUser(Username.Text);
                }
                else
                {
                    user = await SQLHelper.GetUser(Username.Text);
                }
                if (user != null)
                {
                    if (Old_Password.Text != "" && New_Password1.Text != "" && New_Password2.Text != "")
                    {
                        if (SecurePasswordHasher.Verify(Old_Password.Text, user.Password, user.Salt))
                        {
                            if (New_Password1.Text == New_Password2.Text)
                            {
                                string newSalt = SecurePasswordHasher.GetSalt();
```

```
                    if (CurrentNetworkStatus == NetworkAccess.Internet)
                    {
                        await FirebaseHelper.UpdatePassword(user.Username, New_Password1.Text, newSalt);
                    }
                    await SQLHelper.UpdatePassword(user.Username, New_Password1.Text, newSalt);
                    await DisplayAlert("Success", "Password Updated Successfully", "OK");
                    await Navigation.PushModalAsync(new Home(User, CurrentNetworkStatus));
                }
                else
                {
                    await DisplayAlert("Failure", "Passwords dont match, update failed", "OK");
                    Username.Text = Username.Text;
                    resetPasswordFields();
                }
            }
            else
            {
                await DisplayAlert("Failure", "Password update failed", "OK");
                Username.Text = Username.Text;
                resetPasswordFields();
            }
        }
        else
        {
            await DisplayAlert("Password Change Failed", "Please enter credintails", "OK");
            Username.Text = "";
            resetPasswordFields();
        }
    }
    else
    {
        await DisplayAlert("Password Change Failed", "Please enter credintails", "OK");
        Username.Text = "";
        resetPasswordFields();
    }
}
else
{
    await DisplayAlert("Password Change Failed", "Please enter a username", "OK");
    Username.Text = "";
    resetPasswordFields();
}
//}
}

void resetPasswordFields()
{
    Old_Password.Text = "";
    New_Password1.Text = "";
    New_Password2.Text = "";
    Username.Focus();
}
#endregion

#region Picker
void OptionButton_Clicked(object sender, System.EventArgs e)
{
    picker.IsEnabled = true;
    picker.IsVisible = true;
    picker.Focus();
}

void OnPickerSelectedIndexChanged(object sender, EventArgs e)
{
    var picker = (Picker)sender;
    int selectedIndex = picker.SelectedIndex;

    if (selectedIndex == 0)
    {
        Navigation.PushModalAsync(new Home(User, CurrentNetworkStatus));

    }
    else if (selectedIndex == 1)
    {
        Navigation.PushModalAsync(new ChangePassword(User, CurrentNetworkStatus));
    }
    else if (selectedIndex == 2)
    {
        Navigation.PushModalAsync(new LoginView(CurrentNetworkStatus));
    }
}
```

```
        #endregion
    }
}
```

2.1.16 Views/FormDraw.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage
    x:Class="BOOSTER.Views.FormDraw"
    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:local="clr-namespace:BOOSTER;assembly=BOOSTER"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    Title="Form Draw"
    mc:Ignorable="d">
    <ContentPage.Content>
        <StackLayout Grid.Row="1">
            <local:CameraPreview
                Camera="Rear"
                HorizontalOptions="FillAndExpand"
                VerticalOptions="FillAndExpand" />
            <Button
                Margin="-2,-6,-2,-2"
                Clicked="FormDrawVert_OnClicked"
                Style="{StaticResource buttonStyle}"
                Text="Set" />
        </StackLayout>
    </ContentPage.Content>
</ContentPage>
```

2.1.17 Views/FormDraw.xaml.cs

```
using System;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace BOOSTER.Views
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class FormDraw : ContentPage
    {
        public FormDraw()
        {
            InitializeComponent();
        }

        #region Orientation
        protected override void OnAppearing()
        {
            base.OnAppearing();
            MessagingCenter.Send(this, "setLandscape");
        }
        protected override void OnDisappearing()
        {
            base.OnDisappearing();
            MessagingCenter.Send(this, "setPortrait");
        }

        #endregion

        #region onClickedEvents
        private void FormDrawVert_OnClicked(object sender, EventArgs e)
        {
            Navigation.PushModalAsync(new FormDrawVert());
            //Navigation.PushModalAsync(new TEST());
        }
        #endregion
    }
}
```

2.1.18 Views/FormDrawVert.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage
   x:Class="BOOSTER.FormDrawVert"
   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"
   xmlns:oxy="clr-namespace:OxyPlot.Xamarin.Forms;assembly=OxyPlot.Xamarin.Forms"
   Title="Vert Draw"
   mc:Ignorable="d">
   <ContentPage.Content>
      <Grid>
         <Grid.RowDefinitions>
            <RowDefinition Height=".12*" />
            <RowDefinition Height=".45*" />
            <RowDefinition Height=".45*" />
            <RowDefinition Height=".3*" />
         </Grid.RowDefinitions>
         <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*" />
         </Grid.ColumnDefinitions>

         <Grid
            x:Name="LogoGrid"
            Grid.Row="0"
            BackgroundColor="ForestGreen">
            <Grid.RowDefinitions>
               <RowDefinition Height=".1*" />
               <RowDefinition Height=".5*" />
               <RowDefinition Height=".1*" />
            </Grid.RowDefinitions>
            <Grid.ColumnDefinitions>
               <ColumnDefinition Width="*" />
               <ColumnDefinition Width="*" />
               <ColumnDefinition Width="*" />
            </Grid.ColumnDefinitions>
            <Image
               Grid.Row="1"
               Grid.Column="1"
               Source="boosterimg" />
         </Grid>

         <oxy:PlotView
            x:Name="plotmodel1"
            Grid.Row="1"
            Grid.Column="0"
            Margin="0,-2,0,-2"
            AbsoluteLayout.LayoutBounds="0,0,.5,.5"
            AbsoluteLayout.LayoutFlags="All" />

         <oxy:PlotView
            x:Name="plotmodel2"
            Grid.Row="2"
            Grid.Column="0"
            Margin="0,-2,0,-2"
            AbsoluteLayout.LayoutBounds="0,0,.5,.5"
            AbsoluteLayout.LayoutFlags="All" />

         <oxy:PlotView
            x:Name="plotmodel3"
            Grid.Row="3"
            Grid.Column="0"
            Margin="0,-2,0,-2"
            AbsoluteLayout.LayoutBounds="0,0,1,1"
            AbsoluteLayout.LayoutFlags="All" />
      </Grid>
   </ContentPage.Content>
</ContentPage>
```

2.1.19 Views/FormDrawVert.xaml.cs

```
using OxyPlot;
using OxyPlot.Series;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;
using OxyPlot.Annotations;
using OxyPlot.Axes;

namespace BOOSTER
{
   [XamlCompilation(XamlCompilationOptions.Compile)]
   public partial class FormDrawVert : ContentPage
   {
      public FormDrawVert()
      {
         InitializeComponent();
         this.Init();
      }

      public void Init()
      {
         this.Title = "OxyPlot";
         this.MakePlot();
      }

      #region Graph Plot Region
      public void MakePlot()
      {
         #region Create plotmodel objects
         PlotModel plotModel1 = new PlotModel
         {
            Title = "Draw Arm",
            TitleFontSize = 16,
            TextColor = OxyColor.FromRgb(122, 122, 122)
         };
         PlotModel plotModel2 = new PlotModel
         {
            Title = "Bow Arm",
            TitleFontSize = 16,
            TextColor = OxyColor.FromRgb(122, 122, 122)
         };
         PlotModel plotModel3 = new PlotModel
         {
            Title = "Heart Rate",
            TitleFontSize = 16,
            TextColor = OxyColor.FromRgb(122, 122, 122)
         };
         #endregion

         #region Plot points in a line series
         var ps1 = new LineSeries
         {
            MarkerType = MarkerType.None,
            StrokeThickness = .25,
            MarkerFill = OxyColors.Red,
            MarkerStroke = OxyColors.Red,
            TextColor = OxyColor.FromRgb(100,100,100),

         };

         ps1.Points.Add(new DataPoint(0, 120));
         ps1.Points.Add(new DataPoint(2, 125));
         ps1.Points.Add(new DataPoint(6, 145));
         ps1.Points.Add(new DataPoint(15, 130));
         ps1.Points.Add(new DataPoint(20, 140));

         var ps2 = new LineSeries
         {
            MarkerType = MarkerType.None,
            StrokeThickness = .25,
            MarkerFill = OxyColors.Blue,
            MarkerStroke = OxyColors.Blue,
            TextColor = OxyColor.FromRgb(100, 100, 100),
         };

         ps2.Points.Add(new DataPoint(0, 100));
         ps2.Points.Add(new DataPoint(2, 125));
```

```
ps2.Points.Add(new DataPoint(6, 145));
ps2.Points.Add(new DataPoint(15, 130));
ps2.Points.Add(new DataPoint(20, 140));

var ps3 = new LineSeries
{
   MarkerType = MarkerType.None,
   StrokeThickness = .25,
   MarkerFill = OxyColors.Yellow,
   MarkerStroke = OxyColors.Yellow,
   TextColor = OxyColor.FromRgb(100, 100, 100),
};

ps3.Points.Add(new DataPoint(0, 60));
ps3.Points.Add(new DataPoint(2, 65));
ps3.Points.Add(new DataPoint(4, 85));
ps3.Points.Add(new DataPoint(6, 70));
ps3.Points.Add(new DataPoint(8, 75));

#endregion

//foreach (var oxyitem in oxyPlotInfo1.Items)
//{
//    ps1.Slices.Add(new PieSlice(oxyitem.Label, oxyitem.Value) { IsExploded = false, Fill = oxyitem.Color });

//}

#region Add annotation lines
double DrawArmX1 = 0;
double DrawArmY1 = 137.5;
double DrawArmX2 = 0;
double DrawArmY2 = 127.5;
double BowArmX1 = 0;
double BowArmY1 = 130;
double BowArmX2 = 0;
double BowArmY2 = 115;

LineAnnotation DrawArmTopLine = new LineAnnotation()
{
   StrokeThickness = 1,
   Color = OxyColors.Green,
   Type = LineAnnotationType.Horizontal,
   X = DrawArmX1,
   Y = DrawArmY1
};
LineAnnotation DrawArmBottomLine = new LineAnnotation()
{
   StrokeThickness = 1,
   Color = OxyColors.Green,
   Type = LineAnnotationType.Horizontal,
   X = DrawArmX2,
   Y = DrawArmY2
};

LineAnnotation BowArmTopLine = new LineAnnotation()
{
   StrokeThickness = 1,
   Color = OxyColors.Green,
   Type = LineAnnotationType.Horizontal,
   X = BowArmX1,
   Y = BowArmY1
};
LineAnnotation BowArmBottomLine = new LineAnnotation()
{
   StrokeThickness = 1,
   Color = OxyColors.Green,
   Type = LineAnnotationType.Horizontal,
   X = BowArmX2,
   Y = BowArmY2
};

plotModel1.Annotations.Add(DrawArmTopLine);
plotModel1.Annotations.Add(DrawArmBottomLine);
plotModel2.Annotations.Add(BowArmTopLine);
plotModel2.Annotations.Add(BowArmBottomLine);
#endregion

#region Remove axis labels
plotModel1.Axes.Add(new LinearAxis()
{
   Position = AxisPosition.Left,
```

```
            IsAxisVisible = false
        });
        plotModel1.Axes.Add(new LinearAxis()
        {
            Position = AxisPosition.Bottom,
            IsAxisVisible = false
        });

        plotModel2.Axes.Add(new LinearAxis()
        {
            Position = AxisPosition.Left,
            IsAxisVisible = false
        });
        plotModel2.Axes.Add(new LinearAxis()
        {
            Position = AxisPosition.Bottom,
            IsAxisVisible = false
        });

        plotModel3.Axes.Add(new LinearAxis()
        {
            Position = AxisPosition.Bottom,
            IsAxisVisible = false
        });
        #endregion

        plotModel1.Series.Add(ps1);
        plotModel2.Series.Add(ps2);
        plotModel3.Series.Add(ps3);

        this.plotmodel1.Model = plotModel1;
        this.plotmodel2.Model = plotModel2;
        this.plotmodel3.Model = plotModel3;
    }
    #endregion
  }
}
```

2.1.20 Views/LoginView.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage
  x:Class="BOOSTER.LoginView"
  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"
  Title="Login"
  mc:Ignorable="d">
  <ContentPage.Content>
    <Grid>
      <Grid.RowDefinitions>
        <RowDefinition Height=".2*" />
        <RowDefinition Height="*" />
      </Grid.RowDefinitions>
      <Grid.ColumnDefinitions>
        <ColumnDefinition Width="*" />
      </Grid.ColumnDefinitions>

      <Grid
        x:Name="LogoGrid"
        Grid.Row="0"
        BackgroundColor="ForestGreen">
        <Grid.RowDefinitions>
          <RowDefinition Height=".1*" />
          <RowDefinition Height=".5*" />
          <RowDefinition Height=".1*" />
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
          <ColumnDefinition Width="*" />
          <ColumnDefinition Width="*" />
          <ColumnDefinition Width="*" />
        </Grid.ColumnDefinitions>
        <Image
          Grid.Row="1"
          Grid.Column="1"
          Source="boosterimg" />
      </Grid>
      <Grid
```

```
            x:Name="BodyGrid"
            Grid.Row="1"
            Margin="16">

            <Grid.RowDefinitions>
                <RowDefinition Height="Auto" />
                <RowDefinition Height="Auto" />
                <RowDefinition Height="Auto" />
            </Grid.RowDefinitions>
            <StackLayout Grid.Row="0">
                <Label Text="Please enter your credentials" />
                <Entry
                    x:Name="Entry_Username"
                    Placeholder="Username"
                    ReturnType="Next" />
                <Entry
                    x:Name="Entry_Password"
                    IsPassword="True"
                    Placeholder="Password"
                    ReturnType="Done" />
            </StackLayout>
            <StackLayout Grid.Row="1">
                <Button
                    x:Name="btn_Authenticate"
                    Grid.Row="2"
                    Clicked="Authenticate_OnClicked"
                    Style="{StaticResource buttonStyle}"
                    Text="Authenticate" />

            </StackLayout>
            <StackLayout Grid.Row="2">
                <Label Text="Register for an account" />
                <Button
                    x:Name="btn_SignUp"
                    Clicked="SignUp_OnClicked"
                    Style="{StaticResource buttonStyle}"
                    Text="Sign Up" />
            </StackLayout>
        </Grid>
    </Grid>
  </ContentPage.Content>
</ContentPage>
```

2.1.21 Views/LoginView.xaml.cs

```
using System;
using BOOSTER.Views;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;
using BOOSTER.Data;
using Xamarin.Essentials;
using BOOSTER.Models;

namespace BOOSTER
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class LoginView : ContentPage
    {
        public NetworkAccess CurrentNetworkStatus { get; set; }

        //SQLHelper DB = new SQLHelper();
        public LoginView(NetworkAccess currentNetworkStatus)
        {
            InitializeComponent();
            CurrentNetworkStatus = currentNetworkStatus;
        }

        #region Password Authentication
        private async void Authenticate_OnClicked(object sender, EventArgs e)
        {
            Users user;

            if (string.IsNullOrEmpty(Entry_Username.Text) || string.IsNullOrEmpty(Entry_Password.Text))
                await DisplayAlert("Empty Values", "Please enter Username and Password", "OK");
            else
            {
                if (CurrentNetworkStatus == NetworkAccess.Internet)
                {
                    user = await FirebaseHelper.GetUser(Entry_Username.Text);
```

```
            }
            else
            {
                user = await SQLHelper.GetUser(Entry_Username.Text);
            }

            if (user != null)
            {
                if (!SecurePasswordHasher.Verify(Entry_Password.Text, user.Password, user.Salt))
                {
                    await DisplayAlert("Login", "Login failed .. Please try again ", "OK");
                    Entry_Username.Text = "";
                    Entry_Password.Text = "";
                }
                else
                {
                    await DisplayAlert("Login", "Login Success ", "OK");
                    await Navigation.PushModalAsync(new Home(Entry_Username.Text, CurrentNetworkStatus));
                }
            }
            else
            {
                await DisplayAlert("Login", "Login failed .. Please try again ", "OK");
                Entry_Username.Text = "";
                Entry_Password.Text = "";
            }
        }
    }
    #endregion

    #region OnClicked Event
    private void SignUp_OnClicked(object sender, EventArgs e)
    {
        Navigation.PushModalAsync(new Registration(CurrentNetworkStatus));
    }

    protected override bool OnBackButtonPressed()
    {
        return true;
    }
    #endregion
  }
}
```

2.1.22 Views/Registration.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage
  x:Class="BOOSTER.Registration"
  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"
  Title="Registration"
  mc:Ignorable="d">
  <ContentPage.Content>
    <Grid>
      <Grid.RowDefinitions>
        <RowDefinition Height=".2*" />
        <RowDefinition Height="*" />
      </Grid.RowDefinitions>
      <Grid.ColumnDefinitions>
        <ColumnDefinition Width="*" />
      </Grid.ColumnDefinitions>

      <Grid
        x:Name="LogoGrid"
        Grid.Row="0"
        BackgroundColor="ForestGreen">
        <Grid.RowDefinitions>
          <RowDefinition Height=".1*" />
          <RowDefinition Height=".5*" />
          <RowDefinition Height=".1*" />
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
          <ColumnDefinition Width="*" />
          <ColumnDefinition Width="*" />
          <ColumnDefinition Width="*" />
        </Grid.ColumnDefinitions>
```

```
            <Image
                Grid.Row="1"
                Grid.Column="1"
                Source="boosterimg" />
        </Grid>

        <Grid
            x:Name="BodyGrid"
            Grid.Row="1"
            Margin="16">

            <Grid.RowDefinitions>
                <RowDefinition Height="Auto" />
                <RowDefinition Height="*" />
                <RowDefinition Height="Auto" />
            </Grid.RowDefinitions>
            <StackLayout Grid.Row="0">
                <Label Text="Please enter a Username" />
                <Entry
                    x:Name="Reg_Username"
                    Placeholder="Username"
                    ReturnType="Next" />
                <Label Text="Please enter a Password" />
                <Entry
                    x:Name="Reg_Password1"
                    IsPassword="True"
                    Placeholder="Password"
                    ReturnType="Next" />
                <Label Text="Please confirm Password" />
                <Entry
                    x:Name="Reg_Password2"
                    IsPassword="True"
                    Placeholder="Password"
                    ReturnType="Next" />
            </StackLayout>
            <StackLayout Grid.Row="1">
                <Button
                    x:Name="btn_Register"
                    Grid.Row="2"
                    Clicked="Register_OnClicked"
                    Style="{StaticResource buttonStyle}"
                    Text="Register" />

            </StackLayout>
            <StackLayout Grid.Row="2">
                <Label Text="Already have an account?" />
                <Button
                    x:Name="btn_Login"
                    Clicked="Login_OnClicked"
                    Style="{StaticResource buttonStyle}"
                    Text="Login here" />
            </StackLayout>
        </Grid>
    </Grid>
  </ContentPage.Content>
</ContentPage>
```

2.1.23 Views/Registration.xaml.cs

```
using BOOSTER.Data;
using BOOSTER.Models;
using BOOSTER.Views;
using System;
using System.Text.RegularExpressions;
using Xamarin.Essentials;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace BOOSTER
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class Registration : ContentPage
    {
        public NetworkAccess CurrentNetworkStatus { get; set; }
        public Registration(NetworkAccess currentNetworkStatus)
        {
            InitializeComponent();
            CurrentNetworkStatus = currentNetworkStatus;
        }
```

```
    #region Registration


    private async void Register_OnClicked(object sender, EventArgs e)
    {
        Users isuser;
        if (Reg_Username.Text != "")
        {
            if (CurrentNetworkStatus == NetworkAccess.Internet)
            {
                isuser = await FirebaseHelper.GetUser(Reg_Username.Text);
            }
            else
            {
                isuser = await SQLHelper.GetUser(Reg_Username.Text);
            }

            if (isuser == null || isuser.Username != Reg_Username.Text)
            {
                if (!string.IsNullOrEmpty(Reg_Password1.Text) && !string.IsNullOrEmpty(Reg_Password2.Text))
                {
                    if (isValidPassword(Reg_Password1.Text))
                    {
                        if (Reg_Password1.Text == Reg_Password2.Text)
                        {
                            string Salt = SecurePasswordHasher.GetSalt();
                            string hashedPass = SecurePasswordHasher.Hash(Reg_Password1.Text, Salt);

                            if (CurrentNetworkStatus == NetworkAccess.Internet)
                            {
                                await FirebaseHelper.AddUser(Reg_Username.Text, hashedPass, Salt);
                            }
                            await SQLHelper.AddUser(Reg_Username.Text, hashedPass, Salt);
                            Reg_Username.Text = string.Empty;
                            Reg_Password1.Text = string.Empty;
                            Reg_Password2.Text = string.Empty;
                            await DisplayAlert("Success", Reg_Username.Text + " Added Successfully", "OK");
                            await Navigation.PushModalAsync(new Home(Reg_Username.Text, CurrentNetworkStatus));
                        }
                        else
                        {
                            await DisplayAlert("Registration", "Registrtion Fail .. Passwords do not match ", "OK");
                            Reg_Username.Text = Reg_Username.Text;
                            Reg_Password1.Focus();
                        }
                    }
                    else
                    {
                        await DisplayAlert("Registration", "Registrtion Fail .. Password must contain 1 Lowercase, 1 Uppercase, 1 numerical a
nd one special character ", "OK");
                        Reg_Username.Text = Reg_Username.Text;
                        Reg_Password1.Focus();
                    }
                }
                else
                {
                    await DisplayAlert("Registration", "Registrtion Fail .. Passwords must be entered ", "OK");
                    Reg_Username.Text = Reg_Username.Text;
                    Reg_Password1.Focus();
                }
            }
            else
            {
                await DisplayAlert("Registration Failed", "Username already exist .. Please try differnt user name ", "OK");
                Reg_Username.Text = "";
                Reg_Username.Focus();
            }
        }
        else
        {
            await DisplayAlert("Registration Failed", "Enter a username ", "OK");
            Reg_Username.Text = "";
            Reg_Username.Focus();
        }
    }
    #endregion

    #region login clicked event
    private void Login_OnClicked(object sender, EventArgs e)
    {
```

```
            Navigation.PushModalAsync(new LoginView(CurrentNetworkStatus));
        }
        #endregion

        #region Check password is valid
        public static bool isValidPassword(string inputPassword)
        {
            string strRegex = @"(^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[^\da-zA-Z]).{8,15}$)";
            Regex re = new Regex(strRegex);

            if (re.IsMatch(inputPassword))
                return (true);
            else
                return (false);
        }
        #endregion
    }
}
```

2.1.24 Views/ScoreCard.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage
    x:Class="BOOSTER.ScoreCard"
    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">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height=".15*" />
            <RowDefinition Height="*" />
            <RowDefinition Height=".1*" />
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*" />
        </Grid.ColumnDefinitions>

        <Grid
            x:Name="LogoGrid"
            Grid.Row="0"
            BackgroundColor="ForestGreen">
            <Grid.RowDefinitions>
                <RowDefinition Height=".1*" />
                <RowDefinition Height=".5*" />
                <RowDefinition Height=".1*" />
            </Grid.RowDefinitions>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="*" />
                <ColumnDefinition Width="*" />
                <ColumnDefinition Width="*" />
            </Grid.ColumnDefinitions>
            <Image
                Grid.Row="1"
                Grid.Column="1"
                Source="boosterimg" />
        </Grid>

        <Grid
            x:Name="BodyGrid"
            Grid.Row="1"
            Margin="0,0,0,-4"
            BackgroundColor="Black"
            VerticalOptions="StartAndExpand">

            <Grid.RowDefinitions>
                <RowDefinition Height="*" />
                <RowDefinition Height="*" />
                <RowDefinition Height="*" />
                <RowDefinition Height="*" />
                <RowDefinition Height="*" />
                <RowDefinition Height="*" />
                <RowDefinition Height="*" />
                <RowDefinition Height="*" />
                <RowDefinition Height="*" />
                <RowDefinition Height="*" />
                <RowDefinition Height="*" />
                <RowDefinition Height="*" />
```

```
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*" />
            <ColumnDefinition Width="*" />
            <ColumnDefinition Width="*" />
            <ColumnDefinition Width="*" />
            <ColumnDefinition Width="*" />
            <ColumnDefinition Width="*" />
        </Grid.ColumnDefinitions>

        <Label
            Grid.Row="0"
            Grid.ColumnSpan="6"
            Margin="0,0,0,-2"
            BackgroundColor="White"
            FontAttributes="Bold"
            FontSize="Body"
            HorizontalTextAlignment="Center"
            VerticalTextAlignment="Center">
            Scores
        </Label>
        <Label
            Grid.Row="1"
            Grid.Column="0"
            Margin="0,-2,-2,-2"
            BackgroundColor="White"
            HorizontalTextAlignment="Center"
            VerticalTextAlignment="Center">
            End
        </Label>
        <Label
            Grid.Row="1"
            Grid.Column="1"
            Margin="-2,-2,-2,-2"
            BackgroundColor="White"
            HorizontalTextAlignment="Center"
            VerticalTextAlignment="Center">
            Arrow 1
        </Label>
        <Label
            Grid.Row="1"
            Grid.Column="2"
            Margin="-2,-2,-2,-2"
            BackgroundColor="White"
            HorizontalTextAlignment="Center"
            VerticalTextAlignment="Center">
            Arrow 2
        </Label>
        <Label
            Grid.Row="1"
            Grid.Column="3"
            Margin="-2,-2,-2,-2"
            BackgroundColor="White"
            HorizontalTextAlignment="Center"
            VerticalTextAlignment="Center">
            Arrow 3
        </Label>
        <Label
            Grid.Row="1"
            Grid.Column="4"
            Margin="-2,-2,-2,-2"
            BackgroundColor="White"
            HorizontalTextAlignment="Center"
            VerticalTextAlignment="Center">
            Subtotal
        </Label>
        <Label
            Grid.Row="1"
            Grid.Column="5"
            Margin="-2,-2,0,-2"
            BackgroundColor="White"
            HorizontalTextAlignment="Center"
            VerticalTextAlignment="Center">
            Total
        </Label>
        <Label
            Grid.Row="2"
            Grid.Column="0"
            Margin="0,-2,-2,-2"
            BackgroundColor="White"
            HorizontalTextAlignment="Center"
```

```
            VerticalTextAlignment="Center">
            1
        </Label>
        <Entry
            x:Name="end1arrow1"
            Grid.Row="2"
            Grid.Column="1"
            Margin="-2,-2,-2,-2"
            BackgroundColor="White"
            FontSize="Small"
            Keyboard="Numeric"
            MaxLength="2"
            ReturnType="Next"
            TextChanged="End1Entry_TextChanged" />
        <Entry
            x:Name="end1arrow2"
            Grid.Row="2"
            Grid.Column="2"
            Margin="-2,-2,-2,-2"
            BackgroundColor="White"
            FontSize="Small"
            Keyboard="Numeric"
            MaxLength="2"
            ReturnType="Next"
            TextChanged="End1Entry_TextChanged" />
        <Entry
            x:Name="end1arrow3"
            Grid.Row="2"
            Grid.Column="3"
            Margin="-2,-2,-2,-2"
            BackgroundColor="White"
            FontSize="Small"
            Keyboard="Numeric"
            MaxLength="2"
            ReturnType="Next"
            TextChanged="End1Entry_TextChanged" />
        <Entry
            x:Name="end1subtotal"
            Grid.Row="2"
            Grid.Column="4"
            Margin="-2,-2,-2,-2"
            BackgroundColor="White"
            FontSize="Small"
            IsReadOnly="True" />
        <Grid
            Grid.Row="2"
            Grid.Column="5"
            Margin="-2,-2,0,-2" />
        <Label
            Grid.Row="3"
            Grid.Column="0"
            Margin="0,-2,-2,-2"
            BackgroundColor="White"
            HorizontalTextAlignment="Center"
            VerticalTextAlignment="Center">
            2
        </Label>
        <Entry
            x:Name="end2arrow1"
            Grid.Row="3"
            Grid.Column="1"
            Margin="-2,-2,-2,-2"
            BackgroundColor="White"
            FontSize="Small"
            Keyboard="Numeric"
            MaxLength="2"
            ReturnType="Next"
            TextChanged="End2Entry_TextChanged" />
        <Entry
            x:Name="end2arrow2"
            Grid.Row="3"
            Grid.Column="2"
            Margin="-2,-2,-2,-2"
            BackgroundColor="White"
            FontSize="Small"
            Keyboard="Numeric"
            MaxLength="2"
            ReturnType="Next"
            TextChanged="End2Entry_TextChanged" />
        <Entry
            x:Name="end2arrow3"
```

```
            Grid.Row="3"
            Grid.Column="3"
            Margin="-2,-2,-2,-2"
            BackgroundColor="White"
            FontSize="Small"
            Keyboard="Numeric"
            MaxLength="2"
            ReturnType="Next"
            TextChanged="End2Entry_TextChanged" />
        <Entry
            x:Name="end2subtotal"
            Grid.Row="3"
            Grid.Column="4"
            Margin="-2,-2,-2,-2"
            BackgroundColor="White"
            FontSize="Small"
            IsReadOnly="True" />
        <Entry
            x:Name="end2total"
            Grid.Row="3"
            Grid.Column="5"
            Margin="-2,-2,-2,-2"
            BackgroundColor="White"
            FontSize="Small"
            IsReadOnly="True" />
.
.
.
.
.

        <Label
            Grid.Row="11"
            Grid.Column="0"
            Margin="0,-2,-2,0"
            BackgroundColor="White"
            HorizontalTextAlignment="Center"
            VerticalTextAlignment="Center">
            10
        </Label>
        <Entry
            x:Name="end10arrow1"
            Grid.Row="11"
            Grid.Column="1"
            Margin="-2,-2,-2,-2"
            BackgroundColor="White"
            FontSize="Small"
            Keyboard="Numeric"
            MaxLength="2"
            ReturnType="Next"
            TextChanged="End10Entry_TextChanged" />
        <Entry
            x:Name="end10arrow2"
            Grid.Row="11"
            Grid.Column="2"
            Margin="-2,-2,-2,-2"
            BackgroundColor="White"
            FontSize="Small"
            Keyboard="Numeric"
            MaxLength="2"
            ReturnType="Next"
            TextChanged="End10Entry_TextChanged" />
        <Entry
            x:Name="end10arrow3"
            Grid.Row="11"
            Grid.Column="3"
            Margin="-2,-2,-2,-2"
            BackgroundColor="White"
            FontSize="Small"
            Keyboard="Numeric"
            MaxLength="2"
            ReturnType="Done"
            TextChanged="End10Entry_TextChanged" />
        <Entry
            x:Name="end10subtotal"
            Grid.Row="11"
            Grid.Column="4"
            Margin="-2,-2,-2,-2"
            BackgroundColor="White"
            FontSize="Small"
            IsReadOnly="True" />
```

```
        <Entry
            x:Name="end10total"
            Grid.Row="11"
            Grid.Column="5"
            Margin="-2,-2,-2,-2"
            BackgroundColor="White"
            FontSize="Small"
            IsReadOnly="True" />
    </Grid>
    <Grid
        Grid.Row="2"
        Margin="-2,-2,-2,0"
        BackgroundColor="Black"
        VerticalOptions="StartAndExpand">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*" />
            <ColumnDefinition Width="*" />
        </Grid.ColumnDefinitions>
        <Button
            Grid.Column="0"
            Clicked="SaveButton_OnClicked"
            Style="{StaticResource buttonStyle}"
            Text="Save" />
        <Button
            Grid.Column="1"
            Clicked="ClearButton_OnClicked"
            Style="{StaticResource buttonStyle}"
            Text="Clear" />
    </Grid>
  </Grid>
</ContentPage>
```

2.1.25 Views/ScoreCard.xaml.cs

```
using BOOSTER.Data;
using System;
using System.Collections.Generic;
using Xamarin.Essentials;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace BOOSTER
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class ScoreCard : ContentPage
    {
        #region variables
        List<int> End1 = new List<int>();
        List<int> End2 = new List<int>();
        List<int> End3 = new List<int>();
        List<int> End4 = new List<int>();
        List<int> End5 = new List<int>();
        List<int> End6 = new List<int>();
        List<int> End7 = new List<int>();
        List<int> End8 = new List<int>();
        List<int> End9 = new List<int>();
        List<int> End10 = new List<int>();
        public string User { get; set; }
        public NetworkAccess CurrentNetworkStatus { get; set; }

        public int var1, var2, var3, var4, var5, var6, var7, var8, var9, var10,
                   var11, var12, var13, var14, var15, var16, var17, var18, var19, var20,
                   var21, var22, var23, var24, var25, var26, var27, var28, var29, var30;
        public int sub1, sub2, sub3, sub4, sub5, sub6, sub7, sub8, sub9, sub10 = 0;
        public int total2, total3, total4, total5, total6, total7, total8, total9, total10 = 0;

        #endregion
        public ScoreCard(string Username, NetworkAccess currentNetworkStatus)
        {
            InitializeComponent();
            User = Username;
            CurrentNetworkStatus = currentNetworkStatus;
        }

        #region Check value is 0 > n > 10
        public static bool IsValid(string str)
        {
            int i;
            return int.TryParse(str, out i) && i >= 0 && i <= 10;
```

```
    }
    #endregion

    #region Entry Fields Event Changed Functions
    void End1Entry_TextChanged(object sender, TextChangedEventArgs e)
    {
        if (end1arrow1.Text == null) { }
        else
        {
            if (IsValid(end1arrow1.Text))
            {
                var1 = int.Parse(end1arrow1.Text);
            }
            else
            {
                end1arrow1.Text = string.Empty;
            }
        }
        if (end1arrow2.Text == null) { }
        else
        {
            if (IsValid(end1arrow2.Text))
            {
                var2 = int.Parse(end1arrow2.Text);
            }
            else
            {
                end1arrow2.Text = string.Empty;
            }
        }
        if (end1arrow3.Text == null) { }
        else
        {
            if (IsValid(end1arrow3.Text))
            {
                var3 = int.Parse(end1arrow3.Text);
            }
            else
            {
                end1arrow3.Text = string.Empty;
            }
        }

        sub1 = var1 + var2 + var3;
        string substring = sub1.ToString();
        end1subtotal.Text = substring;

    }

    void End2Entry_TextChanged(object sender, TextChangedEventArgs e)
    {
        if (end2arrow1.Text == null) { }
        else
        {
            if (IsValid(end2arrow1.Text))
            {
                var4 = int.Parse(end2arrow1.Text);
            }
            else
            {
                end2arrow1.Text = string.Empty;
            }
        }
        if (end2arrow2.Text == null) { }
        else
        {
            if (IsValid(end2arrow2.Text))
            {
                var5 = int.Parse(end2arrow2.Text);
            }
            else
            {
                end2arrow2.Text = string.Empty;
            }
        }
        if (end2arrow3.Text == null) { }
        else
        {
            if (IsValid(end2arrow3.Text))
            {
                var6 = int.Parse(end2arrow3.Text);
```

```
            }
            else
            {
                end2arrow3.Text = string.Empty;
            }
        }

        sub2 = var4 + var5 + var6;
        string substring = sub2.ToString();
        end2subtotal.Text = substring;

        total2 = sub2 + (int.Parse(end1subtotal.Text));

        string totalString = total2.ToString();
        end2total.Text = totalString;

    }

.
.
.
.
.

    void End10Entry_TextChanged(object sender, TextChangedEventArgs e)
    {
        if (end10arrow1.Text == null) { }
        else
        {
            if (IsValid(end10arrow1.Text))
            {
                var28 = int.Parse(end10arrow1.Text);
            }
            else
            {
                end10arrow1.Text = string.Empty;
            }
        }
        if (end10arrow2.Text == null) { }
        else
        {
            if (IsValid(end10arrow2.Text))
            {
                var29 = int.Parse(end10arrow2.Text);
            }
            else
            {
                end10arrow2.Text = string.Empty;
            }
        }
        if (end10arrow3.Text == null) { }
        else
        {
            if (IsValid(end10arrow3.Text))
            {
                var30 = int.Parse(end2arrow3.Text);
            }
            else
            {
                end10arrow3.Text = string.Empty;
            }
        }

        sub10 = var28 + var29 + var30;
        string substring = sub10.ToString();
        end10subtotal.Text = substring;

        total10 = sub10 + (int.Parse(end9total.Text));

        string totalString = total10.ToString();
        end10total.Text = totalString;
    }
    #endregion

    #region Save Clicked
    private async void SaveButton_OnClicked(object sender, EventArgs e)
    {
        End1.Add(var1);
        End1.Add(var2);
        End1.Add(var3);
        End1.Add(sub1);
```

```
            End2.Add(var4);
            End2.Add(var5);
            End2.Add(var6);
            End2.Add(sub2);
            End2.Add(total2);


.
.
.
.
.


            End10.Add(var28);
            End10.Add(var29);
            End10.Add(var30);
            End10.Add(sub10);
            End10.Add(total10);

            if (CurrentNetworkStatus == NetworkAccess.Internet)
            {
                await FirebaseHelper.AddScore(User, End1, End2, End3, End4, End5, End6, End7, End8, End9, End10);
            }
            else
            {

                string end1 = var1 + "," + var2 + "," + var3 + "," + sub1;
                string end2 = var4 + "," + var5 + "," + var6 + "," + sub2 + "," + total2;
                string end3 = var7 + "," + var8 + "," + var9 + "," + sub3 + "," + total3;
                string end4 = var10 + "," + var11 + "," + var12 + "," + sub4 + "," + total4;
                string end5 = var13 + "," + var14 + "," + var15 + "," + sub5 + "," + total5;
                string end6 = var16 + "," + var17 + "," + var18 + "," + sub6 + "," + total6;
                string end7 = var19 + "," + var20 + "," + var21 + "," + sub7 + "," + total7;
                string end8 = var22 + "," + var23 + "," + var24 + "," + sub8 + "," + total8;
                string end9 = var25 + "," + var26 + "," + var27 + "," + sub9 + "," + total9;
                string end10 = var28 + "," + var29 + "," + var30 + "," + sub10 + "," + total10;
                await SQLHelper.AddScore(User, end1, end2, end3, end4, end5, end6, end7, end8, end9, end10);
            }
            ClearAll();
            await DisplayAlert("Success", "Scores Saved Successfully", "OK");
        }
        #endregion

        #region Clear Button Clicked
        private void ClearButton_OnClicked(object sender, EventArgs e)
        {
            ClearAll();
        }

        void ClearAll()
        {
            end1arrow1.Text = end1arrow2.Text = end1arrow3.Text = string.Empty;
            end2arrow1.Text = end2arrow2.Text = end2arrow3.Text = string.Empty;
            end3arrow1.Text = end3arrow2.Text = end3arrow3.Text = string.Empty;
            end4arrow1.Text = end4arrow2.Text = end4arrow3.Text = string.Empty;
            end5arrow1.Text = end5arrow2.Text = end5arrow3.Text = string.Empty;
            end6arrow1.Text = end6arrow2.Text = end6arrow3.Text = string.Empty;
            end7arrow1.Text = end7arrow2.Text = end7arrow3.Text = string.Empty;
            end8arrow1.Text = end8arrow2.Text = end8arrow3.Text = string.Empty;
            end9arrow1.Text = end9arrow2.Text = end9arrow3.Text = string.Empty;
            end10arrow1.Text = end10arrow2.Text = end10arrow3.Text = string.Empty;

            end1subtotal.Text = end2subtotal.Text = end2total.Text = end3subtotal.Text = end3total.Text = end4subtotal.Text = end4total.Te
xt =
            end5subtotal.Text = end5total.Text = end6subtotal.Text = end6total.Text = end7subtotal.Text = end7total.Text =
            end8subtotal.Text = end8total.Text = end9subtotal.Text = end9total.Text = end10subtotal.Text = end10total.Text = "0";

            var1 = var2 = var3 = var4 = var5 = var6 = var7 = var8 = var9 = var10 =
            var11 = var12 = var13 = var14 = var15 = var16 = var17 = var18 = var19 = var20 =
            var21 = var22 = var23 = var24 = var25 = var26 = var27 = var28 = var29 = var30 = 0;
            sub1 = sub2 = sub3 = sub4 = sub5 = sub6 = sub7 = sub8 = sub9 = sub10 = 0;
            total2 = total3 = total4 = total5 = total6 = total7 = total8 = total9 = total10 = 0;
        }
        #endregion
    }
}
```

2.1.26 Views/ScoreView.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage
   x:Class="BOOSTER.ScoreView"
   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"
   Title="View Scores"
   mc:Ignorable="d">

   <Grid>
      <Grid.RowDefinitions>
         <RowDefinition Height=".15*" />
         <RowDefinition Height=".1*" />
         <RowDefinition Height="*" />
      </Grid.RowDefinitions>
      <Grid.ColumnDefinitions>
         <ColumnDefinition Width="*" />
      </Grid.ColumnDefinitions>

      <Grid
          x:Name="LogoGrid"
         Grid.Row="0"
         BackgroundColor="ForestGreen">
         <Grid.RowDefinitions>
            <RowDefinition Height=".1*" />
            <RowDefinition Height=".5*" />
            <RowDefinition Height=".1*" />
         </Grid.RowDefinitions>
         <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*" />
            <ColumnDefinition Width="*" />
            <ColumnDefinition Width="*" />
         </Grid.ColumnDefinitions>

         <Image
            Grid.Row="1"
            Grid.Column="1"
            Source="boosterimg" />
      </Grid>

      <Grid
         Grid.Row="1"
         BackgroundColor="Black"
         VerticalOptions="FillAndExpand">

         <Picker x:Name="picker1" SelectedIndexChanged="OnPicker1SelectedIndexChanged" />

         <Button
            Margin="-10,-6,-10,-15"
            Clicked="DateButton_OnClicked"
            Style="{StaticResource buttonStyle}"
            Text="Choose Date" />
      </Grid>

      <Grid
         x:Name="BodyGrid"
         Grid.Row="2"
         Margin="0,0,0,-4"
         BackgroundColor="Black"
         VerticalOptions="FillAndExpand">

         <Grid.RowDefinitions>
            <RowDefinition Height="*" />
            <RowDefinition Height="*" />
            <RowDefinition Height="*" />
            <RowDefinition Height="*" />
            <RowDefinition Height="*" />
            <RowDefinition Height="*" />
            <RowDefinition Height="*" />
            <RowDefinition Height="*" />
            <RowDefinition Height="*" />
            <RowDefinition Height="*" />
            <RowDefinition Height="*" />
            <RowDefinition Height="*" />
            <RowDefinition Height="*" />
         </Grid.RowDefinitions>
```

```
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*" />
            <ColumnDefinition Width="*" />
            <ColumnDefinition Width="*" />
            <ColumnDefinition Width="*" />
            <ColumnDefinition Width="*" />
            <ColumnDefinition Width="*" />
        </Grid.ColumnDefinitions>

        <Label
            Grid.Row="0"
            Grid.ColumnSpan="6"
            Margin="0,2,0,-2"
            BackgroundColor="White"
            FontAttributes="Bold"
            FontSize="Body"
            HorizontalTextAlignment="Center"
            Text="Scores"
            VerticalTextAlignment="Center" />
        <Label
            Grid.Row="1"
            Grid.Column="0"
            Margin="0,-2,-2,-2"
            BackgroundColor="White"
            HorizontalTextAlignment="Center"
            Text="End"
            VerticalTextAlignment="Center" />
        <Label
            Grid.Row="1"
            Grid.Column="1"
            Margin="-2,-2,-2,-2"
            BackgroundColor="White"
            HorizontalTextAlignment="Center"
            Text="Arrow 1"
            VerticalTextAlignment="Center" />
        <Label
            Grid.Row="1"
            Grid.Column="2"
            Margin="-2,-2,-2,-2"
            BackgroundColor="White"
            HorizontalTextAlignment="Center"
            Text="Arrow 2"
            VerticalTextAlignment="Center" />
        <Label
            Grid.Row="1"
            Grid.Column="3"
            Margin="-2,-2,-2,-2"
            BackgroundColor="White"
            HorizontalTextAlignment="Center"
            Text="Arrow 3"
            VerticalTextAlignment="Center" />
        <Label
            Grid.Row="1"
            Grid.Column="4"
            Margin="-2,-2,-2,-2"
            BackgroundColor="White"
            HorizontalTextAlignment="Center"
            Text="Subtotal"
            VerticalTextAlignment="Center" />
        <Label
            Grid.Row="1"
            Grid.Column="5"
            Margin="-2,-2,0,-2"
            BackgroundColor="White"
            HorizontalTextAlignment="Center"
            Text="Total"
            VerticalTextAlignment="Center" />

        <Label
            Grid.Row="2"
            Grid.Column="0"
            Margin="0,-2,-2,-2"
            BackgroundColor="White"
            HorizontalTextAlignment="Center"
            Text="1"
            VerticalTextAlignment="Center" />
        <Label
            x:Name="end1arrow1"
            Grid.Row="2"
            Grid.Column="1"
            Margin="-2,-2,-2,-2"
```

```
            BackgroundColor="White"
            FontSize="Small"
            HorizontalTextAlignment="Center"
            VerticalTextAlignment="Center" />
        <Label
            x:Name="end1arrow2"
            Grid.Row="2"
            Grid.Column="2"
            Margin="-2,-2,-2,-2"
            BackgroundColor="White"
            FontSize="Small"
            HorizontalTextAlignment="Center"
            VerticalTextAlignment="Center" />
        <Label
            x:Name="end1arrow3"
            Grid.Row="2"
            Grid.Column="3"
            Margin="-2,-2,-2,-2"
            BackgroundColor="White"
            FontSize="Small"
            HorizontalTextAlignment="Center"
            VerticalTextAlignment="Center" />
        <Label
            x:Name="end1subtotal"
            Grid.Row="2"
            Grid.Column="4"
            Margin="-2,-2,-2,-2"
            BackgroundColor="White"
            FontSize="Small"
            HorizontalTextAlignment="Center"
            VerticalTextAlignment="Center" />
        <Grid
            Grid.Row="2"
            Grid.Column="5"
            Margin="-2,-2,0,-2" />

        <Label
            Grid.Row="3"
            Grid.Column="0"
            Margin="0,-2,-2,-2"
            BackgroundColor="White"
            HorizontalTextAlignment="Center"
            Text="2"
            VerticalTextAlignment="Center" />
        <Label
            x:Name="end2arrow1"
            Grid.Row="3"
            Grid.Column="1"
            Margin="-2,-2,-2,-2"
            BackgroundColor="White"
            FontSize="Small"
            HorizontalTextAlignment="Center"
            VerticalTextAlignment="Center" />
        <Label
            x:Name="end2arrow2"
            Grid.Row="3"
            Grid.Column="2"
            Margin="-2,-2,-2,-2"
            BackgroundColor="White"
            FontSize="Small"
            HorizontalTextAlignment="Center"
            VerticalTextAlignment="Center" />
        <Label
            x:Name="end2arrow3"
            Grid.Row="3"
            Grid.Column="3"
            Margin="-2,-2,-2,-2"
            BackgroundColor="White"
            FontSize="Small"
            HorizontalTextAlignment="Center"
            VerticalTextAlignment="Center" />
        <Label
            x:Name="end2subtotal"
            Grid.Row="3"
            Grid.Column="4"
            Margin="-2,-2,-2,-2"
            BackgroundColor="White"
            FontSize="Small"
            HorizontalTextAlignment="Center"
            VerticalTextAlignment="Center" />
        <Label
```

```
            x:Name="end2total"
            Grid.Row="3"
            Grid.Column="5"
            Margin="-2,-2,-2,-2"
            BackgroundColor="White"
            FontSize="Small"
            HorizontalTextAlignment="Center"
            VerticalTextAlignment="Center" />

.
.
.
.
.

        <Label
            Grid.Row="11"
            Grid.Column="0"
            Margin="0,-2,-2,0"
            BackgroundColor="White"
            HorizontalTextAlignment="Center"
            Text="10"
            VerticalTextAlignment="Center" />
        <Label
            x:Name="end10arrow1"
            Grid.Row="11"
            Grid.Column="1"
            Margin="-2,-2,-2,-2"
            BackgroundColor="White"
            FontSize="Small"
            HorizontalTextAlignment="Center"
            VerticalTextAlignment="Center" />
        <Label
            x:Name="end10arrow2"
            Grid.Row="11"
            Grid.Column="2"
            Margin="-2,-2,-2,-2"
            BackgroundColor="White"
            FontSize="Small"
            HorizontalTextAlignment="Center"
            VerticalTextAlignment="Center" />
        <Label
            x:Name="end10arrow3"
            Grid.Row="11"
            Grid.Column="3"
            Margin="-2,-2,-2,-2"
            BackgroundColor="White"
            FontSize="Small"
            HorizontalTextAlignment="Center"
            VerticalTextAlignment="Center" />
        <Label
            x:Name="end10subtotal"
            Grid.Row="11"
            Grid.Column="4"
            Margin="-2,-2,-2,-2"
            BackgroundColor="White"
            FontSize="Small"
            HorizontalTextAlignment="Center"
            VerticalTextAlignment="Center" />
        <Label
            x:Name="end10total"
            Grid.Row="11"
            Grid.Column="5"
            Margin="-2,-2,-2,0"
            BackgroundColor="White"
            FontSize="Small"
            HorizontalTextAlignment="Center"
            VerticalTextAlignment="Center" />
    </Grid>
  </Grid>
</ContentPage>
```

2.1.27 Views/ScoreView.xaml.cs

```
using BOOSTER.Data;
using BOOSTER.Models;
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using Xamarin.Essentials;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace BOOSTER
{
   [XamlCompilation(XamlCompilationOptions.Compile)]
   public partial class ScoreView : ContentPage
   {
      public string User { get; set; }
      public NetworkAccess CurrentNetworkStatus { get; set; }

      public List<string> list;
      public List<string> List
      {
         get { return list; }
         set { list = value; }
      }
      public ScoreView(string UserName, NetworkAccess currentNetworkStatus)
      {
         InitializeComponent();
         User = UserName;
         CurrentNetworkStatus = currentNetworkStatus;
         GetDates();

      }

      #region Date Picker
      void DateButton_OnClicked(object sender, System.EventArgs e)
      {
         picker1.IsEnabled = true;
         picker1.IsVisible = true;
         picker1.Focus();
      }

      public void OnPicker1SelectedIndexChanged(object sender, EventArgs e)
      {
         var picker1 = (Picker)sender;
         var selected = picker1.SelectedItem.ToString();

         GetAllScoresFromDate(User, selected);
         picker1.IsVisible = false;
         picker1.IsEnabled = false;
      }
      #endregion

      #region Firebase Lookups for dates
      public async void GetDates()
      {
         List<Scores> allSessions;
         List<SQLScores> allSQLSessions;
         if (CurrentNetworkStatus == NetworkAccess.Internet)
         {
            allSessions = await FirebaseHelper.GetAllScores(User);
            List = new List<string>();
            foreach (var item in allSessions)
            {
               var selectedDate = item.DTime.ToString();
               List.Add(selectedDate);
            }
            picker1.ItemsSource = List;
         }
         else
         {
            allSQLSessions = await SQLHelper.GetAllScores(User);
            List = new List<string>();
            foreach (var item in allSQLSessions)
            {
               var selectedDate = item.DTime.ToString();
               List.Add(selectedDate);
            }
            picker1.ItemsSource = List;
         }

      }

      public async void GetAllScoresFromDate(string username, string date)
      {
         List<Scores> allScores;
         List<SQLScores> allSQLScores;
         if (CurrentNetworkStatus == NetworkAccess.Internet)
```

```
        {
            allScores = await FirebaseHelper.GetAllScoresFromDate(username, date);

            var End1 = allScores.Select(a => a.End1).FirstOrDefault();
            end1arrow1.Text = End1[0].ToString();
            end1arrow2.Text = End1[1].ToString();
            end1arrow3.Text = End1[2].ToString();
            end1subtotal.Text = End1[3].ToString();

            var End2 = allScores.Select(a => a.End2).FirstOrDefault();
            end2arrow1.Text = End2[0].ToString();
            end2arrow2.Text = End2[1].ToString();
            end2arrow3.Text = End2[2].ToString();
            end2subtotal.Text = End2[3].ToString();
            end2total.Text = End2[4].ToString();

            var End3 = allScores.Select(a => a.End3).FirstOrDefault();
            end3arrow1.Text = End3[0].ToString();
            end3arrow2.Text = End3[1].ToString();
            end3arrow3.Text = End3[2].ToString();
            end3subtotal.Text = End3[3].ToString();
            end3total.Text = End3[4].ToString();

            var End4 = allScores.Select(a => a.End4).FirstOrDefault();
            end4arrow1.Text = End4[0].ToString();
            end4arrow2.Text = End4[1].ToString();
            end4arrow3.Text = End4[2].ToString();
            end4subtotal.Text = End4[3].ToString();
            end4total.Text = End4[4].ToString();

            var End5 = allScores.Select(a => a.End5).FirstOrDefault();
            end5arrow1.Text = End5[0].ToString();
            end5arrow2.Text = End5[1].ToString();
            end5arrow3.Text = End5[2].ToString();
            end5subtotal.Text = End5[3].ToString();
            end5total.Text = End5[4].ToString();

            var End6 = allScores.Select(a => a.End6).FirstOrDefault();
            end6arrow1.Text = End6[0].ToString();
            end6arrow2.Text = End6[1].ToString();
            end6arrow3.Text = End6[2].ToString();
            end6subtotal.Text = End6[3].ToString();
            end6total.Text = End6[4].ToString();

            var End7 = allScores.Select(a => a.End7).FirstOrDefault();
            end7arrow1.Text = End7[0].ToString();
            end7arrow2.Text = End7[1].ToString();
            end7arrow3.Text = End7[2].ToString();
            end7subtotal.Text = End7[3].ToString();
            end7total.Text = End7[4].ToString();

            var End8 = allScores.Select(a => a.End8).FirstOrDefault();
            end8arrow1.Text = End8[0].ToString();
            end8arrow2.Text = End8[1].ToString();
            end8arrow3.Text = End8[2].ToString();
            end8subtotal.Text = End8[3].ToString();
            end8total.Text = End8[4].ToString();

            var End9 = allScores.Select(a => a.End9).FirstOrDefault();
            end9arrow1.Text = End9[0].ToString();
            end9arrow2.Text = End9[1].ToString();
            end9arrow3.Text = End9[2].ToString();
            end9subtotal.Text = End9[3].ToString();
            end9total.Text = End9[4].ToString();

            var End10 = allScores.Select(a => a.End10).FirstOrDefault();
            end10arrow1.Text = End10[0].ToString();
            end10arrow2.Text = End10[1].ToString();
            end10arrow3.Text = End10[2].ToString();
            end10subtotal.Text = End10[3].ToString();
            end10total.Text = End10[4].ToString();
        }
        else
        {
            allSQLScores = await SQLHelper.GetAllScoresFromDate(username, date);

            var End1 = allSQLScores.Select(a => a.end1).FirstOrDefault().Split(',').ToList();
            end1arrow1.Text = End1[0].ToString();
            end1arrow2.Text = End1[1].ToString();
            end1arrow3.Text = End1[2].ToString();
            end1subtotal.Text = End1[3].ToString();
```

```
            var End2 = allSQLScores.Select(a => a.end2).FirstOrDefault().Split(',').ToList();
            end2arrow1.Text = End2[0].ToString();
            end2arrow2.Text = End2[1].ToString();
            end2arrow3.Text = End2[2].ToString();
            end2subtotal.Text = End2[3].ToString();
            end2total.Text = End2[4].ToString();

            var End3 = allSQLScores.Select(a => a.end3).FirstOrDefault().Split(',').ToList();
            end3arrow1.Text = End3[0].ToString();
            end3arrow2.Text = End3[1].ToString();
            end3arrow3.Text = End3[2].ToString();
            end3subtotal.Text = End3[3].ToString();
            end3total.Text = End3[4].ToString();

            var End4 = allSQLScores.Select(a => a.end4).FirstOrDefault().Split(',').ToList();
            end4arrow1.Text = End4[0].ToString();
            end4arrow2.Text = End4[1].ToString();
            end4arrow3.Text = End4[2].ToString();
            end4subtotal.Text = End4[3].ToString();
            end4total.Text = End4[4].ToString();

            var End5 = allSQLScores.Select(a => a.end5).FirstOrDefault().Split(',').ToList();
            end5arrow1.Text = End5[0].ToString();
            end5arrow2.Text = End5[1].ToString();
            end5arrow3.Text = End5[2].ToString();
            end5subtotal.Text = End5[3].ToString();
            end5total.Text = End5[4].ToString();

            var End6 = allSQLScores.Select(a => a.end6).FirstOrDefault().Split(',').ToList();
            end6arrow1.Text = End6[0].ToString();
            end6arrow2.Text = End6[1].ToString();
            end6arrow3.Text = End6[2].ToString();
            end6subtotal.Text = End6[3].ToString();
            end6total.Text = End6[4].ToString();

            var End7 = allSQLScores.Select(a => a.end7).FirstOrDefault().Split(',').ToList();
            end7arrow1.Text = End7[0].ToString();
            end7arrow2.Text = End7[1].ToString();
            end7arrow3.Text = End7[2].ToString();
            end7subtotal.Text = End7[3].ToString();
            end7total.Text = End7[4].ToString();

            var End8 = allSQLScores.Select(a => a.end8).FirstOrDefault().Split(',').ToList();
            end8arrow1.Text = End8[0].ToString();
            end8arrow2.Text = End8[1].ToString();
            end8arrow3.Text = End8[2].ToString();
            end8subtotal.Text = End8[3].ToString();
            end8total.Text = End8[4].ToString();

            var End9 = allSQLScores.Select(a => a.end9).FirstOrDefault().Split(',').ToList();
            end9arrow1.Text = End9[0].ToString();
            end9arrow2.Text = End9[1].ToString();
            end9arrow3.Text = End9[2].ToString();
            end9subtotal.Text = End9[3].ToString();
            end9total.Text = End9[4].ToString();

            var End10 = allSQLScores.Select(a => a.end10).FirstOrDefault().Split(',').ToList();
            end10arrow1.Text = End10[0].ToString();
            end10arrow2.Text = End10[1].ToString();
            end10arrow3.Text = End10[2].ToString();
            end10subtotal.Text = End10[3].ToString();
            end10total.Text = End10[4].ToString();
        }
    }
    #endregion
  }
}
```

2.1.28 Views/Strength.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage
  x:Class="BOOSTER.Strength"
  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"
  xmlns:opt="clr-namespace:BOOSTER.Data"
```

```
    Title="SnC"
    mc:Ignorable="d">
    <ContentPage.BindingContext>
        <opt:PickerOptions />
    </ContentPage.BindingContext>

    <ContentPage.Content>
        <Grid>
            <Grid.RowDefinitions>
                <RowDefinition Height=".2*" />
                <RowDefinition Height="*" />
            </Grid.RowDefinitions>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="*" />
            </Grid.ColumnDefinitions>

            <Grid
                x:Name="LogoGrid"
                Grid.Row="0"
                BackgroundColor="ForestGreen">
                <Grid.RowDefinitions>
                    <RowDefinition Height=".1*" />
                    <RowDefinition Height=".5*" />
                    <RowDefinition Height=".1*" />
                </Grid.RowDefinitions>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="*" />
                    <ColumnDefinition Width="*" />
                    <ColumnDefinition Width="*" />
                </Grid.ColumnDefinitions>
                <Picker
                    x:Name="picker"
                    ItemsSource="{Binding Options}"
                    SelectedIndexChanged="OnPickerSelectedIndexChanged" />
                <Image
                    Grid.Row="1"
                    Grid.Column="1"
                    Source="boosterimg" />

                <ImageButton
                    Grid.Row="1"
                    Grid.Column="2"
                    BackgroundColor="Transparent"
                    Clicked="OptionButton_Clicked"
                    HeightRequest="25"
                    HorizontalOptions="Center"
                    Source="threegears"
                    VerticalOptions="Center"
                    WidthRequest="25" />
            </Grid>

            <ScrollView
                Grid.Row="1"
                Margin="5,5,5,5"
                Orientation="Vertical"
                VerticalOptions="FillAndExpand">
                <StackLayout>
                    <Label
                        x:Name="StrengthLabel"
                        Margin="5,0,0,0"
                        FontSize="Title"
                        HorizontalTextAlignment="Center"
                        Text="Strength and Conditioning" />
                    <BoxView
                        HeightRequest="5"
                        HorizontalOptions="FillAndExpand"
                        Color="#22A558" />
                    <Label
                        x:Name="StrengthBody1"
                        FontAttributes="Bold"
                        FontSize="Large"
                        Text="Strength: 10 Reps * 3 Sets, increase sets as strength develops" />
                    <BoxView
                        HeightRequest="5"
                        HorizontalOptions="FillAndExpand"
                        Color="#22A558" />
                    <Label
                        x:Name="StrengthBody2"
                        FontSize="Large"
                        Text="Strength Exercise: Full draw for 3 seconds, Half draw for 7 seconds. Ensure proper form and alignment is maintain
ed" />
```

```
                <BoxView
                    HeightRequest="5"
                    HorizontalOptions="FillAndExpand"
                    Color="#22A558" />
                <Label
                    x:Name="StrengthBody3"
                    FontSize="Large"
                    Text="Conditioning Exercise: Full draw for 10 seconds, Rest for 70 seconds. Ensure proper form and alignment is maintained" />
                <BoxView
                    HeightRequest="5"
                    HorizontalOptions="FillAndExpand"
                    Color="#22A558" />
                <Label
                    x:Name="StrengthBody4"
                    FontSize="Large"
                    Text="Gym Exercises: Push ups, Dumbbell Curls, Planks, Situps/Crunches, Barbell Standing Press, Seated Dumbbell Press, Lateral Raise, Upright Row, Front Raises - Gain advice from your local gym supervisor on which exercises will work for you" />
            </StackLayout>
        </ScrollView>
    </Grid>
  </ContentPage.Content>
</ContentPage>
```

2.1.29 Views/Strength.xaml.cs

```
using BOOSTER.Views;
using System;
using Xamarin.Essentials;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace BOOSTER
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class Strength : ContentPage
    {
        public string User { get; set; }
        public NetworkAccess CurrentNetworkStatus { get; set; }
        public Strength(string UserName, NetworkAccess currentNetworkStatus)
        {
            InitializeComponent();
            User = UserName;
            CurrentNetworkStatus = currentNetworkStatus;
        }

        #region Picker
        void OptionButton_Clicked(object sender, System.EventArgs e)
        {
            picker.IsEnabled = true;
            picker.IsVisible = true;
            picker.Focus();
        }

        void OnPickerSelectedIndexChanged(object sender, EventArgs e)
        {
            var picker = (Picker)sender;
            int selectedIndex = picker.SelectedIndex;

            if (selectedIndex == 0)
            {
                Navigation.PushModalAsync(new Home(User, CurrentNetworkStatus));

            }
            else if (selectedIndex == 1)
            {
                Navigation.PushModalAsync(new ChangePassword(User, CurrentNetworkStatus));
            }
            else if (selectedIndex == 2)
            {
                Navigation.PushModalAsync(new LoginView(CurrentNetworkStatus));
            }
        }
        #endregion
    }
}
```

2.1.30 App.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<Application
   x:Class="BOOSTER.App"
   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">
   <Application.Resources>
      <ResourceDictionary>
         <Style x:Key="buttonStyle" TargetType="Button">
            <Setter Property="Margin" Value="5" />
            <Setter Property="BackgroundColor" Value="LightGreen" />
            <Setter Property="BorderColor" Value="Black" />
            <Setter Property="BorderRadius" Value="5" />
            <Setter Property="BorderWidth" Value="2" />
         </Style>
      </ResourceDictionary>
   </Application.Resources>
</Application>
```

2.1.31 App.xaml.cs

```
using BOOSTER.Data;
using BOOSTER.Models;
using PCLStorage;
using SQLite;
using System.IO;
using Xamarin.Essentials;
using Xamarin.Forms;

namespace BOOSTER
{
   public partial class App : Application
   {
      static SQLHelper database;
      public static SQLiteConnection databaseCon;
      public NetworkAccess currentNetworkStatus { get; set; }
      public App()
      {
         InitializeComponent();
         currentNetworkStatus = Connectivity.NetworkAccess;
         MainPage = new BOOSTER.LoginView(currentNetworkStatus);
      }

      public static SQLHelper Database
      {
         get
         {
            if (database == null)
            {
               database = new SQLHelper();
            }
            return database;
         }
      }

      protected override void OnStart()
      {
         var sqliteFilename = "BOOSTER.db3";
         string localPath = System.Environment.GetFolderPath(System.Environment.SpecialFolder.Personal);
         string path = PortablePath.Combine(localPath, sqliteFilename);
         //File.Delete(path);

         if (!File.Exists(path))
         {
            File.Create(path);
            databaseCon = SQLHelper.GetConnection();
            databaseCon.CreateTable<Users>();
            databaseCon.CreateTable<SQLScores>();
         }
      }

      protected override void OnSleep()
      {
```

```
        }

        protected override void OnResume()
        {
        }
    }
}
```

2.2 Android Project

2.2.1 MainActivity.cs

```
using Android.App;
using Android.Content.PM;
using Android.Runtime;
using Android.OS;
using Xamarin.Forms;
using BOOSTER.Views;
using System.Threading.Tasks;
using Android;
using Android.Widget;
using Android.Bluetooth;
using System;
using System.Linq;
using Java.Util;

namespace BOOSTER.Droid
{
    [Activity(Label = "BOOSTER.Droid",
        Icon = "@drawable/boosterimg",
        Theme = "@style/MainTheme",
        MainLauncher = false,
        ConfigurationChanges = ConfigChanges.ScreenSize | ConfigChanges.Orientation, ScreenOrientation = ScreenOrientation.Portrait)]

    public class MainActivity : global::Xamarin.Forms.Platform.Android.FormsAppCompatActivity
    {

        readonly Activity activity;
        BluetoothSocket _socket;
        BluetoothAdapter adapter;
        BluetoothDevice device;
        byte[] buffer = new byte[1024];
        int input;
        protected async override void OnCreate(Bundle savedInstanceState)
        {
            await TryToGetPermissions();
            TabLayoutResource = Resource.Layout.Tabbar;
            ToolbarResource = Resource.Layout.Toolbar;

            base.OnCreate(savedInstanceState);

            Xamarin.Essentials.Platform.Init(this, savedInstanceState);
            Forms.Init(this, savedInstanceState);
            OxyPlot.Xamarin.Forms.Platform.Android.PlotViewRenderer.Init();

            try
            {
                await ConnectToBluetooth();
            }
            catch
            {

            }

            LoadApplication(new App());

            //Register orientation of certain pages

            MessagingCenter.Subscribe<FormDraw>(this, "setLandscape", sender =>
            {
                RequestedOrientation = ScreenOrientation.Landscape;
            });
            MessagingCenter.Subscribe<FormDraw>(this, "setPortrait", sender =>
            {
                RequestedOrientation = ScreenOrientation.Portrait;
            });
```

```
        MessagingCenter.Subscribe<HeartRate>(this, "setLandscape", sender =>
        {
            RequestedOrientation = ScreenOrientation.Landscape;
        });
        MessagingCenter.Subscribe<HeartRate>(this, "setPortrait", sender =>
        {
            RequestedOrientation = ScreenOrientation.Portrait;
        });

    }

    #region Bluetooth
    async Task ConnectToBluetooth()
    {
        adapter = BluetoothAdapter.DefaultAdapter;
        if (adapter == null)
            throw new Exception("No Bluetooth adapter found.");

        if (!adapter.IsEnabled)
            throw new Exception("Bluetooth adapter is not enabled.");

        device = (from bd in adapter.BondedDevices
                  where bd.Name == "HC05"
                  select bd).FirstOrDefault();

        if (device == null)
            throw new Exception("Named device not found.");

        this._socket = device.CreateRfcommSocketToServiceRecord(UUID.FromString("00001101-0000-1000-8000-00805f9b34fb"));
        await _socket.ConnectAsync();
    }

    public async Task<int> readBluetoothAsync()
    {
        ConnectToBluetooth();
        return input = await this._socket.InputStream.ReadAsync(buffer, 0, buffer.Length);
    }

    #endregion

    #region Permissions
    async Task TryToGetPermissions()
    {
        if ((int)Build.VERSION.SdkInt >= 23)
        {
            await GetPermissionsAsync();
            return;
        }
    }

    private static string[] PermissionsGroup =
    {
        Manifest.Permission.Camera,
        Manifest.Permission.Internet,
        Manifest.Permission.BluetoothAdmin,
        Manifest.Permission.Bluetooth
    };

    int RequestLocationId;

    async Task GetPermissionsAsync()
    {
        for (RequestLocationId = 0; RequestLocationId < PermissionsGroup.Length; RequestLocationId++)
        {
            string permission = PermissionsGroup[RequestLocationId];

            if (CheckSelfPermission(permission) == (int)Android.Content.PM.Permission.Granted)
            {
                continue;
            }

            if (ShouldShowRequestPermissionRationale(permission))
            {
                //set alert for executing the task
                AlertDialog.Builder alert = new AlertDialog.Builder(this);
                alert.SetTitle("Permissions Needed");
                alert.SetMessage("The application needs "+ permission + " permission to continue");
                alert.SetPositiveButton("Request Permissions", (senderAlert, args) =>
                {
                    RequestPermissions(PermissionsGroup, RequestLocationId);
                });
```

```
                alert.SetNegativeButton("Cancel", (senderAlert, args) =>
                {
                    Toast.MakeText(this, "Cancelled!", ToastLength.Short).Show();
                });

                Dialog dialog = alert.Create();
                dialog.Show();
                continue;
            }

            RequestPermissions(PermissionsGroup, RequestLocationId);
        }

    }

    public override async void OnRequestPermissionsResult(int requestCode, string[] permissions, [GeneratedEnum] Android.Content.
PM.Permission[] grantResults)
    {

        if (grantResults[0] == (int)Android.Content.PM.Permission.Granted)
        {
            Toast.MakeText(this, "Permissions granted", ToastLength.Short).Show();

        }
        else //Permission Denied
        {
            Toast.MakeText(this, "Permissions denied", ToastLength.Short).Show();
        }

        Xamarin.Essentials.Platform.OnRequestPermissionsResult(requestCode, permissions, grantResults);
        base.OnRequestPermissionsResult(requestCode, permissions, grantResults);
    }
    #endregion
  }
}
```

2.2.2 SplashActivity.cs

```
using Android.App;
using Android.OS;

namespace BOOSTER.Droid
{
  [Activity(Theme="@style/Theme.Splash", MainLauncher = true, NoHistory = true ,Label = "BOOSTER")]
  public class SplashActivity : Activity
  {
    protected override void OnCreate(Bundle savedInstanceState)
    {
      base.OnCreate(savedInstanceState);

      // Create your application here
      StartActivity(typeof(MainActivity));
    }
  }
}
```

2.3 iOS Project

2.3.1 AppDeligate.cs

```
using Foundation;
//using Plugin.BLE.Abstractions.Contracts;
//using Plugin.BLE.iOS;
using UIKit;

namespace BOOSTER.iOS
{
  // The UIApplicationDelegate for the application. This class is responsible for launching the
  // User Interface of the application, as well as listening (and optionally responding) to
  // application events from iOS.
  [Register("AppDelegate")]
  public partial class AppDelegate : global::Xamarin.Forms.Platform.iOS.FormsApplicationDelegate
  {
```

```
        //
        // This method is invoked when the application has loaded and is ready to run. In this
        // method you should instantiate the window, load the UI into it and then make the window
        // visible.
        //
        // You have 17 seconds to return from this method, or iOS will terminate your application.
        //
        public override bool FinishedLaunching(UIApplication app, NSDictionary options)
        {
            global::Xamarin.Forms.Forms.Init();
            OxyPlot.Xamarin.Forms.Platform.iOS.PlotViewRenderer.Init();
            LoadApplication(new App());

            return base.FinishedLaunching(app, options);
        }
    }
}
```

2.4 Wearable Code

The gyroscope code to concert the readings to an angle rather than angular velocity has been adapted from code taken from the Arduino forum [Dhru14].

```
#include <Wire.h>
#include <L3G.h>
#include <SoftwareSerial.h>
L3G gyro0;
L3G gyro1;

#define  CTRL_REG1  0x20
#define  CTRL_REG2  0x21
#define  CTRL_REG3  0x22
#define  CTRL_REG4  0x23
#define  CTRL_REG5  0x24

#define TCAADDR 0x70

int gyroI2CAddr=105;

int gyroRaw[3];
double gyroDPS[3];
float heading[3]={0.0f};

int gyroRaw2[3];
double gyroDPS2[3];
float heading2[3]={0.0f};

int gyroZeroRate[3];
int gyroThreshold[3];

int gyroZeroRate2[3];
int gyroThreshold2[3];

#define  NUM_GYRO_SAMPLES  100
#define  GYRO_SIGMA_MULTIPLE  1

float dpsPerDigit=.0175f;
int calibrateCounter = 0;

#define USE_ARDUINO_INTERRUPTS true    // Set-up low-level interrupts for most acurate BPM math.
#include <PulseSensorPlayground.h>     // Includes the PulseSensorPlayground Library.

//  Variables
const int PulseWire = A0;       // PulseSensor PURPLE WIRE connected to ANALOG PIN 0
int Threshold = 550;
PulseSensorPlayground pulseSensor;  // Creates an instance of the PulseSensorPlayground object called "pulseSensor"
SoftwareSerial BTSerial(10, 11);

void setup() {
  Serial.begin(9600);
  pulseSensor.begin();

  pinMode(9, OUTPUT);  // this pin will pull the HC-05 pin 34 (key pin) HIGH to switch module to AT mode
  digitalWrite(9, HIGH);
  BTSerial.begin(38400);  // HC-05 default speed in AT command more

  pulseSensor.analogInput(PulseWire);
```

```
  pulseSensor.setThreshold(Threshold);

  Wire.begin();

//  Serial.println("################################");
//  Serial.println("Starting Initialization");
//  Serial.println("################################");
  dpsPerDigit=.0175f;

  tcaselect(0);
  gyro0.init();
  gyro0.writeReg(CTRL_REG1, 0x1F);
  gyro0.writeReg(CTRL_REG3, 0x08);
  gyro0.writeReg(CTRL_REG4, 0x90);

  //calibrateGyro();
//  Serial.println();
//  Serial.println("################################");          //30-times #
//  Serial.println("gyro 1 Initialization Finished");
//  Serial.println("################################");          //30-times #

  tcaselect(1);
  gyro1.init();
  gyro1.writeReg(CTRL_REG1, 0x1F);
  gyro1.writeReg(CTRL_REG3, 0x08);
  gyro1.writeReg(CTRL_REG4, 0x90);
  //calibrateGyro();

//  Serial.println();
//  Serial.println("################################");          //30-times #
//  Serial.println("gyro 2 Initialization Finished");
//  Serial.println("################################");          //30-times #
//  Serial.println();
//  Serial.println();
}

void loop() {
  if(calibrateCounter == 50){
     calibrateGyro();
     calibrateGyro2();
  }
  tcaselect(0);
//  Serial.print("Gyro 1 readings: ");
  gyro0.read();
  updateGyroValues();
  updateHeadings();
  printHeadings();
  Serial.println();

  tcaselect(1);
  gyro1.read();
//  Serial.print("Gyro 2 readings: ");
  updateGyroValues2();
  updateHeadings2();
  printHeadings2();
  Serial.println();

  int myBPM = pulseSensor.getBeatsPerMinute();  // Calls function on our pulseSensor object that returns BPM as an "int".
                                                // "myBPM" hold this BPM value now.
  Serial.print("BPM: ");                        // Print phrase "BPM: "
  Serial.println(myBPM);
  BTSerial.write(myBPM);
  calibrateCounter++;
  delay(200);
}

void tcaselect (uint8_t i) {
  if (i > 7) return;

  Wire.beginTransmission(TCAADDR);
  Wire.write(1 << i);
  Wire.endTransmission();
}

void printHeadings()
{
  Serial.print("Heading X: ");
  Serial.print(heading[0]);
  Serial.print(",");
  Serial.print("  Y: ");
  Serial.print(heading[1]);
```

```
  Serial.print(",");
  Serial.print("  Z: ");
  Serial.print(heading[2]);

  BTSerial.write(heading[0]);
  BTSerial.write(",");
  BTSerial.write(heading[1]);
  BTSerial.write(",");
  BTSerial.write(heading[2]);
  BTSerial.write(",");
}

void printHeadings2()
{
  Serial.print("Heading X: ");
  Serial.print(heading2[0]);
  Serial.print(",");
  Serial.print("  Y: ");
  Serial.print(heading2[1]);
  Serial.print(",");
  Serial.print("  Z: ");
  Serial.print(heading2[2]);

  BTSerial.write(heading2[0]);
  BTSerial.write(",");
  BTSerial.write(heading2[1]);
  BTSerial.write(",");
  BTSerial.write(heading2[2]);
  BTSerial.write(",");
}

void updateHeadings()
{

  float deltaT=getDeltaTMicros();

  for (int j=0;j<3;j++)
    heading[j] -= (gyroDPS[j]*deltaT)/1000000.0f;
}

void updateHeadings2()
{

  float deltaT2=getDeltaTMicros2();

  for (int j=0;j<3;j++)
    heading2[j] -= (gyroDPS2[j]*deltaT2)/1000000.0f;
}

unsigned long getDeltaTMicros()
{
  static unsigned long lastTime=0;

  unsigned long currentTime=micros();

  unsigned long deltaT=currentTime-lastTime;
  if (deltaT < 0.0)
    deltaT=currentTime+(4294967295-lastTime);

  lastTime=currentTime;

  return deltaT;
}

unsigned long getDeltaTMicros2()
{
  static unsigned long lastTime2=0;

  unsigned long currentTime2=micros();

  unsigned long deltaT2=currentTime2-lastTime2;
  if (deltaT2 < 0.0)
    deltaT2=currentTime2+(4294967295-lastTime2);

  lastTime2=currentTime2;

  return deltaT2;
}
void testCalibration()
{
  calibrateGyro();
```

```
  for (int j=0;j<3;j++)
  {
    Serial.print(gyroZeroRate[j]);
    Serial.print("  ");
    Serial.print(gyroThreshold[j]);
    Serial.print("  ");
  }
  Serial.println();
  return;
}


void calibrateGyro()
{
  long int gyroSums[3]={0};
  long int gyroSigma[3]={0};
  for (int i=0;i<NUM_GYRO_SAMPLES;i++)
  {
    updateGyroValues();
    for (int j=0;j<3;j++)
    {
      gyroSums[j]+=gyroRaw[j];
      gyroSigma[j]+=gyroRaw[j]*gyroRaw[j];
    }
  }

  for (int j=0;j<3;j++)
  {
    int averageRate=gyroSums[j]/NUM_GYRO_SAMPLES;
    gyroZeroRate[j]=averageRate;
    gyroThreshold[j]=sqrt((double(gyroSigma[j]) / NUM_GYRO_SAMPLES) - (averageRate * averageRate)) *
GYRO_SIGMA_MULTIPLE;
  }
}

void calibrateGyro2()
{
  long int gyroSums2[3]={0};
  long int gyroSigma2[3]={0};
  for (int i=0;i<NUM_GYRO_SAMPLES;i++)
  {
    updateGyroValues2();
    for (int j=0;j<3;j++)
    {
      gyroSums2[j]+=gyroRaw2[j];
      gyroSigma2[j]+=gyroRaw2[j]*gyroRaw2[j];
    }
  }
  for (int j=0;j<3;j++)
  {
    int averageRate2=gyroSums2[j]/NUM_GYRO_SAMPLES;
    gyroZeroRate2[j]=averageRate2;
    gyroThreshold2[j]=sqrt((double(gyroSigma2[j]) / NUM_GYRO_SAMPLES) - (averageRate2 * averageRate2)) *
GYRO_SIGMA_MULTIPLE;
  }
}

void updateGyroValues() {

  //while (!(gyroReadI2C(0x27) & B00001000)){}
  int reg=0x28;
  for (int j=0;j<3;j++)
  {
    gyroRaw[j]=(gyroReadI2C(reg) | (gyroReadI2C(reg+1)<<8));
    reg+=2;
  }
  //Serial.println("updateGyro delta array variables");
  int deltaGyro[3];
  for (int j=0;j<3;j++)
  {
    deltaGyro[j]=gyroRaw[j]-gyroZeroRate[j];
    if (abs(deltaGyro[j]) < gyroThreshold[j])
      deltaGyro[j]=0;
    gyroDPS[j]= dpsPerDigit * deltaGyro[j];
  }
}

void updateGyroValues2() {

  //while (!(gyroReadI2C(0x27) & B00001000)){}
  int reg=0x28;
```

```
  for (int j=0;j<3;j++)
  {
    gyroRaw2[j]=(gyroReadI2C(reg) | (gyroReadI2C(reg+1)<<8));
    reg+=2;
  }
  //Serial.println("updateGyro delta array variables");
  int deltaGyro2[3];
  for (int j=0;j<3;j++)
  {
    deltaGyro2[j]=gyroRaw2[j]-gyroZeroRate2[j];
    if (abs(deltaGyro2[j]) < gyroThreshold2[j])
      deltaGyro2[j]=0;
    gyroDPS2[j]= dpsPerDigit * deltaGyro2[j];
  }
}

int gyroReadI2C (byte regAddr)
{
  Wire.beginTransmission(gyroI2CAddr);
  //Serial.println("gyroI2CAddr read");
  Wire.write(regAddr);
  //Serial.println("gyroI2CAddr wire.write");
  Wire.endTransmission();
  //Serial.println("gyroI2CAddr end transmission");
  Wire.requestFrom(gyroI2CAddr, 1);
  //Serial.println("gyroI2CAddr request from");
  while(!Wire.available()) {};
  return (Wire.read());

}

int gyroWriteI2C( byte regAddr, byte val)
{
  Wire.beginTransmission(gyroI2CAddr);
//  Serial.println("gyroI2CAddr write");
  Wire.write(regAddr);
  Wire.write(val);
  Wire.endTransmission();
}
```

Appendix

i. Bibliography

Dell16 Dellai, H., (2016). [online] Available at: <https://www.youtube.com/watch?v=DJYLrVNY2ak> [Accessed 12 February 2020].

Docs18 Docs.microsoft.com. (2018). *Implementing A View - Xamarin*. [online] Available at: <https://docs.microsoft.com/en-us/xamarin/xamarin-forms/app-fundamentals/custom-renderer/view> [Accessed 11 February 2020].

Pala17 Palani, L., (2017). *Xamarin.Forms - Camera App*. [online] C-sharpcorner.com. Available at: <https://www.c-sharpcorner.com/article/xamarin-forms-camera-app3/> [Accessed 10 February 2020].

Dhru14 dhruba127, (2014). *L3G4200D 3 Axis Gyroscope With Voltage Regulator For Robot Tilting Control*. [online] Forum.arduino.cc. Available at: <https://forum.arduino.cc/index.php?topic=208021.0> [Accessed 27 February 2020].

ii. Plagiarism Declaration

· I declare that all material in this submission e.g. thesis/essay/project/assignment is entirely my/our own work except where duly acknowledged.

· I have cited the sources of all quotations, paraphrases, summaries of information, tables, diagrams or other material; including software and other electronic media in which intellectual property rights may reside.

· I have provided a complete bibliography of all works and sources used in the preparation of this submission.

· I understand that failure to comply with the Institute's regulations governing plagiarism constitutes a serious offense.

Student Name: Brendan Browne

Student Number: C00223413

Signature:

20/04/2020