

NORTHWEST NAZARNE UNIVERSITY

Basketball Workout Application

Thesis

Submitted to the department of Mathematics and Computer Science

In partial fulfillment of the requirements

For the degree of

BACHELOR OF SCIENCE

Ethan McPherson

2021

Thesis

Submitted to the department of Mathematics and Computer Science

In partial fulfillment of the requirements

For the degree of

BACHELOR OF SCIENCE

Ethan McPherson

2021

Basketball Workout Application

Author:



Ethan McPherson

Approved:



Dale Hamilton, Ph.D., Professor, Department of Mathematics and
Computer Science, Faculty Advisor

Approved:



Anna Lee, Adjunct Professor, Department of Communication Arts
and Sciences, Second Reader

Approved:



Barry L. Myers, Ph.D., Chair, Department of Mathematics and
Computer Science

Abstract

Mobile application for sending and tracking basketball workouts between coaches and players.

MCPHERSON, ETHAN (Department of Mathematics and Computer Science),

MYERS, DR. BARRY (Department of Mathematics and Computer Science)

Communication is key in the life of an athlete. This project is intended to help collegiate athletes reach their full potential for communication with coaches about workouts throughout the year. The main problem that this project sought to solve was the ability for both the IOS and Android platforms to have a place where they would receive the same information with no disconnect, eliminating a need for coaches to require a certain platform in communicating with their players. The next largest task was to give coaches the ability to send workouts to players on their roster and not make them mandatory in order to comply with the countable athletically related activity, or CARA, rules regarding workouts. Finally, the last part of this project was to create a way for players to track their personal workouts to see their growth in the off season. Future work includes data calculations to allow athletes to see their growth as a player.

Acknowledgements

First of all, I would like to thank my mom, Melissa, my dad, Charles, and siblings, Garrett, Cayden, and Macy. Without their love and support I would not have been able to get through college and earn my degree. I would also like to thank all of my basketball coaches, Paul Rush, Levi Stuart, Jon Hawkins, and Chris Foss. Their constant outpour of support helped shape me into the man I am today. Finally, I would like to thank Dr. Dale Hamilton and Dr. Barry Myers for pushing me to work hard in all of my classes and reach for goals that might have seemed out of reach in the beginning.

Table of Contents

Abstract	iii
Acknowledgements	iv
Table of Figures	vii
Background	1
Xamarin	2
Firebase	3
Realtime Database	4
Classes	5
Player Side	8
Coaching Side	9
Registering	10
Different Pages for Users	11
Challenges	12
Results	13
Future Work	13
References	15
Appendices	16
Appendix A – Functionality code:	16
Appendix A1 – CoachTeamInfo.xaml.cs	16
Appendix A2 – CoachTeamRegistration.xaml.cs	18
Appendix A3 – CoachWorkoutEntry.xaml.cs	20
Appendix A4 – CoachPlayerWorkouts.xaml.cs	22
Appendix A5 – Dashboard.xaml.cs	24
Appendix A6 – HomePage.xaml.cs	28
Appendix A7 – LoginPage.xaml.cs	29
Appendix A8 – PlayerRegistration.xaml.cs	32
Appendix A9 – RegistrationPage.xaml.cs	34
Appendix A10 – TeamWorkoutList.xaml.cs	37
Appendix A11 – WorkoutEntry.xaml.cs	39
Appendix A12 – WorkoutList.xaml.cs	41
Appendix A13 – WorkoutViewPage.xaml.cs	43
Appendix A14 – WorkoutViewPageP.xaml.cs	45

Appendix B – Styling code:	47
Appendix B1 – CoachTeamInfo.xaml	47
Appendix B2 – CoachTeamRegistration.xaml	47
Appendix B3 – CoachWorkoutEntry.xaml	49
Appendix B4 – CoachPlayerWorkouts.xaml	50
Appendix B5 – Dashboard.xaml	51
Appendix B6 – HomePage.xaml	53
Appendix B7 – LoginPage.xaml	54
Appendix B8 – PlayerRegistration.xaml	55
Appendix B9 – RegistrationPage.xaml	56
Appendix B10 – TeamWorkoutList.xaml	57
Appendix B11 – WorkoutEntry.xaml	58
Appendix B12 – WorkoutList.xaml	59
Appendix B13 – WorkoutViewPage.xaml	60
Appendix B14 – WorkoutViewPageP.xaml	61
Appendix C – Classes	62
Appendix C1 – DashboardMenu.cs	62
Appendix C2 – FireBHelp.cs	63
Appendix C3 – Team.cs	72
Appendix C4 – TeamWorkouts.cs	73
Appendix C5 – UserInfo.cs	74
Appendix C6 – Users.cs	76
Appendix C7 – UserSettings.cs	77
Appendix C8 – Workouts.cs	79

Table of Figures

Figure 1: Table Setup of Firebase Database.....	4
Figure 2: FireBHelp Snippet	5
Figure 3: UserSettings Class	6
Figure 4: Workouts Class.....	7
Figure 5: Player Personal Workout Page	8
Figure 6: Coaches Roster Page	9
Figure 7: Registration Page	10
Figure 8: Player's Side Dashboard.....	11
Figure 9: Coach's Side Dashboard.....	11

Background

Communication is an important, if not the most important, aspect of any team. This especially rings true for basketball teams. Any given person on the team must know their role, where they personally are supposed to be at, what to say to let their teammates know what is happening, and let others know where they are supposed to be at. Outside of the gym, communication is just as important for the team. Players must know when practices, film, meetings, and workouts are. The workout application seeks to make communication about optional summer workouts easier, as well as provide a way for players to track their past workouts.

The idea for this project comes from some involvement over the past four years with the Northwest Nazarene University (NNU) men's basketball team. Communicating over the summer with players by text is doable; however, with some players having iPhones and others having Android, there were times when texting all players at once became a problem. Not everyone would receive the texts, making it difficult to hold conversations with everyone at once, and multiple chats were needed to communicate. Because these conversations were mainly about workouts, and shooting drills and conditioning, it was proposed that an application should be created to allow coaches to send workouts to their players, and for players to store and track past workouts that they have completed.

This project follows the process of the development of an app for athletes and coaches that meets the needs of tracking and communication about workouts.

Xamarin

The first decision made was which language should be used to code this application in. Android development is done mainly in Java. With having experience in using Java, starting there seemed like the best option, but IOS does not support Java development. Now, IOS has its own coding language called Swift. Learning Swift would take up more development time and proved daunting to try and learn a new language to code in, especially with the short amount of time to get the application operational.

After a bit of research, the decision was made to code using a language called Xamarin, whose sole purpose is the creation of cross-platform applications and development. This would allow the code to be produced in one language and then giving it the ability to be deployed on both Android and Apple products.

Xamarin uses a mixture of C# and HTML to create their applications. With a small amount of knowledge in C#, some more practice and research were needed in order to create an easy to use and functional user interface (UI). Along with attention to details of application design, Xamarin has many integrated libraries and most importantly a way to connect with a cloud database on both platforms.

Firestore

After deciding to use C# and Xamarin to create the application, it was time to implement an important aspect that would provide some security, authentication. Authentication is used to give access to certain users after they authenticate their identity. The app intended to give only registered users access to the application and the associated data, and to control who has access to it. A connection with Google Firestore was established. Firestore is a platform originally developed by an independent team in 2011 but was acquired by Google in 2014. Google's intended use is for mobile and web application developers. More specifically it is a back-end as a service platform (BAAS). A BAAS is a model for providing web app and mobile app developers with a way to link their applications to back-end cloud storage and API's exposed by backend applications while also providing features such as user management, push notifications, and integration with social networking services. This allows developers to focus on the front-end of development such as the user interface and the client-side logic.

For the purposes of this project, Firestore worked extremely well. The authentication that they provide is fairly light-weight, and it allowed the creation of rules to let certain users read and write data. Now not everyone has access to alter all information that is stored.

Firestore not only provides the means to have user authentication, but another service that they provide is a real time database. This database allows a user to store the information about the workouts that they are doing, and it will also allow a coach to add optional workouts that they want their players to complete.

Realtime Database

Along with some of the other features that Firestore offers, it provides their users with access to a real time database. This database is a cloud-hosted database. The data is stored as JSON, an open standard file format, and data interchange format, that uses human-readable text to store and transmit data objects (Figure 1).

This data is synchronized in real time to every connected client. The synchronization is

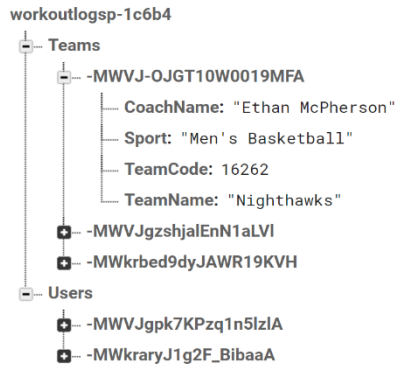


Figure 1: Table Setup of Firestore Database

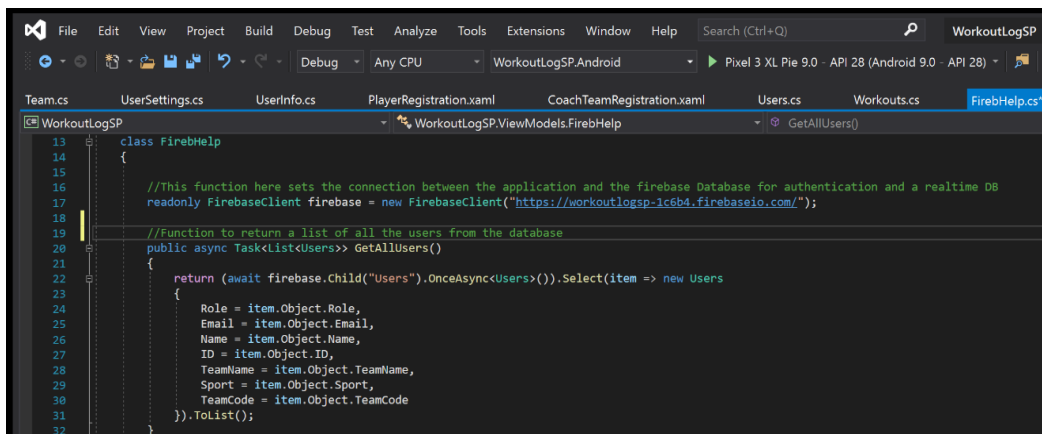
to make sure all the information is relevant and consistent between all users. Information becomes immediately visible to anyone who is utilizing the app.

Another feature of the database that is offered is that Firestore applications remain responsive even when offline. The Firestore database SDK persists the data to the disk. Once a connectivity is reestablished, then a user will receive all

the updates to any information that was made by a user. This feature is important because the workouts that were added while offline will still be available for viewing by players and coaches even when they go offline, and then updating the database as soon as they are connected again.

Classes

In order to give a user access to the information that they needed, as well as a way to store all of the information as an entry in the database, the use of classes was implemented. In the project there are a total of eight classes, and they can be split into two categories to help describe the role that they have in the application. The first four classes can be labeled as utility. This group includes a class that is dedicated to the declaration of functions to be used with the Firebase database called FireBHelp (Figure 2). The functions in the class use C.R.U.D which stands for create, read, update, and delete. These functions involve adding and deleting players, adding and deleting teams, adding and deleting workouts,

The image shows a screenshot of the Visual Studio code editor. The top menu bar includes File, Edit, View, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, and Help. The search bar contains 'Search (Ctrl+Q)'. The current project is 'WorkoutLogSP' and the target device is 'Pixel 3 XL Pie 9.0 - API 28 (Android 9.0 - API 28)'. The file explorer on the left shows the project structure, with 'WorkoutLogSP' expanded to show 'WorkoutLogSP.ViewModels.FireBHelp'. The code editor displays the following C# code snippet:

```
13 class FireBHelp
14 {
15
16     //This function here sets the connection between the application and the firebase Database for authentication and a realtime DB
17     readonly FirebaseClient firebase = new FirebaseClient("https://workoutlogsp-1c6b4.firebaseio.com/");
18
19     //Function to return a list of all the users from the database
20     public async Task<List<Users>> GetAllUsers()
21     {
22         return (await firebase.Child("Users").OnceAsync<Users>()).Select(item => new Users
23         {
24             Role = item.Object.Role,
25             Email = item.Object.Email,
26             Name = item.Object.Name,
27             ID = item.Object.ID,
28             TeamName = item.Object.TeamName,
29             Sport = item.Object.Sport,
30             TeamCode = item.Object.TeamCode
31         }).ToList();
32     }
}
```

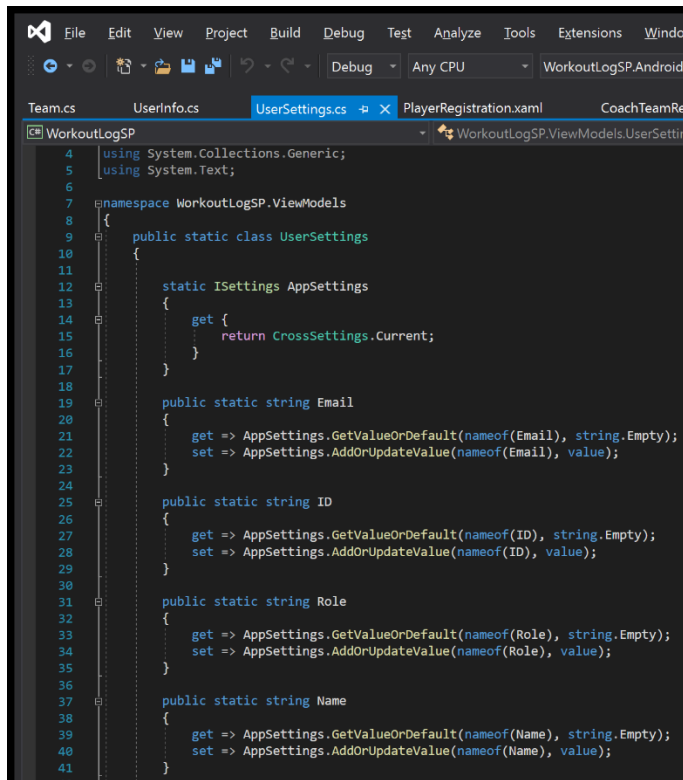
Figure 2: FireBHelp Snippet

updating teams, updating user information, as well as functions to search for entries within the database and return them.

The next one is a class called DashboardMenu. This class functions as a driver into the main layout of the application. This class allows for a master detail page to display all the other pages of the application in a left sidebar menu. On top of that, it allows a custom picture and display of the menu, showing the current users email, and a main logo for the app.

The third and fourth classes build off each other, and they are the UserInfo and the UserSettings. The

UserSettings class is used to set the application settings of the local device to the current user, which has been collected from the firebase database upon login (Figure 3).

A screenshot of the Visual Studio IDE showing the code for the UserSettings class. The code is in C# and is part of the WorkoutLogSP project. It defines a static class UserSettings with properties for Email, ID, Role, and Name, each with a getter and setter that interacts with AppSettings. The code is as follows:

```
4 using System.Collections.Generic;
5 using System.Text;
6
7 namespace WorkoutLogSP.ViewModels
8 {
9     public static class UserSettings
10    {
11        static ISettings AppSettings
12        {
13            get {
14                return CrossSettings.Current;
15            }
16        }
17
18        public static string Email
19        {
20            get => AppSettings.GetValueOrDefault(nameof(Email), string.Empty);
21            set => AppSettings.AddOrUpdateValue(nameof(Email), value);
22        }
23
24        public static string ID
25        {
26            get => AppSettings.GetValueOrDefault(nameof(ID), string.Empty);
27            set => AppSettings.AddOrUpdateValue(nameof(ID), value);
28        }
29
30        public static string Role
31        {
32            get => AppSettings.GetValueOrDefault(nameof(Role), string.Empty);
33            set => AppSettings.AddOrUpdateValue(nameof(Role), value);
34        }
35
36        public static string Name
37        {
38            get => AppSettings.GetValueOrDefault(nameof(Name), string.Empty);
39            set => AppSettings.AddOrUpdateValue(nameof(Name), value);
40        }
41    }
42 }
```

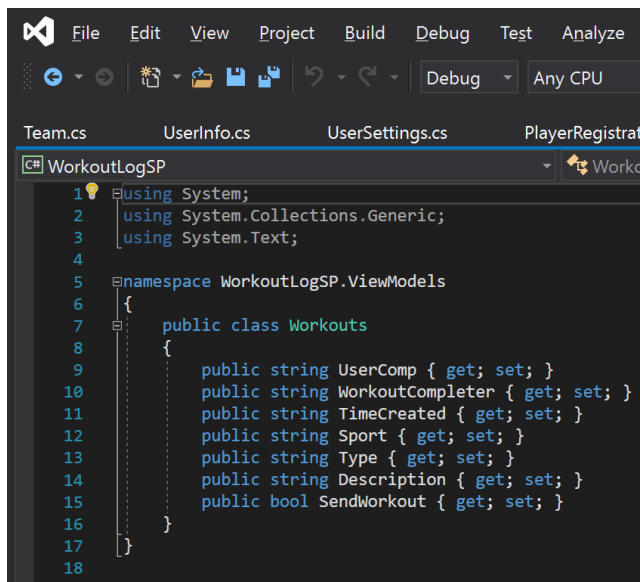
Figure 3: UserSettings Class

The UserSettings/UserInfo information is also used in the creation of a workout as well as displaying all the past workouts that a player has saved.

The second set of classes all have one goal in common: to store information as an object. These classes are named Users, Team, Workouts, and TeamWorkouts. The Users class stores all the information of each individual, both coaches and players, an associated email address, a unique I.D, their name, a team name, the sport they participate in, their role on the team, and the team code. This team code will connect the user to their coach and their team.

The Team class stores the team name, a unique I.D, the sport type, and the name of the coach. This creates a connection between the coach and player tables in the database.

The Workout class is used by players to store a workout type, a user component, the time created, a description, the sport, who completed it, and an option to make the workout visible to the coach (Figure 4). The send workout



```
1 using System;
2 using System.Collections.Generic;
3 using System.Text;
4
5 namespace WorkoutLogSP.ViewModels
6 {
7     public class Workouts
8     {
9         public string UserComp { get; set; }
10        public string WorkoutCompleter { get; set; }
11        public string TimeCreated { get; set; }
12        public string Sport { get; set; }
13        public string Type { get; set; }
14        public string Description { get; set; }
15        public bool SendWorkout { get; set; }
16    }
17 }
18
```

Figure 4: Workouts Class

option is an important addition to the individual workouts. During the summer, college coaches are not able to make any workouts or sessions mandatory as per the CARA rules. By having the send workout option, it allows this

application to be used year-round by coaches, rather than having access to it only at certain times.

Finally, the TeamWorkouts class is used by a coach to send the workouts they have created to the players. When a team workout is created, the workout is stored and appears as a list to the players, starting with the most recently added one. Coaches are not able to view who does the workouts, but they do have the option to delete a workout so that it does not clutter or take up too much space within the database.

Player Side

As a player using the application, players have access to options that will help improve their skill set. They have the ability to store any workout they choose. In the workout creation page, there are three options for the type of workout that a player might complete, both while in and out of season. The options are Weightlifting, Conditioning, or Sport Specific. The goal for this is to make the application usable by any sport later on. When players save a workout, they will enter in a short description of the workout. This ranges from the repetitions they completed, to the time taken, all the way down to the specific moves they worked on (Figure 5). By having the descriptions



Figure 5: Player Personal Workout Page

identifying what they did, players will be able to look back at their past workouts and see the progress they have made.

Players will have access to see the workouts that their coach creates as well. A separate page in the application was created allowing players to clearly see what the workout to be completed is. The page is set up to display the type of workout in chronological order. Underneath the declared type of workout is a description of the rest of the workout, i.e., how many reps, how much they should be expecting to lift, and if they need to do any conditioning or other extra additions to the workout.

Coaching Side

This application is intended to be utilized by coaches to send workouts to their

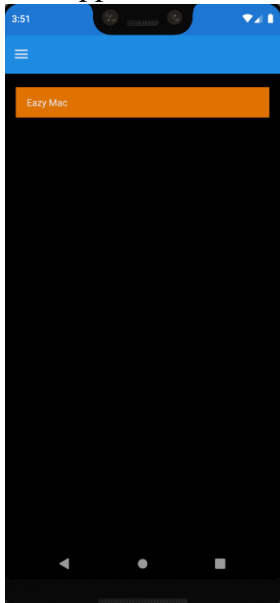


Figure 6: Coaches Roster Page

players, and the goal was to make the application practical and easy for them to enter in a workout. Coaches also need access to see the workouts their players send to them. The coaches have an easy workout entry page as well as a team info page, which shows all players that are signed up to the application using the team code associated with the coach and team. From this page, A coach should be able to click on the player's name, and

have a page display the workouts players have opted to share with their coach (Figure 6).

In a tradeoff for having access to the team roster page, a coach does not have access to create individual workouts. Because of this, the application will not have a clutter of pages in the sidebar menu. Coaches have no need for entering individual workouts either, so taking this away saves space within the database. This will save time accessing workouts while using the application.

Registering

When a user first registers for the application, they will have the option to select whether they are a player or a coach. If coach is selected then it will store their name, email, a unique randomly generated I.D, a team code, and a team name. (Figure 7).

When a user registers and the player box is checked, it will store the same information except it will require players to enter in a team code that a coach gives to them. This way it will connect all the users to the same team and team code, allowing easier and more secure access to the information

being stored in the workout tables of the database, as well as give them access within the application to the pages that they need to have a quality experience with the app.

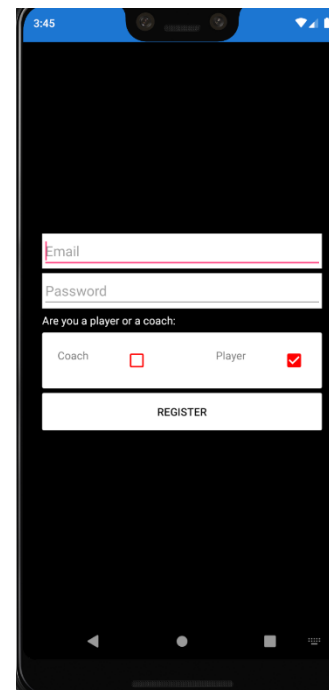


Figure 7: Registration Page

Different Pages for Users

After a user is registered, the application will take them back to the login page. The user then enters their credentials, and the app will bring them to the

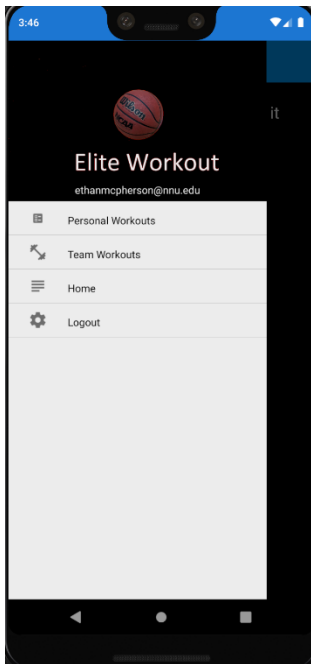


Figure 8: Player's Side Dashboard

home page of the app giving a small overview of the application.

When a user logs in, the unique I.D is retrieved from the database, then saved locally in the application. This will give the application the information it needs to store a workout and then later on, go and find the information when a user is looking at the past workouts page.

When the login button is hit, the application will use the authentication and email of the user to see whether or not a player, or coach is the one logging in. If it is a player (Figure 8), then they will have the workout entry, team workout, and workout view pages. If it is a coach that just logged in (Figure 9), then they will have the team workout entry, and a team roster info page. Thanks to the user settings, the app is able to seamlessly divide which user will have access to which page.

home page of the app giving a small overview of the application.

When a user logs in, the unique I.D is retrieved from the database, then saved locally in the application. This will give the application the information it needs to store a workout and then later on, go and find the information when a user is looking at the past workouts page.

When the login button is

hit, the application will use the

authentication and email of the user to see whether or not a player, or coach is the one logging in. If it is a player (Figure 8), then they will have the workout entry, team workout, and workout view pages. If it is a coach that just logged in (Figure 9), then they will have the team workout entry, and a team roster info page. Thanks to the user settings, the app is able to seamlessly divide which user will have access to which page.

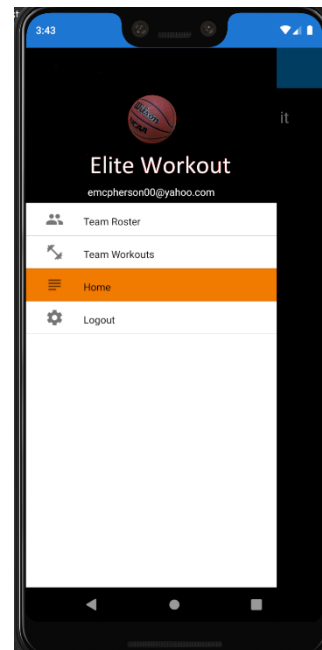


Figure 9: Coach's Side Dashboard

Within the pages, if a coach is going to have access to a certain feature and the players are not allowed to have the same access, then the user settings will also help to determine what is available on each page for each separate user.

Challenges

One of the biggest challenges faced in the creation of this application, was what kind of database was going to be used. The original idea was to store the information about individual workouts and user settings on the device in a local database file and having the information for it locally. This worked in the beginning of the development because it would not require a connection to the internet in order to store a workout that was going to be done. While testing the application, it was discovered that even if a user logged in and authenticated themselves, the information about personal workouts was being stored locally on the phone, making the information available to anyone using the application on another user's phone, giving access to the device owner's workouts. For this reason, the decision was made to store all the information in a database online and not on the phone. Using a cloud database essentially eliminated the possibility of a player having access to information that was not theirs.

Another challenge that showed itself was the creation of the different tables in the database. The original idea was to have one table for a team object but there was a hard time storing a list of names in the roster object on the team table. To work with this, a table was created for teams, coaches, and players. These tables are all connected using the Team Code as a key. This allows coaches

to have access to their team information and it allows players to have access to the same information, except they will also have access to see workouts that they have saved in the past.

Results

As of now the application is able to be deployed on both IOS and Android software. It is running smoothly at this time, and players are able to track their workouts and coaches are able to send workouts that they want players to complete..

At this time, the application is also compliant with CARA rules and should be able to be used during the summer because the workouts that are saved and tracked are for the player's use and purpose only. Coaches are not able to see a workout completed by a player unless they have specifically marked the workout as one that will be sent to the coach.

Future Work

Right now, the application is only operational for the sport of basketball. If the use of the application provides more ease for coaches and allows players to help develop their game and track their progress as a player, then a plan to deploy the application for use with other sports outside of basketball should be implemented. With the way the workout entry page is setup, it stores the description of a workout as a string, allowing a possibility for use with other sports teams. It also has the labels set as sport specific, so a coach should be able to use it for any sport team.

The next large update to be performed is the option for displaying a player's shooting percentage for the workouts that they have been doing. With the workout entry being stored as a string, it is harder to perform mathematical operations on the information. By changing the type of information that a workout will be stored as, with access to more information like shooting percentage and how many times a move is performed, then a player will be able to see the stats they need to work on to improve their skills both during the season and over the summer.

All in all, this is a very good start for an application that has the potential to change the way that a player sees their game, and it will help coaches to track the stats of their players as well, but only at times that the players want to send the workout (only relevant during the summer).

It would not even have to be used solely by college coaches but has the potential to be used by personal trainers, high school and junior high coaches, and potentially all the way up to the professional level. The possibilities are endless, and this app can help the development of all athletes at any level.

References

- Google, (2021, February 7). “Firebase Security Rules Documentation”
Basic Security Rules | Firebase (google.com).
- Balarju, Venkay, V.B (2018, May 17). “Learn about user settings in
Xamarin.Forms”. C Sharp Corner. Learn About User Settings in Xamarin.Forms
(c-sharpcorner.com).

Appendices

Appendix A – Functionality code:

Appendix A1 – CoachTeamInfo.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using WorkoutLogSP.ViewModels;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace WorkoutLogSP.Views
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class CoachTeamInfo : ContentPage
    {
        readonly FirebHelp firebaseHelper = new FirebHelp();

        public CoachTeamInfo()
        {
            InitializeComponent();
        }

        protected async override void OnAppearing()
        {
```

```

base.OnAppearing();

var role = "Player";

var sport = UserSettings.Sport;

var team = UserSettings.TeamName;

var players = await firebaseHelper.GetTeamRoster(role, sport, team);

Roster.ItemsSource = players;
}

async void OnPlayerSelection(Object sender, SelectedItemChangedEventArgs e)
{
    if (e.SelectedItem != null)
    {

        PlayWSet.ClearData();

        var tappedPlayer = e.SelectedItem.ToString();

        await firebaseHelper.GetPlayer(tappedPlayer);

        PlayWSet.Name = tappedPlayer;

        PlayWSet.Sport = UserSettings.Sport;
    }
}

```



```

        PlayWSet.SendWorkout = "true";

        await Navigation.PushAsync(new NavigationPage(new
CoachPlayerWorkouts()));
    }
}
}
}

```

Appendix A2 – CoachTeamRegistration.xaml.cs

```

using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using WorkoutLogSP.ViewModels;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace WorkoutLogSP.Views
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class CoachTeamRegistration : ContentPage
    {
        readonly FirebHelp firebaseHelper = new FirebHelp();
    }
}

```

```
public CoachTeamRegistration()
{
    InitializeComponent();
}

public async void TeamRegButton_Clicked(Object sender, EventArgs e)
{

    Random CoachIdGen = new Random();
    Random TeamCodeGen = new Random();

    int coachId = CoachIdGen.Next(1, 100000);

    int teamCode = TeamCodeGen.Next(1, 100000);

    string coach = Name.Text;

    string team = TName.Text;

    string role = "Coach";

    string email = UserSettings.Email;

    string sp = SportPick.SelectedItem.ToString();

    await firebaseHelper.AddCoach(role, email, coachId, coach, team, sp, teamCode);
```

```

        await firebaseHelper.AddTeam(TName.Text, teamCode, Name.Text, sp);

        Name.Text = string.Empty;
        TName.Text = string.Empty;

        await App.Current.MainPage.DisplayAlert("Team Code", "Your Team Code is " +
teamCode, "Next");

        await App.Current.MainPage.DisplayAlert("Success", "Coaches Team Registered",
"Continue");

        await Navigation.PushAsync(new LoginPage());

    }
}
}

```

Appendix A3 – CoachWorkoutEntry.xaml.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using WorkoutLogSP.ViewModels;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

```

```
namespace WorkoutLogSP.Views
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class CoachWorkoutEntry : ContentPage
    {

        readonly FirebHelp firebaseHelper = new FirebHelp();

        public CoachWorkoutEntry()
        {
            InitializeComponent();
        }

        async void OnSaveClicked(Object sender, EventArgs e)
        {

            string userComponent = UserSettings.TeamCode;

            string sport = UserSettings.Sport;

            string type = WorkType.SelectedItem.ToString();

            string description = WorkSum.Text;

            string timeCreated = DateTime.Now.ToString();
```

```

        await firebaseHelper.AddWorkout(userComponent, timeCreated, sport, type,
description);

        await Navigation.PopAsync();

    }
}
}

```

Appendix A4 – CoachPlayerWorkouts.xaml.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using WorkoutLogSP.ViewModels;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace WorkoutLogSP.Views
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class CoachPlayerWorkouts : ContentPage
    {
        readonly FirebHelp firebaseHelper = new FirebHelp();

        public CoachPlayerWorkouts()

```

```

    {
        InitializeComponent();
    }

    protected async override void OnAppearing()
    {
        base.OnAppearing();

        var playerWorkouts = await firebaseHelper.GetPlayerWorkouts(PlayWSet.Name,
        PlayWSet.Sport, PlayWSet.SendWorkout);

        PlayerWorkoutList.ItemsSource = playerWorkouts;
    }

    public async void OnItemSelected(Object Sender, SelectedItemChangedEventArgs)
    {
        {
            if(e.SelectedItem != null)
            {
                await Navigation.PushAsync(new WorkoutViewPageP
                {
                    BindingContext = e.SelectedItem as Workouts
                });
            }
        }
    }
}

```

Appendix A5 – Dashboard.xaml.cs

```
using Firebase.Auth;
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using WorkoutLogSP.ViewModels;
using Xamarin.Essentials;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace WorkoutLogSP.Views
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class Dashboard : MasterDetailPage
    {
        public string firebaseAPIKey = "AlzaSyB7J0wlqzW8sx-W_DFpqMOwbX5sEKqOXvI";

        public List<DashboardMenu> MenuItems { get; set; }

        public Dashboard()
        {
            InitializeComponent();
        }
    }
}
```

```

GetProfileInfoAndRefToken();

MenuItems = new List<DashboardMenu>();

var teamWorkout = new DashboardMenu() { Title = "Team Workouts", Icon =
"WorkoutPage.png", TargetType = typeof(TeamWorkoutList) };

var infoPage = new DashboardMenu() { Title = "Home", Icon =
"WorkoutEntries.png", TargetType = typeof(HomePage) };

var workoutListPage = new DashboardMenu() { Title = "Personal Workouts", Icon
= "workEnter.png", TargetType = typeof(WorkoutListPage) };

var logoutListItem = new DashboardMenu() { Title = "Logout", Icon =
"Logout.png", TargetType = typeof(LoginPage) };

var teamInfo = new DashboardMenu() { Title = "Team Roster", Icon =
"TeamPage.png", TargetType = typeof(CoachTeamInfo) };

if(UserSettings.Role == "Coach")
{
    MenuItems.Add(teamInfo);
}
else if(UserSettings.Role == "Player")
{
    MenuItems.Add(workoutListPage);
}

```



```

MenuItems.Add(teamWorkout);
MenuItems.Add(infoPage);
MenuItems.Add(logoutListItem);

navDrawer.ItemsSource = MenuItems;

Detail = new
NavigationPage((Page)Activator.CreateInstance(typeof(HomePage)));

this.BindingContext = new
{
    Header = "Workout Tracking Application",
    Image = "LoginImage.png",
    Footer = UserSettings.Email
};
}

async void OnPageSelected(Object sender, SelectedItemChangedEventArgs e)
{
    var item = (DashboardMenu)e.SelectedItem;

    Type page = item.TargetType;

    if (e.SelectedItem.ToString() == "Logout")
    {
        Preferences.Remove("FirebaseRefreshToken");
    }
}

```

```
UserSettings.ClearAllData();
```

```
Detail = new NavigationPage((Page)(new NavigationPage(new LoginPage())));
```

```
IsPresented = false;
```

```
}
```

```
else
```

```
{
```

```
Detail = new NavigationPage((Page)Activator.CreateInstance(page));
```

```
IsPresented = false;
```

```
}
```

```
}
```

```
async private void GetProfileInfoAndRefToken()
```

```
{
```

```
var authProv = new FirebaseAuthProvider(new FirebaseConfig(firebaseAPIKey));
```

```
try
```

```
{
```

```

        var savedAuth =
JsonConvert.DeserializeObject<Firebase.Auth.FirebaseAuth>(Preferences.Get("Firebase
RefreshToken", ""));

        var RefCont = await authProv.RefreshAuthAsync(savedAuth);

        Preferences.Set("FirebaseRefreshToken",
JsonConvert.SerializeObject(RefCont));
    }

    catch(Exception ex)
    {
        Console.WriteLine(ex.Message);

        await App.Current.MainPage.DisplayAlert("LogAlert", "Token has Expired",
"Ok");
    }

}

}

}

```

Appendix A6 – HomePage.xaml.cs

```

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

```

```
using System.Threading.Tasks;

using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace WorkoutLogSP.Views
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class HomePage : ContentPage
    {
        public HomePage()
        {
            InitializeComponent();
        }
    }
}
```

Appendix A7 – LoginPage.xaml.cs

```
using Firebase.Auth;
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.CompilerServices;
using System.Text;
using System.Threading.Tasks;
```

```

using WorkoutLogSP.ViewModels;

using Xamarin.Essentials;

using Xamarin.Forms;

using Xamarin.Forms.Xaml;

namespace WorkoutLogSP.Views
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class LoginPage : ContentPage
    {
        public LoginPage()
        {
            InitializeComponent();

            public string firebaseAPIKey = "AlzaSyB7J0wlqzW8sx-W_DFpqMOwbX5sEKqOXvI";

            readonly FirebHelp firebase = new FirebHelp();

            async void LoginClicked(Object sender, EventArgs e)
            {

                var authProv = new FirebaseAuthProvider(new FirebaseConfig(firebaseAPIKey));

                try
                {

```

```

        var authorize = await
authProv.SignInWithEmailAndPasswordAsync(EmailLoginEntry.Text,
PasswordLoginEntry.Text);

        var cont = await authorize.GetFreshAuthAsync();
        var serialCont = JsonConvert.SerializeObject(cont);
        Preferences.Set("FirebaseRefreshToken", serialCont);

        UserSettings.Email = EmailLoginEntry.Text;

        var user = await firebase.GetUser(EmailLoginEntry.Text);

        if (user != null)
        {
            UserSettings.Role = user.Role.ToString();
            UserSettings.ID = user.ID.ToString();
            UserSettings.Name = user.Name.ToString();
            UserSettings.TeamCode = user.TeamCode.ToString();
            UserSettings.TeamName = user.TeamName.ToString();
            UserSettings.Sport = user.Sport.ToString();
        }

        App.Current.MainPage = new Dashboard();

    }
    catch(Exception ex)
    {
        await App.Current.MainPage.DisplayAlert("Login Alert", ex.Message, "Ok");
    }

```

```

    }
}

async void RegisterClicked(Object sender, EventArgs e)
{
    await Navigation.PushAsync(new RegistrationPage());
}

}
}

```

Appendix A8 – PlayerRegistration.xaml.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using WorkoutLogSP.ViewModels;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace WorkoutLogSP.Views
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class PlayerRegistration : ContentPage
    {

```

```

readonly FirebHelp firebaseHelper = new FirebHelp();

public PlayerRegistration()
{
    InitializeComponent();
}

public async void PlayRegButton_Clicked(Object sender, EventArgs e)
{
    Random playerIdGen = new Random();

    int playerId = playerIdGen.Next(1, 100000);

    string player = pName.Text;

    string sport = SportPick.SelectedItem.ToString();

    string role = "Player";

    string email = UserSettings.Email;

    int teamCode = Convert.ToInt32(tCode.Text);

    string teamName = tName.Text;

    await firebaseHelper.AddPlayer(role, email, playerId, player, teamName, sport,
teamCode);
}

```



```

        pName.Text = string.Empty;
        tCode.Text = string.Empty;

        await App.Current.MainPage.DisplayAlert("Success", "Player Added to team",
"Continue");

        await Navigation.PushAsync(new LoginPage());
    }
}
}

```

Appendix A9 – RegistrationPage.xaml.cs

```

using Firebase.Auth;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using WorkoutLogSP.ViewModels;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace WorkoutLogSP.Views
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class RegistrationPage : ContentPage

```

```

{
    public string firebaseAPIKey = "AlzaSyB7J0wlqzW8sx-W_DFpqMOwbX5sEKqOXvI";

    public RegistrationPage()
    {
        InitializeComponent();
    }

    async void RegisterClicked(Object sender, EventArgs e)
    {
        try
        {
            if (CoachesBox.IsChecked)
            {
                var authProvider = new FirebaseAuthProvider(new
                FirebaseConfig(firebaseAPIKey));

                var auth = await
                authProvider.CreateUserWithEmailAndPasswordAsync(EmailEntry.Text,
                PasswordEntry.Text);

                string getToken = auth.FirebaseToken;

                await App.Current.MainPage.DisplayAlert("Success", "Registered
                Successfully", "OK");

                UserSettings.Email = EmailEntry.Text;

                await Navigation.PushAsync(new CoachTeamRegistration());
            }
        }
    }
}

```

```

else if(PlayersBox.IsChecked)
{
    var authProvider = new FirebaseAuthProvider(new
FirebaseConfig(firebaseAPIKey));

    var auth = await
authProvider.CreateUserWithEmailAndPasswordAsync(EmailEntry.Text,
PasswordEntry.Text);

    string getToken = auth.FirebaseToken;

    await App.Current.MainPage.DisplayAlert("Success", "Registered
Successfully", "OK");

    UserSettings.Email = EmailEntry.Text;

    await Navigation.PushAsync(new PlayerRegistration());
}

else
{
    await App.Current.MainPage.DisplayAlert("Alert", "Must Select Player or
Coach", "Ok");

    await Navigation.PushAsync(new RegistrationPage());
}

}
catch (Exception exc)
{
    await App.Current.MainPage.DisplayAlert("Alert", exc.Message, "OK");
}
}

```

```
    }  
  }  
}
```

Appendix A10 – TeamWorkoutList.xaml.cs

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
using WorkoutLogSP.ViewModels;  
using Xamarin.Forms;  
using Xamarin.Forms.Xaml;  
  
namespace WorkoutLogSP.Views  
{  
    [XamlCompilation(XamlCompilationOptions.Compile)]  
    public partial class TeamWorkoutList : ContentPage  
    {  
  
        readonly FirebHelp firebaseHelper = new FirebHelp();  
  
        public TeamWorkoutList()  
        {  
  
            InitializeComponent();  
  
            if(UserSettings.Role == "Coach")
```

```

    {
        addBtn.IsEnabled = true;
        addBtn.IsVisible = true;
    }
else
    {
        addBtn.IsEnabled = false;
        addBtn.IsVisible = false;
    }
}

protected async override void OnAppearing()
{
    base.OnAppearing();

    var workouts = await firebaseHelper.GetAllWorkouts(UserSettings.TeamCode);

    TeamWorkouts.ItemsSource = workouts;

}

async void AddWorkoutClicked(Object sender, EventArgs e)
{
    await Navigation.PushAsync(new CoachWorkoutEntry
    {
        BindingContext = new TeamWorkouts()
    }
    );
}

```

```

    });
}

async void OnItemSelected(Object sender, SelectedItemChangedEventArgs e)
{
    if (e.SelectedItem != null)
    {
        await Navigation.PushAsync(new WorkoutViewPage
        {
            BindingContext = e.SelectedItem as TeamWorkouts
        });
    }
}
}
}
}

```

Appendix A11 – WorkoutEntry.xaml.cs

```

using System;
using System.IO;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using Xamarin.Forms;

```

```

using Xamarin.Forms.Xaml;
using WorkoutLogSP.ViewModels;

namespace WorkoutLogSP.Views
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class WorkoutEntry : ContentPage
    {

        readonly FirebHelp firebaseHelper = new FirebHelp();

        public WorkoutEntry()
        {
            InitializeComponent();
        }

        async void OnSaveClicked(object sender, EventArgs e)
        {
            string userComponent = UserSettings.ID;

            string personCompleted = UserSettings.Name;

            string sport = UserSettings.Sport;

            string type = WorkType.SelectedItem.ToString();

            string description = WorkSum.Text;

```

```
string timeCreated = DateTime.Now.ToString();

string sendWorkout = "false";

if(sendToCoach.IsChecked)
{
    sendWorkout = "true";
}

await firebaseHelper.AddPersonalWorkout(userComponent, personCompleted,
timeCreated, sport, type, description, sendWorkout);

await Navigation.PopAsync();

}

}

}
```

Appendix A12 – WorkoutList.xaml.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```



```

using WorkoutLogSP.ViewModels;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace WorkoutLogSP.Views
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class WorkoutListPage : ContentPage
    {
        readonly FirebHelp firebaseHelper = new FirebHelp();

        public WorkoutListPage()
        {
            InitializeComponent();

            protected async override void OnAppearing()
            {
                base.OnAppearing();

                var workouts = await firebaseHelper.GetPersonalWorkouts(UserSettings.ID);

                WorkoutList.ItemsSource = workouts;
            }
        }
    }
}

```

```

    async void AddWorkoutClicked(Object sender, EventArgs e)
    {
        await Navigation.PushAsync(new WorkoutEntry
        {
            BindingContext = new Workouts()
        });
    }

    async void OnItemSelection(Object sender, SelectedItemChangedEventArgs e)
    {
        if(e.SelectedItem != null)
        {
            await Navigation.PushAsync(new WorkoutViewPageP
            {
                BindingContext = e.SelectedItem as Workouts
            });
        }
    }
}

```

Appendix A13 – WorkoutViewPage.xaml.cs

```

using System;
using System.Collections.Generic;
using System.Linq;

```

```

using System.Text;
using System.Threading.Tasks;
using WorkoutLogSP.ViewModels;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace WorkoutLogSP.Views
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class WorkoutViewPage : ContentPage
    {
        readonly FirebHelp firebseHelper = new FirebHelp();

        public WorkoutViewPage()
        {
            InitializeComponent();

            if(UserSettings.Role == "Coach")
            {
                Del.IsVisible = true;
                Del.IsEnabled = true;
            }
            else
            {
                Del.IsVisible = false;
                Del.IsEnabled = false;
            }
        }
    }
}

```

```

    }

    async void OnDeleteClicked(Object sender, EventArgs e)
    {
        string userComp = UserSettings.TeamCode;

        string time = timeCreated.Text.ToString();

        await firebaseHelper.DeleteWorkout(userComp, time);

        await Navigation.PopAsync();
    }
}
}
}

```

Appendix A14 – WorkoutViewPageP.xaml.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using WorkoutLogSP.ViewModels;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

```

```

namespace WorkoutLogSP.Views
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class WorkoutViewPageP : ContentPage
    {
        readonly FirebHelp firebseHelper = new FirebHelp();

        public WorkoutViewPageP()
        {
            InitializeComponent();
        }

        async void OnDeleteClicked(Object sender, EventArgs e)
        {
            string userComp = UserSettings.ID;

            string time = timeCreated.Text.ToString();

            await firebseHelper.DeletePersonalWorkout(userComp, time);

            await Navigation.PopAsync();
        }
    }
}

```

Appendix B – Styling code:

Appendix B1 – CoachTeamInfo.xaml

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

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="WorkoutLogSP.Views.CoachTeamInfo"
    BackgroundColor="Black">
    <ContentPage.Content>
        <StackLayout>
            <ListView x:Name="Roster" Margin="20" ItemSelected="OnPlayerSelection">
                <ListView.ItemTemplate>
                    <DataTemplate>
                        <TextCell x:Name="PlayerCell" Text="{Binding Name}"
                            TextColor="White"/>
                    </DataTemplate>
                </ListView.ItemTemplate>
            </ListView>
        </StackLayout>
    </ContentPage.Content>
</ContentPage>
```

Appendix B2 – CoachTeamRegistration.xaml

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

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="WorkoutLogSP.Views.CoachTeamRegistration"
    BackgroundColor="Black"
```

```

        NavigationPage.HasNavigationBar="False">
    <ContentPage.Content>
        <StackLayout Orientation="Vertical" HorizontalOptions="CenterAndExpand"
VerticalOptions="CenterAndExpand">
            <Label x:Name="Greetings" Text="Welcome Coach" TextColor="White"
FontSize="18"/>
            <Label x:Name="ThankYou" Text="Thank you for registering. Please enter the
information below." TextColor="White"/>
            <Label x:Name="CoachName" Text="Name:" TextColor="White"/>
            <Entry Placeholder="First Last" x:Name="Name" BackgroundColor="White"/>
            <Label x:Name="TeamName" Text="Team Name:" TextColor="White"/>
            <Entry Placeholder="Team Name" x:Name="TName"
BackgroundColor="White"/>
            <Label x:Name="SportType" Text="Select a sport: " TextColor="White"/>
            <Picker x:Name="SportPick" FontSize="Large" TextColor="Black"
BackgroundColor="White">
                <Picker.ItemsSource>
                    <x:Array Type="{x:Type x:String}">
                        <x:String>Men's Basketball</x:String>
                    </x:Array>
                </Picker.ItemsSource>
            </Picker>
            <Button x:Name="TeamRegistrationButton" Text="Enter" TextColor="Black"
BackgroundColor="White" WidthRequest="115" Clicked="TeamRegButton_Clicked"/>
        </StackLayout>
    </ContentPage.Content>
</ContentPage>

```

Appendix B3 – CoachWorkoutEntry.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
  x:Class="WorkoutLogSP.Views.CoachWorkoutEntry"
  BackgroundColor="Black"
  NavigationPage.HasNavigationBar="False">
  <ContentPage.Content>
    <StackLayout>
      <Label Text="Workout Entry Page:" FontAttributes="Bold" TextColor="White"
        FontSize="Large" VerticalOptions="Start" HorizontalOptions="CenterAndExpand"/>
      <Label FontAttributes="Bold" Text="Select Workout Type:" TextColor="White"/>
      <Picker x:Name="WorkType" FontSize="Large" TextColor="Black"
        BackgroundColor="White">
        <Picker.ItemsSource>
          <x:Array Type="{x:Type x:String}">
            <x:String>Sport Specific</x:String>
            <x:String>Strength Related</x:String>
            <x:String>Conditioning</x:String>
          </x:Array>
        </Picker.ItemsSource>
      </Picker>
      <Label FontAttributes="Bold" Text="Workout Description:" TextColor="White"/>
      <Editor x:Name="WorkSum" BackgroundColor="White" Placeholder="Brief
        description of Workout..." HeightRequest="300"/>
      <Button Text="Save" Clicked="OnSaveClicked" TextColor="Black"
        BackgroundColor="White"/>
    </StackLayout>
  </ContentPage.Content>
</ContentPage>
```



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

Appendix B4 – CoachPlayerWorkouts.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
  x:Class="WorkoutLogSP.Views.CoachPlayerWorkouts"
  BackgroundColor="Black"
  NavigationPage.HasNavigationBar="False"
  >
  <ContentPage.Content>
    <StackLayout>
      <ListView x:Name="PlayerWorkoutList" Margin="20"
        ItemSelected="OnItemSelected">
        <ListView.ItemTemplate>
          <DataTemplate>
            <TextCell Text="{Binding Description}" Detail="{Binding CurrTime}"
              TextColor="White" DetailColor="White"/>
          </DataTemplate>
        </ListView.ItemTemplate>
      </ListView>
    </StackLayout>
  </ContentPage.Content>
</ContentPage>
```

Appendix B5 – Dashboard.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<MasterDetailPage xmlns="http://xamarin.com/schemas/2014/forms"
  xmlns:local="clr-namespace:WorkoutLogSP.Views"
  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
  x:Class="WorkoutLogSP.Views.Dashboard"
  Title="Dashboard">

  <MasterDetailPage.Master>
    <ContentPage Title="Menu"
      BackgroundColor="Black">
      <StackLayout Orientation="Vertical">

        <Label Text="{Binding Header}"/>

        <Image Source="{Binding Image}" Aspect="AspectFit"/>

        <Label Text="{Binding Footer}" TextColor="White" FontSize="Small"
          HorizontalTextAlignment="Center"/>

        <ListView x:Name="navDrawer"
          RowHeight="45"
          SeparatorVisibility="Default"
          SeparatorColor="Black"
          BackgroundColor="White"
          ItemSelected="OnPageSelected">
```

```
<ListView.ItemTemplate>
  <DataTemplate>
    <ViewCell>

      <StackLayout VerticalOptions="FillAndExpand"
        Orientation="Horizontal"
        Padding="20,10,0,10"
        Spacing="20">

        <Image Source="{Binding Icon}"
          WidthRequest="40"
          HeightRequest="40"
          VerticalOptions="Start"/>

        <Label Text="{Binding Title}"
          FontSize="Small"
          VerticalOptions="End"
          TextColor="Black"/>

      </StackLayout>

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

</ListView>
```

</StackLayout>

</ContentPage>

</MasterDetailPage.Master>

</MasterDetailPage>

Appendix B6 – HomePage.xaml

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

```
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
  x:Class="WorkoutLogSP.Views.HomePage"
  NavigationPage.HasBackButton="False"
  Title="Home Page"
  BackgroundColor="Black">
```

```
<ContentPage.Content>
```

```
<StackLayout>
```

```
<Label Text="Welcome to the Elite workout app, it does the following: &#10;
&#10;* Track individual workouts&#10; &#10;* Send workouts to players and
coaches&#10; &#10;* Keep track of your roster &#10; " FontSize="Large"
TextColor="White" Margin="15,25,15,25"/>
```

```
</StackLayout>
```

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

```
</ContentPage>
```

Appendix B7 – LoginPage.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
  x:Class="WorkoutLogSP.Views.LoginPage"
  BackgroundColor="Black"
  NavigationPage.HasNavigationBar="False">
  <ContentPage.Content>
    <StackLayout Orientation="Vertical" Margin="10,0,10,0"
      VerticalOptions="CenterAndExpand">
      <Entry Placeholder="Email" x:Name="EmailLoginEntry"
        BackgroundColor="White" VerticalOptions="Center"/>
      <Entry Placeholder="Password" BackgroundColor="White" IsPassword="True"
        x:Name="PasswordLoginEntry" VerticalOptions="Center"/>
      <Button Text="Login" BackgroundColor="White" TextColor="Black"
        Clicked="LoginClicked" VerticalOptions="Center"/>
      <StackLayout Orientation="Vertical" VerticalOptions="End">
        <Label Text="Haven't Registered? Click below" TextColor="White"
          FontSize="Small" FontAttributes="Italic" VerticalOptions="End"/>
        <Button Text="Register" BackgroundColor="White" TextColor="Black"
          Clicked="RegisterClicked" VerticalOptions="End"/>
      </StackLayout>
    </StackLayout>
  </ContentPage.Content>
</ContentPage>
```

Appendix B8 – PlayerRegistration.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
  x:Class="WorkoutLogSP.Views.PlayerRegistration"
  BackgroundColor="Black"
  NavigationPage.HasNavigationBar="False">
  <ContentPage.Content>
    <StackLayout Orientation="Vertical" HorizontalOptions="FillAndExpand"
      VerticalOptions="CenterAndExpand">
      <Label x:Name="playerName" Text="Name: " TextColor="White"/>
      <Entry x:Name="pName" Placeholder="First Last" BackgroundColor="White"/>
      <Label x:Name="teamName" Text="Team Name: " TextColor="White"/>
      <Entry x:Name="tName" Placeholder="Team Name" BackgroundColor="White"/>
      <Label x:Name="teamCode" Text="Team Code: " TextColor="White"/>
      <Entry x:Name="tCode" Placeholder="Team Code" BackgroundColor="White"
        Keyboard="Numeric"/>
      <Label x:Name="sportType" Text="Select a sport: " TextColor="White"/>
      <Picker x:Name="SportPick" FontSize="Large" TextColor="Black"
        BackgroundColor="White">
        <Picker.ItemsSource>
          <x:Array Type="{x:Type x:String}">
            <x:String>Men's Basketball</x:String>
          </x:Array>
        </Picker.ItemsSource>
      </Picker>
      <Button x:Name="PlayerRegistrationButton" Text="Enter" TextColor="Black"
        BackgroundColor="White" WidthRequest="115" Clicked="PlayRegButton_Clicked"/>
    </StackLayout>
  </ContentPage.Content>
</ContentPage>
```

```

    </StackLayout>
  </ContentPage.Content>
</ContentPage>

```

Appendix B9 – RegistrationPage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
  x:Class="WorkoutLogSP.Views.RegistrationPage"
  BackgroundColor="Black"
  NavigationPage.HasNavigationBar="False">
  <ContentPage.Content>
    <StackLayout x:Name="Entries" Orientation="Vertical"
      VerticalOptions="CenterAndExpand" Margin="25,0,25,0">
      <Entry Placeholder="Email" BackgroundColor="White" x:Name="EmailEntry"/>
      <Entry Placeholder="Password" BackgroundColor="White"
        x:Name="PasswordEntry"/>
      <Label x:Name="PlayerCoachPrompt" Text="Are you a player or a coach: "
        TextColor="White"/>
      <Frame BorderColor="White">
        <StackLayout Orientation="Horizontal" VerticalOptions="CenterAndExpand">
          <Label x:Name="CoachesLabel" Text="Coach" BackgroundColor="White"
            HorizontalOptions="StartAndExpand"/>
          <CheckBox x:Name="CoachesBox" Color="Red" BackgroundColor="White"
            HorizontalOptions="StartAndExpand"/>
          <Label x:Name="PlayersLabel" Text="Player" BackgroundColor="White"
            HorizontalOptions="EndAndExpand"/>
          <CheckBox x:Name="PlayersBox" Color="Red" BackgroundColor="White"
            HorizontalOptions="EndAndExpand"/>

```

```

        </StackLayout>
    </Frame>
    <Button Text="Register" BackgroundColor="White" TextColor="Black"
WidthRequest="115" Clicked="RegisterClicked"/>
</StackLayout>
</ContentPage.Content>
</ContentPage>

```

Appendix B10 – TeamWorkoutList.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="WorkoutLogSP.Views.TeamWorkoutList"
    Title="Team Workouts"
    BackgroundColor="Black">
    <ContentPage.Content>
        <StackLayout>
            <ListView x:Name="TeamWorkouts" Margin="20"
ItemSelected="OnItemSelected">
                <ListView.ItemTemplate>
                    <DataTemplate>
                        <TextCell Text="{Binding Sport}" Detail="{Binding Type}"
TextColor="White" DetailColor="White"/>
                    </DataTemplate>
                </ListView.ItemTemplate>
            </ListView>
            <Button x:Name="addBtn" Text="Add" Clicked="AddWorkoutClicked"
BackgroundColor="White" TextColor="Black" VerticalOptions="End"/>

```



```
</StackLayout>
</ContentPage.Content>
</ContentPage>
```

Appendix B11 – WorkoutEntry.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
  x:Class="WorkoutLogSP.Views.WorkoutEntry"
  NavigationPage.HasBackButton="False"
  Title="Workout Entry Page"
  BackgroundColor="Black">
  <ContentPage.Content>
    <StackLayout>
      <StackLayout x:Name="Contents" VerticalOptions="Center">
        <Label VerticalOptions="Center" FontAttributes="Bold" Text="Enter Workout
Type:" TextColor="White"/>
        <Picker x:Name="WorkType" VerticalOptions="Center" FontSize="Large"
TextColor="Black" BackgroundColor="White">
          <Picker.ItemsSource>
            <x:Array Type="{x:Type x:String}">
              <x:String>Sport Specific</x:String>
              <x:String>Strength Related</x:String>
              <x:String>Conditioning</x:String>
            </x:Array>
          </Picker.ItemsSource>
        </Picker>
      </StackLayout>
    </ContentPage.Content>
  </ContentPage>
```

```

        <Label VerticalOptions="Center" FontAttributes="Bold" Text="Workout
Summary:" TextColor="White" />

        <Editor x:Name="WorkSum" VerticalOptions="Center"
BackgroundColor="White" Placeholder="Brief Summary of Workout..."
HeightRequest="300" />

        <CheckBox x:Name="sendToCoach" VerticalOptions="Center"
HorizontalOptions="Start" BackgroundColor="White"/>

    </StackLayout>

    <StackLayout x:Name="EndContents" VerticalOptions="End">

        <Button Text="Save" Clicked="OnSaveClicked" TextColor="Black"
BackgroundColor="White" VerticalOptions="End"/>

    </StackLayout>

</StackLayout>

</ContentPage.Content>

</ContentPage>

```

Appendix B12 – WorkoutList.xaml

```

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

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
x:Class="WorkoutLogSP.Views.WorkoutListPage"
Title="Personal Workouts"
BackgroundColor="Black">

    <StackLayout>

        <ListView x:Name="WorkoutList" Margin="20" ItemSelected="OnItemSelected">

            <ListView.ItemTemplate>

                <DataTemplate>

                    <TextCell Text="{Binding Description}" Detail="{Binding CurrTime}"
TextColor="White" DetailColor="White"/>

```

```

        </DataTemplate>
    </ListView.ItemTemplate>
</ListView>
    <Button Text="Add" Clicked="AddWorkoutClicked" BackgroundColor="White"
TextColor="Black" VerticalOptions="End"/>
</StackLayout>
</ContentPage>

```

Appendix B13 – WorkoutViewPage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="WorkoutLogSP.Views.WorkoutViewPage"
    NavigationPage.HasBackButton="True"
    BackgroundColor="Black">
    <ContentPage.Content>
        <StackLayout>
            <Label x:Name="spTop" Text="Sport: " FontSize="Large" FontAttributes="Bold"
TextColor="White" TextDecorations="Underline"/>
            <Label x:Name="Sport" Text="{Binding Sport}" TextColor="White"/>
            <Label x:Name="tyTop" Text="Type: " FontSize="Large" FontAttributes="Bold"
TextColor="White" TextDecorations="Underline"/>
            <Label x:Name="Type" Text="{Binding Type}" TextColor="White"/>
            <Label x:Name="deTop" Text="Description: " FontSize="Large"
FontAttributes="Bold" TextColor="White" TextDecorations="Underline"/>
            <Label x:Name="Desc" Text="{Binding Description}"
LineBreakMode="WordWrap" TextColor="White"/>
            <Label x:Name="timCr" Text="Created: " TextColor="White" FontSize="Medium"
FontAttributes="Bold" TextDecorations="Underline"/>

```

```

        <Label x:Name="timeCreated" Text="{Binding TimeCreated}" TextColor="Gray"
        FontSize="Small"/>

        <Button x:Name="Del" Text="Delete" BackgroundColor="White"
        TextColor="Black" WidthRequest="115" Clicked="OnDeleteClicked"
        VerticalOptions="End"/>

    </StackLayout>

</ContentPage.Content>

</ContentPage>

```

Appendix B14 – WorkoutViewPageP.xaml

```

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

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="WorkoutLogSP.Views.WorkoutViewPageP"
    BackgroundColor="Black"
    NavigationPage.HasBackButton="True">

    <ContentPage.Content>

        <StackLayout>

            <Label x:Name="spTop" Text="Sport: " FontSize="Large" FontAttributes="Bold"
            TextColor="White" TextDecorations="Underline"/>

            <Label x:Name="Sport" Text="{Binding Sport}" TextColor="White"/>

            <Label x:Name="tyTop" Text="Type: " FontSize="Large" FontAttributes="Bold"
            TextColor="White" TextDecorations="Underline"/>

            <Label x:Name="Type" Text="{Binding Type}" TextColor="White"/>

            <Label x:Name="deTop" Text="Description: " FontSize="Large"
            FontAttributes="Bold" TextColor="White" TextDecorations="Underline"/>

            <Label x:Name="Desc" Text="{Binding Description}"
            LineBreakMode="WordWrap" TextColor="White"/>

```

```

        <Label x:Name="timCr" Text="Created: " TextColor="White" FontSize="Medium"
FontAttributes="Bold" TextDecorations="Underline"/>

        <Label x:Name="timeCreated" Text="{Binding TimeCreated}" TextColor="Gray"
FontSize="Small"/>

        <Button x:Name="Del" Text="Delete" BackgroundColor="White"
TextColor="Black" WidthRequest="115" Clicked="OnDeleteClicked"
VerticalOptions="End"/>

    </StackLayout>

</ContentPage.Content>

</ContentPage>

```

Appendix C – Classes

Appendix C1 – DashboardMenu.cs

```

using System;

using System.Collections.Generic;

using System.Text;

namespace WorkoutLogSP.ViewModels
{
    public class DashboardMenu
    {
        public string Title { get; set; }

        public string Icon { get; set; }

        public Type TargetType { get; set; }
    }
}

```

Appendix C2 – FireBHelp.cs

```
using Firebase.Database;
using Firebase.Database.Query;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WorkoutLogSP.ViewModels
{
    class FirebHelp
    {

        //This function here sets the connection between the application and the firebase
        Database for authentication and a realtime DB

        readonly FirebaseClient firebase = new FirebaseClient("https://workoutlogsp-
        1c6b4.firebaseio.com/");

        //Function to return a list of all the users from the database

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

```

    {
        Role = item.Object.Role,
        Email = item.Object.Email,
        Name = item.Object.Name,
        ID = item.Object.ID,
        TeamName = item.Object.TeamName,
        Sport = item.Object.Sport,
        TeamCode = item.Object.TeamCode
    }).ToList();
}

//This function returns a list of all the teams on the app
public async Task<List<Team>> GetAllTeams()
{
    return (await firebase.Child("Teams").OnceAsync<Team>()).Select(item => new
Team
    {
        TeamName = item.Object.TeamName,
        TeamCode = item.Object.TeamCode,
        CoachName = item.Object.CoachName,
        Sport = item.Object.Sport
    }).ToList();
}

//Function to get all the workouts sent by a coach and from a certain team
public async Task<List<TeamWorkouts>> GetAllWorkouts(string us)
{

```

```

        return (await firebase.Child("Team
Workouts").OnceAsync<TeamWorkouts>()).Select(item => new TeamWorkouts
    {
        UserComp = item.Object.UserComp,
        TimeCreated = item.Object.TimeCreated,
        Type = item.Object.Type,
        Sport = item.Object.Sport,
        Description = item.Object.Description
    }).Where(x => x.UserComp == us).ToList();
    }

```

```

public async Task<List<Workouts>> GetPersonalWorkouts(string user)
{
    return (await firebase.Child("Personal
Workouts").OnceAsync<Workouts>()).Select(item => new Workouts
    {
        UserComp = item.Object.UserComp,
        WorkoutCompleter = item.Object.WorkoutCompleter,
        TimeCreated = item.Object.TimeCreated,
        Type = item.Object.Type,
        Sport = item.Object.Sport,
        Description = item.Object.Description,
        SendWorkout = item.Object.SendWorkout
    }).Where(x => x.UserComp == user).ToList();
}

```

//Function to get all the players under a team roster

```

public async Task<List<Users>> GetTeamRoster(string ro, string sp, string te)

```



```

    {
        return (await firebase.Child("Users").OnceAsync<Users>()).Select(item => new
Users
    {
        Role = item.Object.Role,
        Email = item.Object.Email,
        Name = item.Object.Name,
        ID = item.Object.ID,
        TeamName = item.Object.TeamName,
        Sport = item.Object.Sport,
        TeamCode = item.Object.TeamCode
    }).Where(x => x.Role == ro).Where(y => y.Sport == sp).Where(z => z.TeamName
== te).ToList();
    }

//Function to get the workouts specified by the player
public async Task<List<Workouts>> GetPlayerWorkouts(string playerName, string
sport, string sendWorkout)
{
    return (await firebase.Child("Personal
Workouts").OnceAsync<Workouts>()).Select(item => new Workouts
    {
        UserComp = item.Object.UserComp,
        WorkoutCompleter = item.Object.WorkoutCompleter,
        TimeCreated = item.Object.TimeCreated,
        Type = item.Object.Type,
        Sport = item.Object.Sport,
        Description = item.Object.Description,
        SendWorkout = item.Object.SendWorkout
    }

```

```
    }).Where(x => x.WorkoutCompleter == playerName).Where(x => x.Sport ==  
sport).Where(x => x.SendWorkout == sendWorkout).ToList();
```

```
    }
```

```
//Adds a new user to the table
```

```
public async Task AddPlayer(string role, string email, int id, string name, string  
teamName, string sport, int teamCode)
```

```
{
```

```
    await firebase.Child("Users").PostAsync(new Users()
```

```
    {
```

```
        Role = role,
```

```
        Email = email,
```

```
        ID = id,
```

```
        Name = name,
```

```
        TeamName = teamName,
```

```
        Sport = sport,
```

```
        TeamCode = teamCode
```

```
    });
```

```
}
```

```
public async Task AddCoach(string role, string email, int id, string name, string  
teamName, string sport, int teamCode)
```

```
{
```

```
    await firebase.Child("Users").PostAsync(new Users()
```

```
    {
```

```
        Role = role,
```

```
        Email = email,
```

```
        ID = id,
```

```
        Name = name,  
        TeamName = teamName,  
        Sport = sport,  
        TeamCode = teamCode  
    });  
}
```

```
//This function is to add a team to the database  
public async Task AddTeam(string tn, int tc, string cn, string sp)  
{  
    await firebase.Child("Teams").PostAsync(new Team()  
    {  
        TeamName = tn,  
        TeamCode = tc,  
        CoachName = cn,  
        Sport = sp  
    });  
}
```

```
public async Task AddWorkout(string tm, string tc, string sp, string ty, string des)  
{  
    await firebase.Child("Team Workouts").PostAsync(new TeamWorkouts()  
    {  
        UserComp = tm,  
        TimeCreated = tc,  
        Sport = sp,  
        Type = ty,  
        Description = des  
    });  
}
```

```

    Sport = sp,
    Type = ty,
    Description = des
  });
}

```

```

public async Task AddPersonalWorkout(string tm, string user, string tc, string sp,
string ty, string des, string send)

```

```

{
  await firebase.Child("Personal Workouts").PostAsync(new Workouts()
  {
    UserComp = tm,
    WorkoutCompleter = user,
    TimeCreated = tc,
    Sport = sp,
    Type = ty,
    Description = des,
    SendWorkout = send
  });
}

```

```

//Gets the specific user from the database

```

```

public async Task<Users> GetUser(string em)

```

```

{
  var allUsers = await GetAllUsers();
  await firebase.Child("Users").OnceAsync<Users>();
}

```

```

        return allUsers.Where(c => c.Email == em).FirstOrDefault();
    }

    //Tests for a user with a specific ID
    public async Task<Users> GetPlayer(String Name)
    {
        var allUserTest = await GetAllUsers();
        await firebase.Child("Users").OnceAsync<Users>();
        return allUserTest.Where(c => c.Name == Name).FirstOrDefault();
    }

    //Searches for a team using the Team Code
    public async Task<Team> GetTeam(int tc)
    {
        var allTeams = await GetAllTeams();
        await firebase.Child("Teams").OnceAsync<Team>();
        return allTeams.Where(x => x.TeamCode == tc).FirstOrDefault();
    }

    public async Task UpdateUser(int id, string email, string name, string team, string
role)
    {
        var toUpdateUser = (await firebase.Child("Users").OnceAsync<Users>()).Where(c
=> c.Object.ID == id).FirstOrDefault();

        await firebase.Child("Users").Child(toUpdateUser.Key).PutAsync(new Users()
    {

```

```

        Email = email,
        Name = name,
        TeamName = team,
        Role = role
    });
}

//Deletes a team from the system
public async Task DeleteTeam(string tnam)
{
    var delTeam = (await firebase.Child("Teams").OnceAsync<Team>()).Where(x =>
x.Object.TeamName == tnam).FirstOrDefault();

    await firebase.Child("Teams").Child(delTeam.Key).DeleteAsync();
}

//Deletes a User From the system
public async Task DeleteUser(string em)
{
    var delUser = (await firebase.Child("Users").OnceAsync<Users>()).Where(x =>
x.Object.Email == em).FirstOrDefault();

    await firebase.Child("Users").Child(delUser.Key).DeleteAsync();
}

//Deletes a workout that is being selected
public async Task DeleteWorkout(string userComp, string timeCreated)
{

```

```

        var delWorkout = (await firebase.Child("Team
Workouts").OnceAsync<TeamWorkouts>()).Where(w => w.Object.UserComp ==
userComp).Where(x => x.Object.TimeCreated == timeCreated).FirstOrDefault();

        await firebase.Child("Team Workouts").Child(delWorkout.Key).DeleteAsync();
    }

    public async Task DeletePersonalWorkout(string userComp, string timeCreated)
    {
        var delpWorkout = (await firebase.Child("Personal
Workouts").OnceAsync<Workouts>()).Where(w => w.Object.UserComp ==
userComp).Where(x => x.Object.TimeCreated == timeCreated).FirstOrDefault();

        await firebase.Child("Personal
Workouts").Child(delpWorkout.Key).DeleteAsync();
    }
}
}
}

```

Appendix C3 – Team.cs

```

using System;
using System.Collections.Generic;
using System.Text;

namespace WorkoutLogSP.ViewModels
{
    public class Team
    {

```

```
public string TeamName { get; set; }  
public int TeamCode { get; set; }  
public string CoachName { get; set; }  
public string Sport { get; set; }  
}  
}
```

Appendix C4 – TeamWorkouts.cs

```
using System;  
using System.Collections.Generic;  
using System.Text;  
  
namespace WorkoutLogSP.ViewModels  
{  
    public class TeamWorkouts  
    {  
        public string UserComp { get; set; }  
        public string TimeCreated { get; set; }  
        public string Sport { get; set; }  
        public string Type { get; set; }  
        public string Description { get; set; }  
    }  
}
```


Appendix C5 – UserInfo.cs

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

namespace WorkoutLogSP.ViewModels
{
    class UserInfo
    {
        public string Email
        {
            get => UserSettings.Email;
            set
            {
                UserSettings.Email = value;
            }
        }

        public string Name
        {
            get => UserSettings.Name;
            set
            {
                UserSettings.Name = value;
            }
        }
    }
}
```

```
public string ID
{
    get => UserSettings.ID;
    set
    {
        UserSettings.ID = value;
    }
}
```

```
public string Role
{
    get => UserSettings.Role;
    set => UserSettings.Role = value;
}
```

```
public string TeamName
{
    get => UserSettings.TeamName;
    set
    {
        UserSettings.TeamName = value;
    }
}
```

```
public string TeamCode
{
```

```
    get => UserSettings.TeamCode;
    set
    {
        UserSettings.TeamCode = value;
    }
}
}
```

Appendix C6 – Users.cs

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

namespace WorkoutLogSP.ViewModels
{
    public class Users
    {
        public string Role { get; set; }
        public string Email { get; set; }
        public int ID { get; set; }
        public string Name { get; set; }
        public string TeamName { get; set; }
        public string Sport { get; set; }
        public int TeamCode { get; set; }
    }
}
```

```
        public override string ToString() => Name;
    }

}
```

Appendix C7 – UserSettings.cs

```
using Plugin.Settings;
using Plugin.Settings.Abstractions;
using System;
using System.Collections.Generic;
using System.Text;

namespace WorkoutLogSP.ViewModels
{
    public static class UserSettings
    {
        static ISettings AppSettings
        {
            get {
                return CrossSettings.Current;
            }
        }

        public static string Email
        {
```

```
    get => AppSettings.GetValueOrDefault(nameof(Email), string.Empty);
    set => AppSettings.AddOrUpdateValue(nameof(Email), value);
}

public static string ID
{
    get => AppSettings.GetValueOrDefault(nameof(ID), string.Empty);
    set => AppSettings.AddOrUpdateValue(nameof(ID), value);
}

public static string Role
{
    get => AppSettings.GetValueOrDefault(nameof(Role), string.Empty);
    set => AppSettings.AddOrUpdateValue(nameof(Role), value);
}

public static string Name
{
    get => AppSettings.GetValueOrDefault(nameof(Name), string.Empty);
    set => AppSettings.AddOrUpdateValue(nameof(Name), value);
}

public static string Sport
{
    get => AppSettings.GetValueOrDefault(nameof(Sport), string.Empty);
    set => AppSettings.AddOrUpdateValue(nameof(Sport), value);
}
```

```

public static string TeamName
{
    get => AppSettings.GetValueOrDefault(nameof(TeamName), string.Empty);
    set => AppSettings.AddOrUpdateValue(nameof(TeamName), value);
}

public static string TeamCode
{
    get => AppSettings.GetValueOrDefault(nameof(TeamCode), string.Empty);
    set => AppSettings.AddOrUpdateValue(nameof(TeamCode), value);
}

public static void ClearAllData()
{
    AppSettings.Clear();
}
}
}

```

Appendix C8 – Workouts.cs

```

using System;
using System.Collections.Generic;
using System.Text;

namespace WorkoutLogSP.ViewModels

```

```
{  
  public class Workouts  
  {  
    public string UserComp { get; set; }  
    public string WorkoutCompleter { get; set; }  
    public string TimeCreated { get; set; }  
    public string Sport { get; set; }  
    public string Type { get; set; }  
    public string Description { get; set; }  
    public string SendWorkout { get; set; }  
  }  
}
```