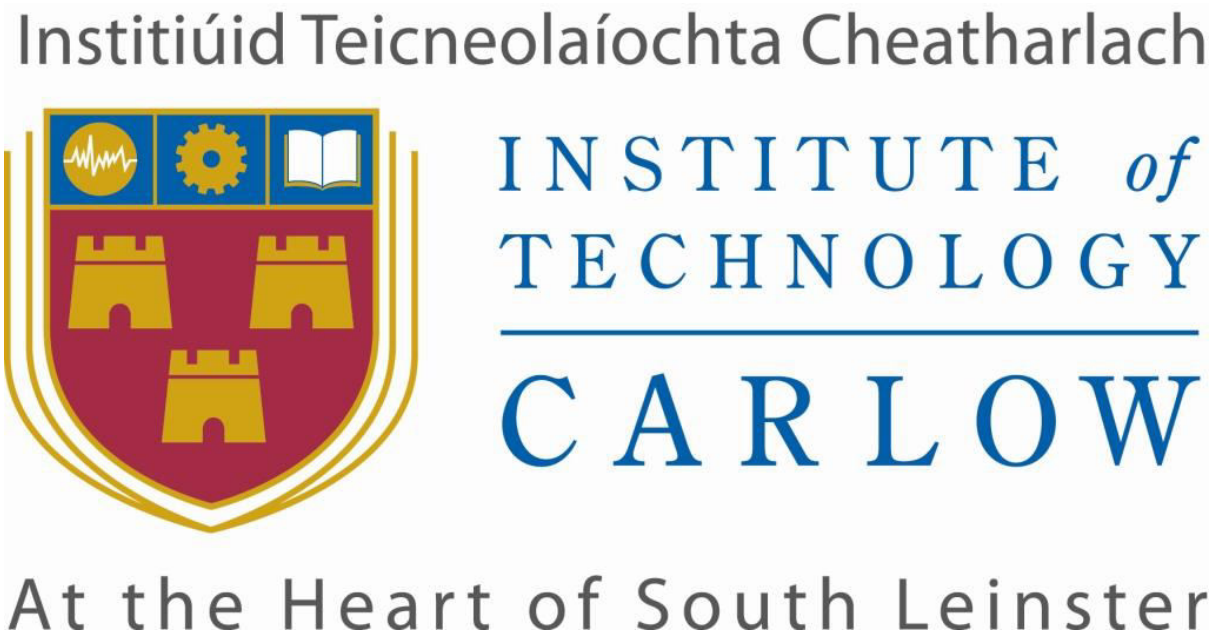
Pro Events

Technical Manual



Name: Jonathan Finlay

Student Number: C00193379

Course: Bachelor of Science (Honours) Software Development

Tutor: Hisain Elshaafi

Date: 18-04-18

Contents

Installation guide

Installing this app to your Android device is quite simple. With these few steps it will be step up in no time at all.

1. Some Android devices require you to turn on “Unknown Sources”, some may not. To check, go into settings, security and “Unknown Sources” is an option along the list, simply tick the box to turn it on. If you don’t see this option, then your phone doesn’t require you to turn it on.

2. A connection to internet is a must in order to download this application.

3. Please visit this link (insert link) and click in download apk.

4. The app will then download and open up. The application is now ready for use. Enjoy!

AttendingDB.cs

```
namespace ProEventsApp
{
    class AttendingDB
    {
        public string ID { get; set; }
        public string ACreator { get; set; }
        public string AUsername { get; set; }
        public string AEventID { get; set; }
        public string AEventName { get; set; }
        public string AUserEmail { get; set; }
        public string ALocation { get; set; }
        public string AAdress { get; set; }
        public string ADate { get; set; }
        public string ATime { get; set; }
        public string ACategory { get; set; }
        public string ADescription { get; set; }
    }
}
```

Connections.cs

```
namespace ProEventsApp
{
    [Activity(Label = "Connections")]
    public class Connections : Activity
    {
        public static MobileServiceClient Client =
        new MobileServiceClient("https://proevents.azurewebsites.net");

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

        string username = pref.GetString("Username", "User");

        private List<FollowingUser> tableFollow = new List<FollowingUser>();
        private ListView fListView;
        private List<FollowingUser> fItems;

        protected async override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);
            SetContentView(Resource.Layout.Connections_Page);

            fListView = FindViewById<ListView>(Resource.Id.myListViewConn);

            var searchBTN = FindViewById(Resource.Id.btnSeachUsers);
            var homeBTN = FindViewById(Resource.Id.btnHome);
            var eventsBTN = FindViewById(Resource.Id.btnEvents);
            var connectionsBTN = FindViewById(Resource.Id.btnConnections);
            var logout = FindViewById(Resource.Id.btnLogout);

            homeBTN.Click += HomeProcess;
            eventsBTN.Click += EventProcess;
            connectionsBTN.Click += ConnectionsProcess;
```

```
            logout.Click += LogoutProcess;
            searchBTN.Click += AllUsersSearch;

            CurrentPlatform.Init();
            IMobileServiceTable<FollowingUser> tableFollow =
Client.GetTable<FollowingUser>();
            fItems = await tableFollow
                .Where(x => x.Username == username)
                .ToListAsync();
            ListViewAdapterSearch adapterLV = new ListViewAdapterSearch(this, fItems);
            fListView.Adapter = adapterLV;

            fListView.ItemClick += FListView_ItemClick;
        }

        private void FListView_ItemClick(object sender, AdapterView.ItemClickEventArgs e)
        {
            var popup = new PopupMenu(this, fListView.GetChildAt(e.Position -
fListView.FirstVisiblePosition));
            CurrentPlatform.Init();
            string followActID = fItems[e.Position].ID;
            string followingID = fItems[e.Position].FuserID;
            string followingUserName = fItems[e.Position].FUsername;
            string followingFirst = fItems[e.Position].FFirstname;
            string followingSecond = fItems[e.Position].FLastname;
            string followingProf = fItems[e.Position].FProfession;
            string followingLocation = fItems[e.Position].FCounty;
            string followingDes = fItems[e.Position].FCounty;
            popup.Inflate(Resource.Layout.ConnMenu);
            popup.Show();
            popup.MenuItemClick += async (s, a) =>
            {
                switch (a.Item.ItemId)
                {
                    case Resource.Id.ViewProfile:
                        ISharedPreferences prefs =
Application.Context.GetSharedPreferences("UserInfo", FileCreationMode.Private);
                        ISharedPreferencesEditor edit = prefs.Edit();
                        string userprofile = fItems[e.Position].FUsername;
                        username = pref.GetString("Username", "User");
                        edit.PutString("Username", username);
                        edit.PutString("Userprofile", userprofile);
                        edit.Commit();
                        StartActivity(typeof(ViewProfile));
                        break;
                    case Resource.Id.Unfollow:
                        FollowingUser unfollowUser = new FollowingUser
                        {
                            ID = followActID,
                            Username = username,
                            FuserID = followingID,
                            FUsername = followingUserName,
                            FFirstname = followingFirst,
                            FLastname = followingSecond,
                            FProfession = followingProf,
                            FCounty = followingLocation,
                            FDescription = followingDes
                        };
                        await Client.GetTable<FollowingUser>().DeleteAsync(unfollowUser);
                        Toast.MakeText(this, "Success", ToastLength.Short).Show(); break;
                    default:
                        {
                            Toast.MakeText(this, "An error has occured, please try
again.", ToastLength.Short).Show();
```

```
                        }
                        break;
                }
            };
        }

            private void AllUsersSearch(object sender, EventArgs e)
        {
            ISharedPreferences prefs =
Application.Context.GetSharedPreferences("UserInfo", FileCreationMode.Private);
            ISharedPreferencesEditor edit = prefs.Edit();
            Toast.MakeText(this, "Search for an Event", ToastLength.Short).Show();
            StartActivity(typeof(ConnSearch));
        }
        private void HomeProcess(object sender, EventArgs e)
        {
            ISharedPreferences prefs =
Application.Context.GetSharedPreferences("UserInfo", FileCreationMode.Private);
            ISharedPreferencesEditor edit = prefs.Edit();
            Toast.MakeText(this, "Home Page", ToastLength.Short).Show();
            StartActivity(typeof(MainProfile));
        }
        private void EventProcess(object sender, EventArgs e)
        {
            ISharedPreferences prefs =
Application.Context.GetSharedPreferences("UserInfo", FileCreationMode.Private);
            ISharedPreferencesEditor edit = prefs.Edit();
            Toast.MakeText(this, "Events(soon)", ToastLength.Short).Show();
            StartActivity(typeof(Events));
        }
        private void ConnectionsProcess(object sender, EventArgs e)
        {
            ISharedPreferences prefs =
Application.Context.GetSharedPreferences("UserInfo", FileCreationMode.Private);
            ISharedPreferencesEditor edit = prefs.Edit();
            Toast.MakeText(this, "Your Connections", ToastLength.Short).Show();
            StartActivity(typeof(Connections));
        }
        private void LogoutProcess(object sender, EventArgs e)
        {
            ISharedPreferences prefs =
Application.Context.GetSharedPreferences("UserInfo", FileCreationMode.Private);
            ISharedPreferencesEditor edit = prefs.Edit();
            edit.Clear();
            edit.Commit();
            Toast.MakeText(this, "Logged Out", ToastLength.Short).Show();
            StartActivity(typeof(MainActivity));
        }
    }
}
```

ConnSearch.cs

```
namespace ProEventsApp
{
    [Activity(Label = "ConnSearch")]
    public class ConnSearch : Activity
    {
        public static MobileServiceClient Client =
        new MobileServiceClient("https://proevents.azurewebsites.net");
```

```
        private List<UserProf> sItems;
        private List<UserProf> sItemsSP = new List<UserProf>();
        private ListView sListView;

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

        string username = pref.GetString("Username", "User");

        protected async override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);

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

            SetContentView(Resource.Layout.Conn_Search);

            var homeBTN = FindViewById(Resource.Id.btnHome);
            var eventsBTN = FindViewById(Resource.Id.btnEvents);
            var connectionsBTN = FindViewById(Resource.Id.btnConnections);
            var logout = FindViewById(Resource.Id.btnLogout);
            var nearbyBTN = FindViewById(Resource.Id.btnNearby);

            homeBTN.Click += HomeProcess;
            eventsBTN.Click += EventProcess;
            connectionsBTN.Click += ConnectionsProcess;
            logout.Click += LogoutProcess;
            nearbyBTN.Click += NearbyProcess;

            sListView = FindViewById<ListView>(Resource.Id.myListViewSU);

            #region setUpDropdowns
            Spinner spinner = FindViewById<Spinner>(Resource.Id.spnUserCat);

            spinner.ItemSelected += new
EventHandler<AdapterView.ItemSelectedEventArgs>(spinner_ItemSelected);
            var adapter = ArrayAdapter.CreateFromResource(
                    this, Resource.Array.professions,
Android.Resource.Layout.SimpleSpinnerItem);

            adapter.SetDropDownViewResource(Android.Resource.Layout.SimpleSpinnerDropDownIt
em);
            spinner.Adapter = adapter;
            #endregion

            sListView.ItemClick += EListView_ItemClick;
        }

        private async void spinner_ItemSelected(object sender,
AdapterView.ItemSelectedEventArgs e)
        {
            Spinner spinner = (Spinner)sender;
            string spnCat = string.Format("{0}", spinner.GetItemAtPosition(e.Position));
            sItems = await Client.GetTable<UserProf>().ToListAsync();

            IMobileServiceTable<UserProf> table = Client.GetTable<UserProf>();
            sItems = await table
```

```
                .Where(x => x.Profession == spnCat)
                .ToListAsync();

            ListViewAdapterDisplayUser adapterLV = new ListViewAdapterDisplayUser(this,
sItems);
            sListView.Adapter = adapterLV;
        }

        private void EListView_ItemClick(object sender, AdapterView.ItemClickEventArgs e)
        {
            var popup = new PopupMenu(this, sListView.GetChildAt(e.Position -
sListView.FirstVisiblePosition));
            CurrentPlatform.Init();
            string followingID = sItems[e.Position].userID;
            string followingUserName = sItems[e.Position].Username;
            string followingFirst = sItems[e.Position].Firstname;
            string followingSecond = sItems[e.Position].Lastname;
            string followingProf = sItems[e.Position].Profession;
            string followingLocation = sItems[e.Position].County;
            string followingDes = sItems[e.Position].Description;
            popup.Inflate(Resource.Layout.WFE);
            popup.Show();
            popup.MenuItemClick += async (s, a) =>
            {
                switch (a.Item.ItemId)
                {
                    case Resource.Id.FollowWFE:
                        FollowingUser followUser = new FollowingUser
                        {
                            Username = username,
                            FuserID = followingID,
                            FUsername = followingUserName,
                            FFirstname = followingFirst,
                            FLastname = followingSecond,
                            FProfession = followingProf,
                            FCounty = followingLocation,
                            FDescription = followingDes
                        };
                        await Client.GetTable<FollowingUser>().InsertAsync(followUser);
                        Toast.MakeText(this, "Success", ToastLength.Short).Show();
                        break;
                    case Resource.Id.CancelWFE:
                        Toast.MakeText(this, "Cancelled", ToastLength.Short).Show();
                        break;
                    default:
                        {
                            Toast.MakeText(this, "An error has occured, please try again.",
ToastLength.Short).Show();
                        }
                        break;
                }
            };
        }

        private void NearbyProcess(object sender, EventArgs e)
        {
            edit.PutString("Username", "User");
            StartActivity(typeof(NearbyConnections));
        }

        private void HomeProcess(object sender, EventArgs e)
        {
            Toast.MakeText(this, "Home Page", ToastLength.Short).Show();
```

```
            edit.PutString("Username", "User");
            StartActivity(typeof(MainProfile));
        }
        private void EventProcess(object sender, EventArgs e)
        {
            Toast.MakeText(this, "Events(soon)", ToastLength.Short).Show();
            edit.PutString("Username", "User");
            StartActivity(typeof(Events));
        }
        private void ConnectionsProcess(object sender, EventArgs e)
        {
            Toast.MakeText(this, "Your Connections", ToastLength.Short).Show();
            edit.PutString("Username", "User");
            StartActivity(typeof(Connections));
        }

        private void LogoutProcess(object sender, EventArgs e)
        {
            edit.Clear();
            edit.Apply();
            Toast.MakeText(this, "Logged Out", ToastLength.Short).Show();
            StartActivity(typeof(MainActivity));
        }
    }
}
```

CreateEventForm.cs

```
namespace ProEventsApp
{
    [Activity(Label = "CreateEventForm")]
    public class CreateEventForm : Activity
    {
        public static MobileServiceClient Client =
        new MobileServiceClient("https://proevents.azurewebsites.net");

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

        string username = pref.GetString("Username", "User");

        protected override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);
            SetContentView(Resource.Layout.Create_Event_Form);

            Button dateBTN = FindViewById<Button>(Resource.Id.btnDate);
            dateBTN.Click += DateOnClick;
            Button timeBTN = FindViewById<Button>(Resource.Id.btnTime);
            timeBTN.Click += TimeOnClick;

            #region setUpDropdowns
            Spinner eventSelect = FindViewById<Spinner>(Resource.Id.spnEvents);
            var eventAdapter = ArrayAdapter.CreateFromResource(
            this, Resource.Array.professions, Android.Resource.Layout.SimpleSpinnerItem);

            eventAdapter.SetDropDownViewResource(Android.Resource.Layout.SimpleSpinnerDropD
ownItem);
```

```
            eventSelect.Adapter = eventAdapter;

            Spinner locationSelect = FindViewById<Spinner>(Resource.Id.spnLocation);
            var locationAdapter = ArrayAdapter.CreateFromResource(
            this, Resource.Array.county, Android.Resource.Layout.SimpleSpinnerItem);

            locationAdapter.SetDropDownViewResource(Android.Resource.Layout.SimpleSpinnerDr
opDownItem);
            locationSelect.Adapter = locationAdapter;
            #endregion

            var saveButton = FindViewById(Resource.Id.btnConfirm);
            saveButton.Click += ValidateForm;
        }
        private async void TrySave()
        {
            CurrentPlatform.Init();
            EditText eventName = (EditText)FindViewById(Resource.Id.txtEventName);
            Spinner location = (Spinner)FindViewById(Resource.Id.spnLocation);
            TextView address = (TextView)FindViewById(Resource.Id.txtEventAddress);
            TextView date = (TextView)FindViewById(Resource.Id.txtDate);
            Button dateBTN = FindViewById<Button>(Resource.Id.btnDate);
            Button timeBTN = FindViewById<Button>(Resource.Id.btnTime);
            TextView time = (TextView)FindViewById(Resource.Id.txtTime);
            Spinner category = (Spinner)FindViewById(Resource.Id.spnEvents);
            EditText description = (EditText)FindViewById(Resource.Id.txtEventDescripton);
            EventsDB newDBInfo = new EventsDB
            {
                CUsername = username,
                EventName = eventName.Text,
                Location = location.SelectedItem.ToString(),
                Address = address.Text,
                Time = time.Text,
                Date = date.Text,
                Category = category.SelectedItem.ToString(),
                Description = description.Text
            };
            await Client.GetTable<EventsDB>().InsertAsync(newDBInfo);
            List<EventsDB> newEvent = await Client.GetTable<EventsDB>()
                .Where(x => x.EventName == eventName.Text)
                .ToListAsync();
            AttendingDB attendingEvent = new AttendingDB
            {
                ACreator = username,
                AUsername = username,
                AEventName = eventName.Text,
                AEventID = newEvent[0].ID,
                ALocation = location.SelectedItem.ToString(),
                AAdress = address.Text,
                ADate = date.Text,
                ATime = time.Text,
                ACategory = category.SelectedItem.ToString(),
                ADescription = description.Text
            };
            await Client.GetTable<AttendingDB>().InsertAsync(attendingEvent);
        }

        private void DateOnClick(object sender, EventArgs e)
        {
            DatePickerFragment DPicker = DatePickerFragment.NewInstance(delegate (DateTime
time)
            {
                TextView datetxt = (TextView)FindViewById(Resource.Id.txtDate);
```

```
                DateTime eventDate;
                datetxt.Text = time.ToLongDateString();
                eventDate = time;
            });
            DPicker.Show(FragmentManager, DatePickerFragment.TAG);
        }
        private void TimeOnClick(object sender, EventArgs e)
        {
            TimePickerFragment TPicker = TimePickerFragment.NewInstance(delegate (DateTime
time)
                {
                    TextView timetxt = (TextView)FindViewById(Resource.Id.txtTime);
                    DateTime eventTime;
                    timetxt.Text = time.ToShortTimeString();
                    eventTime = time;
                });
            TPicker.Show(FragmentManager, TimePickerFragment.TAG);
        }

        private void ValidateForm(object sender, EventArgs e)
        {
            EditText eventName = (EditText)FindViewById(Resource.Id.txtEventName);
            Spinner location = (Spinner)FindViewById(Resource.Id.spnLocation);
            TextView datetxt = (TextView)FindViewById(Resource.Id.txtDate);
            Spinner category = (Spinner)FindViewById(Resource.Id.spnEvents);
            EditText description = (EditText)FindViewById(Resource.Id.txtEventDescripton);
            bool validInput = false;
            #region fieldValidation
            if (eventName.Text == "")
            {
                eventName.Error = "Cannot be Empty";
                eventName.RequestFocus();
            }
            else if (datetxt.Text == "")
            {
                datetxt.Error = "Cannot be Empty";
                datetxt.RequestFocus();
            }
            else if (description.Text == "")
            {
                description.Error = "Cannot be Empty";
                description.RequestFocus();
            }
            else
            {
                validInput = true;
            }
            #endregion
            if (validInput)
            {
                TrySave();
                Toast.MakeText(this, "Event Created Successfully",
ToastLength.Short).Show();
                StartActivity(typeof(Events));
            }
        }
    }

    public class DatePickerFragment : DialogFragment, DatePickerDialog.IOnDateSetListener
    {
        public static readonly string TAG = "X:" +
typeof(DatePickerFragment).Name.ToUpper();

        Action<DateTime> dateSelected = delegate { };
```

```
        public static DatePickerFragment NewInstance(Action<DateTime> date)
        {
            DatePickerFragment DPicker = new DatePickerFragment();
            DPicker.dateSelected = date;
            return DPicker;
        }

        public override Dialog OnCreateDialog(Bundle savedInstanceState)
        {
            DateTime currently = DateTime.Now;
            DatePickerDialog DPicker = new DatePickerDialog(Activity, this, currently.Year,
currently.Month - 1, currently.Day);
            return DPicker;
        }

        public void OnDateSet(DatePicker view, int year, int month, int day)
        {
            DateTime selectedDate = new DateTime(year, month + 1, day);
            Log.Debug(TAG, selectedDate.ToLongDateString());
            dateSelected(selectedDate);
        }
    }

    public class TimePickerFragment : DialogFragment, TimePickerDialog.IOnTimeSetListener
    {
        public static readonly string TAG = "MyTimePickerFragment";
        Action<DateTime> timeSelected = delegate { };

        public static TimePickerFragment NewInstance(Action<DateTime> time)
        {
            TimePickerFragment TPicker = new TimePickerFragment();
            TPicker.timeSelected = time;
            return TPicker;
        }

        public override Dialog OnCreateDialog(Bundle savedInstanceState)
        {
            DateTime currentTime = DateTime.Now;
            bool is24HourFormat = DateFormat.Is24HourFormat(Activity);
            TimePickerDialog TPicker = new TimePickerDialog(Activity, this,
currentTime.Hour, currentTime.Minute, is24HourFormat);
            return TPicker;
        }

        public void OnTimeSet(TimePicker view, int hour, int minute)
        {
            DateTime currentTime = DateTime.Now;
            DateTime selectedTime = new DateTime(currentTime.Year, currentTime.Month,
currentTime.Day, hour, minute, 0);
            Log.Debug(TAG, selectedTime.ToLongTimeString());
            timeSelected(selectedTime);
        }
    }
}
```

DBContext.cs

```
namespace ProEventsApp
```

```
{
    public static class DBContext
    {
        public static MobileServiceClient MobileService =
        new MobileServiceClient("https://proevents.azurewebsites.net");

        public static async void InsertNewUser(users u)
        {
            CurrentPlatform.Init();
            await MobileService.GetTable<users>().InsertAsync(u);

        }
        public static async void InsertNewUser(string userName, string password)
        {
            CurrentPlatform.Init();
            string hashedPassword = PasswordSaltHash.CreateHash(password);
            users u = new users { username = userName, password = hashedPassword };
            await MobileService.GetTable<users>().InsertAsync(u);
        }
        public static async Task<users> GetUser(string userName)
        {
            CurrentPlatform.Init();
            List<users> ls = await MobileService.GetTable<users>().ToListAsync();
            users u = ls.FirstOrDefault(x => x.username == userName);
            return u;
        }
        public static async Task<List<users>> GetAllUsers()
        {
            CurrentPlatform.Init();
            List<users> ls = await MobileService.GetTable<users>().ToListAsync();
            return ls;
        }
        public static async Task<bool> DoesUserExist(string userName)
        {
            CurrentPlatform.Init();
            List<users> ls = await MobileService.GetTable<users>().ToListAsync();
            users u = ls.FirstOrDefault(x => x.username == userName);
            if (u != null)
            {
                return true;
            }
            else
            {
                return false;
            }
        }
        public static void InsertUserProf(UserProf up)
        {
            CurrentPlatform.Init();
            MobileService.GetTable<UserProf>().InsertAsync(up);
        }
        public static void InsertUserProf(string username, string firstName, string
lastName, string profession, string county, string description)
        {
            CurrentPlatform.Init();
            UserProf up = new UserProf
            {
                Username = username,
                Firstname = firstName,
                Lastname = lastName,
                Profession = profession,
                County = county,
                Description = description
            };
```

```
            MobileService.GetTable<UserProf>().InsertAsync(up);
        }
        public static async Task<UserProf> GetUserProfile(string userid)
        {
            CurrentPlatform.Init();
            List<UserProf> ls = await MobileService.GetTable<UserProf>().ToListAsync();
            UserProf u = ls.FirstOrDefault(x => x.userID == userid);
            return u;
        }
        public static async Task<bool> IsUserProfileCreated(string userid)
        {
            CurrentPlatform.Init();
            List<UserProf> ls = await MobileService.GetTable<UserProf>().ToListAsync();
            UserProf u = ls.FirstOrDefault(x => x.userID == userid);
            if (u != null)
            {
                return true;
            }
            else
            {
                return false;
            }
        }
        public static async Task<UserProf> ProfileSetUp(string username)
        {
            CurrentPlatform.Init();
            List<UserProf> ls = await MobileService.GetTable<UserProf>().ToListAsync();
            UserProf prof = ls.FirstOrDefault(x => x.Username == username);
            return prof;
        }
    }
}
```

Dialog_SignIn.cs

```
namespace ProEventsApp
{
    [Activity(Label = "Dialog_SignIn")]
    public class Dialog_SignIn : Activity
    {

        public static MobileServiceClient Client =
        new MobileServiceClient("https://proevents.azurewebsites.net");

        protected override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);
            SetContentView(Resource.Layout.Dialog_Sign_In);


            EditText username = (EditText)FindViewById(Resource.Id.txtUsername);
            EditText password = (EditText)FindViewById(Resource.Id.txtPassword);

            var login = FindViewById(Resource.Id.btnDialogSignin);
            login.Click += Login;
        }

        private async void Login(object sender, EventArgs e)
        {
```

```
            ProgressDialog progressBar;
            progressBar = new Android.App.ProgressDialog(this);
            progressBar.Indeterminate = true;
            progressBar.SetProgressStyle(Android.App.ProgressDialogStyle.Spinner);
            progressBar.SetMessage("Logging In, Please Wait");
            progressBar.SetCancelable(false);
            progressBar.Show();
            EditText txtUsername = (EditText)FindViewById(Resource.Id.txtUsername);
            EditText txtPassword = (EditText)FindViewById(Resource.Id.txtPassword);
            string user = txtUsername.Text.Trim();
            string zPassword = txtPassword.Text;

            if (!await DBContext.DoesUserExist(user))
            {
                progressBar.Hide();
                Toast.MakeText(this, "Invalid Login", ToastLength.Short).Show();
            }
            else
            {
                users u = await DBContext.GetUser(user);
                if (PasswordSaltHash.VerifyPassword(zPassword, u.password))
                {
                    ISharedPreferences pref = GetSharedPreferences("UserInfo",
FileCreationMode.Private);
                    ISharedPreferencesEditor edit = pref.Edit();
                    edit.PutString("Username", u.username);
                    edit.PutString("Password", u.password);
                    edit.PutString("UserID", u.ID);
                    edit.PutString("LoggedIn", "true");
                    edit.Commit();
                    Toast.MakeText(ApplicationContext, "Success",
ToastLength.Short).Show();
                    if (await DBContext.IsUserProfileCreated(u.ID))
                    {
                        UserProf userprofile = await DBContext.GetUserProfile(u.ID);
                        edit.PutString("Username", userprofile.Username);
                        edit.PutString("UserID", userprofile.ID);
                        edit.Commit();
                        progressBar.Hide();
                        StartActivity(typeof(MainProfile));
                    }
                    else
                    {
                        progressBar.Hide();
                        StartActivity(typeof(SetUpProfile));
                    }
                }
                else
                {
                    progressBar.Hide();
                    Toast.MakeText(this, "Invalid Login", ToastLength.Short).Show();
                }
            }
        }
    }
}
```

Dialog_SignUp.cs

```
namespace ProEventsApp
{
    public class OnSignUpEventArgs : EventArgs
    {
        public int zID;
        public string zUsername;
        public string zPassword;
        public string zConfirmPassword;

        public OnSignUpEventArgs(string username, string password, string
confirmPassword) : base()
        {
            zUsername = username;
            zPassword = password;
            zConfirmPassword = confirmPassword;
        }
    }
    [Activity(Label = "Dialog_SignUp")]
    public class Dialog_SignUp : Activity
    {
        public static MobileServiceClient Client =
        new MobileServiceClient("https://proevents.azurewebsites.net");

        // public event EventHandler<OnSignUpEventArgs> ZOnSignUpComplete;
        protected override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);
            SetContentView(Resource.Layout.Dialog_Sign_Up);

            var zBtnSignUp = FindViewById<Button>(Resource.Id.btnDialogSignup);
            zBtnSignUp.Click += ZBtnSignUp_Click;
            EditText zTxtUsername = FindViewById<EditText>(Resource.Id.txtUsername);
            zTxtUsername.TextChanged += ValidateInput;
            EditText zTxtPassword = FindViewById<EditText>(Resource.Id.txtPassword);
            zTxtPassword.TextChanged += ValidateInput;
            EditText zTxtConfirmPassword =
FindViewById<EditText>(Resource.Id.txtConfirmPassword);
            zTxtConfirmPassword.TextChanged += ValidateInput;
        }

        private async void ZBtnSignUp_Click(object sender, EventArgs e)
        {
            EditText Username = FindViewById<EditText>(Resource.Id.txtUsername);
            EditText Password = FindViewById<EditText>(Resource.Id.txtPassword);
            EditText ConfirmPassword =
FindViewById<EditText>(Resource.Id.txtConfirmPassword);

            if (Password.Text == ConfirmPassword.Text)
            {
                string hashedPassword = PasswordSaltHash.CreateHash(Password.Text);
                string userName = Username.Text.Trim();
                users newU = new users { username = userName, password = hashedPassword
};
                List<users> allUsers = await Client.GetTable<users>().ToListAsync();
                users u = allUsers.FirstOrDefault(x => x.username == newU.username);
                if (u == null)
                {
                    DBContext.InsertNewUser(newU);
                    Toast.MakeText(this, "User " + newU.username + " created! You can now
log in!", ToastLength.Short).Show();
                    StartActivity(typeof(MainActivity));
                }
                else
```

```
                {
                    Toast.MakeText(this, "User " + u.username + " already exists!",
ToastLength.Short).Show();
                }
            }
            else
            {
                string message = "Passwords don't match.";
                Toast.MakeText(this, message, ToastLength.Short).Show();
            }
        }

        private void ValidateInput(object sender, TextChangedEventArgs e)
        {
            EditText input = (EditText)sender;
            string pattern = "[^a-zA-Z0-9#?]";
            if (Regex.IsMatch(input.Text, pattern))
            {
                string message = "Input must only contain alphabetic characters, numbers,
? and #";
                Toast.MakeText(this, message, ToastLength.Long).Show();
                input.Text = "";
            }
        }
    }
}
```

EmailContext.cs

```
namespace ProEventsApp
{
    public static class EmailContext
    {
        public static MobileServiceClient MobileService =
        new MobileServiceClient("https://proevents.azurewebsites.net");
        public const string SendGridAPIKey = "SG.NbTyRtBaT3ymmYsk-
Bj9yA.6Wm23coz0Kk4ZX0Pq5uNbuIJpfJfwKebfpT0s3tuf1Q";

        public static async System.Threading.Tasks.Task SendEmailNoticeAsync(string
emailTo, EventsDB Event)
        {
            var client = new SendGridClient(SendGridAPIKey);
            string emailMessage = string.Format("You have an Event upcoming! \nEvent
Name: " + Event.EventName +
                "\nLocation: " + Event.Location +
                "\nDate: " + Event.Date +
                " \nTime: " + Event.Time +
                "\nTo find out more, please visit the app!");
            var email = new SendGridMessage()
            {
                From = new EmailAddress("Do-Not_Reply@ProEvents.ie", "ProEvents"),
                Subject = "ProEvents Event Reminder",
                PlainTextContent = emailMessage,
                HtmlContent = "<strong>" + emailMessage.Replace("\n", "<br />") +
"</strong>"
            };
            email.AddTo(new EmailAddress(emailTo, "User"));
            var response = await client.SendEmailAsync(email);
            string mess = response.Body.ToString();
```

```
        }
    }
}
```

EventDetails.cs

```
namespace ProEventsApp
{
    [Activity(Label = "EventDetails")]
    public class EventDetails : Activity
    {
        public static MobileServiceClient Client =
        new MobileServiceClient("https://proevents.azurewebsites.net");

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

        string username = pref.GetString("Username", "User");
        string eventname = pref.GetString("EventName", "");
        AttendingDB prof;
        protected async override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);
            SetContentView(Resource.Layout.Event_Details);

            var homeBTN = FindViewById(Resource.Id.btnHome);
            var eventsBTN = FindViewById(Resource.Id.btnEvents);
            var connectionsBTN = FindViewById(Resource.Id.btnConnections);
            var logout = FindViewById(Resource.Id.btnLogout);
            var whoFollowingBTN = FindViewById(Resource.Id.btnWhoIsFollowing);

            CurrentPlatform.Init();
            List<AttendingDB> ls = await Client.GetTable<AttendingDB>().ToListAsync();
            prof = ls.FirstOrDefault(x => x.AEventName == eventname);
            TextView textName = (TextView)FindViewById(Resource.Id.txtViewEventName);
            textName.Text = "Event Name: " + prof.AEventName;
            textName.SetBackgroundResource(Resource.Drawable.round_style);
            TextView txtEventCreator =
(TextView)FindViewById(Resource.Id.txtViewEventCreator);
            txtEventCreator.Text = "Hosted by: " + prof.ACreator;
            txtEventCreator.SetBackgroundResource(Resource.Drawable.round_style);
            TextView txtAddress =
(TextView)FindViewById(Resource.Id.txtViewEventAddress);
            txtAddress.Text = "Held at: " + prof.AAdress;
            txtAddress.SetBackgroundResource(Resource.Drawable.round_style);
            TextView textCounty = (TextView)FindViewById(Resource.Id.txtViewEventCounty);
            textCounty.Text = "County: " + prof.ALocation;
            textCounty.SetBackgroundResource(Resource.Drawable.round_style);
            TextView textProfession =
(TextView)FindViewById(Resource.Id.txtViewEventProfession);
            textProfession.Text = "It is related to: " + prof.ACategory;
            textProfession.SetBackgroundResource(Resource.Drawable.round_style);
            TextView textDescription =
(TextView)FindViewById(Resource.Id.txtViewEventDescription);
            textDescription.Text = prof.ADescription;
            textDescription.SetBackgroundResource(Resource.Drawable.round_style);

            homeBTN.Click += HomeProcess;
```

```
            eventsBTN.Click += EventProcess;
            connectionsBTN.Click += ConnectionsProcess;
            logout.Click += LogoutProcess;
            whoFollowingBTN.Click += WhoFollowing;
        }

        private void WhoFollowing(object sender, EventArgs e)
        {
            edit.PutString("Username", username);
            edit.PutString("EventName", eventname);
            edit.Commit();
            StartActivity(typeof(WhoFollowEvent));
        }

        private void HomeProcess(object sender, EventArgs e)
        {
            Toast.MakeText(this, "Home Page", ToastLength.Short).Show();
            edit.PutString("Username", "User");
            StartActivity(typeof(MainProfile));
        }
        private void EventProcess(object sender, EventArgs e)
        {
            Toast.MakeText(this, "Events(soon)", ToastLength.Short).Show();
            edit.PutString("Username", "User");
            StartActivity(typeof(Events));
        }
        private void ConnectionsProcess(object sender, EventArgs e)
        {
            Toast.MakeText(this, "Your Connections", ToastLength.Short).Show();
            edit.PutString("Username", "User");
            StartActivity(typeof(Connections));
        }
        private void LogoutProcess(object sender, EventArgs e)
        {
            edit.Clear();
            edit.Apply();
            Toast.MakeText(this, "Logged Out", ToastLength.Short).Show();
            StartActivity(typeof(MainActivity));
        }
    }
}
```

Events.cs

```
namespace ProEventsApp
{
    [Activity(Label = "Events")]
    public class Events : Activity
    {
        public static MobileServiceClient Client =
        new MobileServiceClient("https://proevents.azurewebsites.net");

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

        string username = pref.GetString("Username", "User");

        private List<AttendingDB> table = new List<AttendingDB>();
```

```
        private ListView aListView;
        private List<AttendingDB> aItems;

        protected async override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);
            SetContentView(Resource.Layout.Events_Page);

            aListView = FindViewById<ListView>(Resource.Id.myListViewEP);

            var searchBTN = FindViewById(Resource.Id.searchEvents);
            var myEventsBTN = FindViewById(Resource.Id.myEvents);
            var beginCreate = FindViewById(Resource.Id.btnCreate);
            var homeBTN = FindViewById(Resource.Id.btnHome);
            var eventsBTN = FindViewById(Resource.Id.btnEvents);
            var connectionsBTN = FindViewById(Resource.Id.btnConnections);
            var logout = FindViewById(Resource.Id.btnLogout);

            homeBTN.Click += HomeProcess;
            eventsBTN.Click += EventProcess;
            connectionsBTN.Click += ConnectionsProcess;
            logout.Click += LogoutProcess;
            searchBTN.Click += AllEventsSearch;
            myEventsBTN.Click += MyEventsPage;
            beginCreate.Click += CreateEvent;

            CurrentPlatform.Init();
            IMobileServiceTable<AttendingDB> table = Client.GetTable<AttendingDB>();
            aItems = await table
                .Where(x => x.AUsername == username)
                .ToListAsync();
            ListViewAdapterAtt adapterLV = new ListViewAdapterAtt(this, aItems);
            aListView.Adapter = adapterLV;

            aListView.ItemClick += AListView_ItemClick;
        }

        private void AListView_ItemClick(object sender, AdapterView.ItemClickEventArgs e)
        {
            var popup = new PopupMenu(this, aListView.GetChildAt(e.Position -
aListView.FirstVisiblePosition));
            CurrentPlatform.Init();
            string AttendID = aItems[e.Position].ID;
            string AttendingEID = aItems[e.Position].AEventID;
            string AttendEventName = aItems[e.Position].AEventName;
            string AttendingEUserName = aItems[e.Position].AUsername;
            string AttendingEventCreator = aItems[e.Position].ACreator;
            string AttendingUserEmail = aItems[e.Position].AUserEmail;
            string AttendingECat = aItems[e.Position].ACategory;
            string FollowingELocation = aItems[e.Position].ALocation;
            string followingEDate = aItems[e.Position].ADate;
            string followingETime = aItems[e.Position].ATime;
            string followingEDes = aItems[e.Position].ADescription;
            popup.Inflate(Resource.Layout.ConnMenu);
            popup.Show();
            popup.MenuItemClick += async (s, a) =>
            {
                switch (a.Item.ItemId)
                {
                    case Resource.Id.ViewProfile:
                        ISharedPreferences prefs =
Application.Context.GetSharedPreferences("UserInfo", FileCreationMode.Private);
```

```
                        ISharedPreferencesEditor edit = prefs.Edit();
                        //string userprofile = aItems[e.Position].FUsername;
                        username = pref.GetString("Username", "User");
                        edit.PutString("Username", username);
                        edit.PutString("EventName", AttendEventName);
                        edit.Commit();
                        StartActivity(typeof(EventDetails));
                        break;
                    case Resource.Id.Unfollow:
                        AttendingDB unfollowEvent = new AttendingDB
                        {
                            ID = AttendID,
                            AEventName = AttendEventName,
                            AEventID = AttendingEID,
                            AUsername = username,
                            ACreator = AttendingEventCreator,
                            AUserEmail = AttendingEUserName,
                            ACategory = AttendingECat,
                            ALocation = followingEDes,
                            ADate = followingEDate,
                            ATime = followingETime,
                            ADescription = followingEDes
                        };
                        await Client.GetTable<AttendingDB>().DeleteAsync(unfollowEvent);
                        Toast.MakeText(this, "Success", ToastLength.Short).Show(); break;
                    default:
                        {
                            Toast.MakeText(this, "An error has occured, please try
again.", ToastLength.Short).Show();
                        }
                        break;
                }
            };
        }

        private void AllEventsSearch(object sender, EventArgs e)
        {
            ISharedPreferences prefs =
Application.Context.GetSharedPreferences("UserInfo", FileCreationMode.Private);
            ISharedPreferencesEditor edit = prefs.Edit();
            StartActivity(typeof(SearchEvent));
        }

        private void MyEventsPage(object sender, EventArgs e)
        {
            ISharedPreferences prefs =
Application.Context.GetSharedPreferences("UserInfo", FileCreationMode.Private);
            ISharedPreferencesEditor edit = prefs.Edit();
            edit.PutString("Username", username);
            StartActivity(typeof(MyEvents));
        }

        private void CreateEvent(object sender, EventArgs e)
        {
            ISharedPreferences prefs =
Application.Context.GetSharedPreferences("UserInfo", FileCreationMode.Private);
            ISharedPreferencesEditor edit = prefs.Edit();
            StartActivity(typeof(CreateEventForm));
        }

        private void HomeProcess(object sender, EventArgs e)
        {
            ISharedPreferences prefs =
```

```
Application.Context.GetSharedPreferences("UserInfo", FileCreationMode.Private);
            ISharedPreferencesEditor edit = prefs.Edit();
            StartActivity(typeof(MainProfile));
        }
        private void EventProcess(object sender, EventArgs e)
        {
            ISharedPreferences prefs =
Application.Context.GetSharedPreferences("UserInfo", FileCreationMode.Private);
            ISharedPreferencesEditor edit = prefs.Edit();
            StartActivity(typeof(Events));
        }
        private void ConnectionsProcess(object sender, EventArgs e)
        {
            ISharedPreferences prefs =
Application.Context.GetSharedPreferences("UserInfo", FileCreationMode.Private);
            ISharedPreferencesEditor edit = prefs.Edit();
            StartActivity(typeof(Connections));
        }
        private void LogoutProcess(object sender, EventArgs e)
        {
            ISharedPreferences prefs =
Application.Context.GetSharedPreferences("UserInfo", FileCreationMode.Private);
            ISharedPreferencesEditor edit = prefs.Edit();
            edit.Clear();
            edit.Commit();
            Toast.MakeText(this, "Logged Out", ToastLength.Short).Show();
            StartActivity(typeof(MainActivity));
        }
    }
}
```

EventsDB.cs

```
namespace ProEventsApp
{
    public class EventsDB
    {
        public string ID { get; set; }
        public string CUsername { get; set; }
        public string EventName { get; set; }
        public string Location { get; set; }
        public string Address { get; set; }
        public string Date { get; set; }
        public string Time { get; set; }
        public string Category { get; set; }
        public string Description { get; set; }
    }
}
```

FollowingUser.cs

```
namespace ProEventsApp
{
    public class FollowingUser
    {
```

```
        public string ID { get; set; }
        public string Username { get; set; }
        public string FuserID { get; set; }
        public string FUsername { get; set; }
        public string FFirstname { get; set; }
        public string FLastname { get; set; }
        public string FProfession { get; set; }
        public string FCounty { get; set; }
        public string FDescription { get; set; }
    }
}
```

ListViewAdapterSearch .cs

```
namespace ProEventsApp
{
    class ListViewAdapterSearch : BaseAdapter<FollowingUser>
    {
        public static MobileServiceClient Client =
        new MobileServiceClient("https://proevents.azurewebsites.net");

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

        string username = pref.GetString("Username", "User");

        private List<FollowingUser> fItems;
        private Context fContext;
        private int[] fChangeColour;
        private ConnSearch connSearch;
        private IMobileServiceTable<FollowingUser> tableFollow;

        public ListViewAdapterSearch(Context context, List<FollowingUser> items)
        {
            fItems = items;
            fContext = context;
            fChangeColour = new int[] { 0xF2F2F2, 0x00A0DC };
        }

        public ListViewAdapterSearch(ConnSearch connSearch,
IMobileServiceTable<FollowingUser> tableFollow)
        {
            this.connSearch = connSearch;
            this.tableFollow = tableFollow;
        }

        public override int Count
        {
            get { return fItems.Count; }
        }
        public override long GetItemId(int position)
        {
            return position;
        }
        public override FollowingUser this[int position]
        {
            get { return fItems[position]; }
        }
```

```
        public override View GetView(int position, View convertView, ViewGroup parent)
        {
            View row = convertView;

            if (row == null)
            {
                row =
LayoutInflater.From(fContext).Inflate(Resource.Layout.ListView_RowFollow, null, false);
            }

            row.SetBackgroundColor(GetColorFromInteger(fChangeColour[position %
fChangeColour.Length]));
            TextView txtUsername =
row.FindViewById<TextView>(Resource.Id.txtUserNameRowFol);
            txtUsername.Text = fItems[position].FUsername;
            TextView txtFName =
row.FindViewById<TextView>(Resource.Id.txtFullNameRowFol);
            txtFName.Text = fItems[position].FFirstname;
            txtFName.Text += " " + fItems[position].FLastname;
            TextView txtFPro = row.FindViewById<TextView>(Resource.Id.txtCatRowFol);
            txtFPro.Text = fItems[position].FProfession;
            TextView txtFLocation =
row.FindViewById<TextView>(Resource.Id.txtLocationRowFol);
            txtFLocation.Text = fItems[position].FCounty;
            TextView txtFDes =
row.FindViewById<TextView>(Resource.Id.txtDescriptionRowFol);
            txtFDes.Text = fItems[position].FDescription;

            if ((position % 2) == 1)
            {
                txtUsername.SetTextColor(Color.White);
                txtFName.SetTextColor(Color.White);
                txtFPro.SetTextColor(Color.White);
                txtFLocation.SetTextColor(Color.White);
                txtFDes.SetTextColor(Color.White);
            }
            else
            {
                txtUsername.SetTextColor(Color.Black);
                txtFName.SetTextColor(Color.Black);
                txtFPro.SetTextColor(Color.Black);
                txtFLocation.SetTextColor(Color.Black);
                txtFDes.SetTextColor(Color.Black);
            }

            return row;
        }

        private Color GetColorFromInteger(int color)
        {
            return Color.Rgb(Color.GetRedComponent(color),
Color.GetGreenComponent(color), Color.GetBlueComponent(color));
        }

    }
}
```

ListViewAdapter .cs

```
namespace ProEventsApp
{
    class ListViewAdapter : BaseAdapter<EventsDB>
    {
        public static MobileServiceClient Client =
        new MobileServiceClient("https://proevents.azurewebsites.net");

        private List<EventsDB> eItems;
        private Context eContext;
        private int[] eChangeColour;

        public ListViewAdapter(Context context, List<EventsDB> items)
        {
            eItems = items;
            eContext = context;
            eChangeColour = new int[] { 0xF2F2F2, 0x00A0DC };
        }
        public override int Count
        {
            get { return eItems.Count; }
        }
        public override long GetItemId(int position)
        {
            return position;
        }
        public override EventsDB this[int position]
        {
            get { return eItems[position]; }
        }
        public override View GetView(int position, View convertView, ViewGroup parent)
        {
            View row = convertView;

            if (row == null)
            {
                row = LayoutInflater.From(eContext).Inflate(Resource.Layout.ListView_Row,
null, false);
            }

            row.SetBackgroundColor(GetColorFromInteger(eChangeColour[position %
eChangeColour.Length]));

            TextView txtEventName =
row.FindViewById<TextView>(Resource.Id.txtEventNameRow);
            txtEventName.Text = eItems[position].EventName;
            TextView txtEventCreator =
row.FindViewById<TextView>(Resource.Id.txtEventCreatorRow);
            txtEventCreator.Text = eItems[position].CUsername;
            TextView txtEventLocation =
row.FindViewById<TextView>(Resource.Id.txtEventLocationRow);
            txtEventLocation.Text = eItems[position].Location;
            TextView txtEventDate =
row.FindViewById<TextView>(Resource.Id.txtEventDateRow);
            txtEventDate.Text = eItems[position].Date;
            TextView txtEventTime =
row.FindViewById<TextView>(Resource.Id.txtEventTimeRow);
            txtEventTime.Text = eItems[position].Time;
            TextView txtEventCategory =
row.FindViewById<TextView>(Resource.Id.txtEventCategoryRow);
            txtEventCategory.Text = eItems[position].Category;
            TextView txtEventDescription =
row.FindViewById<TextView>(Resource.Id.txtEventDescriptionRow);
            txtEventDescription.Text = eItems[position].Description;
```

```
            if ((position % 2) == 1)
            {
                txtEventName.SetTextColor(Color.White);
                txtEventCreator.SetTextColor(Color.White);
                txtEventLocation.SetTextColor(Color.White);
                txtEventDate.SetTextColor(Color.White);
                txtEventTime.SetTextColor(Color.White);
                txtEventCategory.SetTextColor(Color.White);
                txtEventDescription.SetTextColor(Color.White);
            }
            else
            {
                txtEventName.SetTextColor(Color.Black);
                txtEventCreator.SetTextColor(Color.Black);
                txtEventLocation.SetTextColor(Color.Black);
                txtEventDate.SetTextColor(Color.Black);
                txtEventTime.SetTextColor(Color.Black);
                txtEventCategory.SetTextColor(Color.Black);
                txtEventDescription.SetTextColor(Color.Black);
            }

            return row;
        }

        private Color GetColorFromInteger(int color)
        {
            return Color.Rgb(Color.GetRedComponent(color),
Color.GetGreenComponent(color), Color.GetBlueComponent(color));
        }
    }
}
```

ListViewAdapterAtt .cs

```
namespace ProEventsApp
{
    class ListViewAdapterAtt : BaseAdapter<AttendingDB>
    {
        public static MobileServiceClient Client =
        new MobileServiceClient("https://proevents.azurewebsites.net");

        private List<AttendingDB> aItems;
        private Context aContext;
        private int[] aChangeColour;
        private Events events;
        private IMobileServiceTable<AttendingDB> table;

        public ListViewAdapterAtt(Context context, List<AttendingDB> items)
        {
            aItems = items;
            aContext = context;
            aChangeColour = new int[] { 0xF2F2F2, 0x00A0DC };
        }

        public ListViewAdapterAtt(Events events, IMobileServiceTable<AttendingDB> table)
        {
            this.events = events;
            this.table = table;
```

```
        }

        public override int Count
        {
            get { return aItems.Count; }
        }
        public override long GetItemId(int position)
        {
            return position;
        }
        public override AttendingDB this[int position]
        {
            get { return aItems[position]; }
        }
        public override View GetView(int position, View convertView, ViewGroup parent)
        {
            View row = convertView;

            if (row == null)
            {
                row = LayoutInflater.From(aContext).Inflate(Resource.Layout.ListView_Row,
null, false);
            }

            row.SetBackgroundColor(GetColorFromInteger(aChangeColour[position %
aChangeColour.Length]));

            TextView txtEventName =
row.FindViewById<TextView>(Resource.Id.txtEventNameRow);
            txtEventName.Text = "Event Name: " + aItems[position].AEventName;
            TextView txtEventCreator =
row.FindViewById<TextView>(Resource.Id.txtEventCreatorRow);
            txtEventCreator.Text = "Event Creator: "  +
aItems[position].ACreator;
            TextView txtEventLocation =
row.FindViewById<TextView>(Resource.Id.txtEventLocationRow);
            txtEventLocation.Text = "Location: " + aItems[position].ALocation;
            TextView txtEventDate =
row.FindViewById<TextView>(Resource.Id.txtEventDateRow);
            txtEventDate.Text = "Date: " + aItems[position].ADate;
            TextView txtEventTime =
row.FindViewById<TextView>(Resource.Id.txtEventTimeRow);
            txtEventTime.Text = "Time: " + aItems[position].ATime;
            TextView txtEventCategory =
row.FindViewById<TextView>(Resource.Id.txtEventCategoryRow);
            txtEventCategory.Text = "Category: " + aItems[position].ACategory;
            TextView txtEventDescription =
row.FindViewById<TextView>(Resource.Id.txtEventDescriptionRow);
            txtEventDescription.Text = "Addational Details: " +
aItems[position].ADescription;

            if ((position % 2) == 1)
            {
                txtEventName.SetTextColor(Color.White);
                txtEventCreator.SetTextColor(Color.White);
                txtEventLocation.SetTextColor(Color.White);
                txtEventDate.SetTextColor(Color.White);
                txtEventTime.SetTextColor(Color.White);
                txtEventCategory.SetTextColor(Color.White);
                txtEventDescription.SetTextColor(Color.White);
            }
            else
            {
```

```
                txtEventName.SetTextColor(Color.Black);
                txtEventCreator.SetTextColor(Color.Black);
                txtEventLocation.SetTextColor(Color.Black);
                txtEventDate.SetTextColor(Color.Black);
                txtEventTime.SetTextColor(Color.Black);
                txtEventCategory.SetTextColor(Color.Black);
                txtEventDescription.SetTextColor(Color.Black);
            }

            return row;
        }

        private Color GetColorFromInteger(int color)
        {
            return Color.Rgb(Color.GetRedComponent(color),
Color.GetGreenComponent(color), Color.GetBlueComponent(color));
        }

    }
}
```

ListViewAdapterDisplayUser .cs

```
namespace ProEventsApp
{
    class ListViewAdapterDisplayUser : BaseAdapter<UserProf>
    {
        public static MobileServiceClient Client =
        new MobileServiceClient("https://proevents.azurewebsites.net");

        private List<UserProf> sItems;
        private Context sContext;
        private int[] sChangeColour;
        private ConnSearch connSearch;
        private IMobileServiceTable<UserProf> tableSearch;

        public ListViewAdapterDisplayUser(Context context, List<UserProf> items)
        {
            sItems = items;
            sContext = context;
            sChangeColour = new int[] { 0xF2F2F2, 0x00A0DC };
        }

        public ListViewAdapterDisplayUser(ConnSearch connSearch,
IMobileServiceTable<UserProf> tableSearch)
        {
            this.connSearch = connSearch;
            this.tableSearch = tableSearch;
        }

        public override int Count
        {
            get { return sItems.Count; } // crash if null
        }
        public override long GetItemId(int position)
        {
            return position;
```

```
        }
        public override UserProf this[int position]
        {
            get { return sItems[position]; }
        }
        public override View GetView(int position, View convertView, ViewGroup parent)
        {
            View row = convertView;

            if (row == null)
            {
                row =
LayoutInflater.From(sContext).Inflate(Resource.Layout.ListView_RowUser, null, false);
            }

            row.SetBackgroundColor(GetColorFromInteger(sChangeColour[position %
sChangeColour.Length]));

            TextView txtUserName =
row.FindViewById<TextView>(Resource.Id.txtUserNameRow);
            txtUserName.Text = sItems[position].Firstname;
            txtUserName.Text += " " + sItems[position].Lastname;
            TextView txtUserPro =
row.FindViewById<TextView>(Resource.Id.txtUserCategoryRow);
            txtUserPro.Text = sItems[position].Profession;
            TextView txtUserLocation =
row.FindViewById<TextView>(Resource.Id.txtUserLocationRow);
            txtUserLocation.Text = sItems[position].County;
            TextView txtUserDes =
row.FindViewById<TextView>(Resource.Id.txtUserDescriptionRow);
            txtUserDes.Text = sItems[position].Description;

            if ((position % 2) == 1)
            {
                txtUserName.SetTextColor(Color.White);
                txtUserPro.SetTextColor(Color.White);
                txtUserLocation.SetTextColor(Color.White);
                txtUserDes.SetTextColor(Color.White);
            }
            else
            {
                txtUserName.SetTextColor(Color.Black);
                txtUserPro.SetTextColor(Color.Black);
                txtUserLocation.SetTextColor(Color.Black);
                txtUserDes.SetTextColor(Color.Black);
            }

            return row;
        }

        private Color GetColorFromInteger(int color)
        {
            return Color.Rgb(Color.GetRedComponent(color),
Color.GetGreenComponent(color), Color.GetBlueComponent(color));
        }

    }
}
```

ListViewAdapterWhoFollowEvent .cs

```
namespace ProEventsApp
{
    class ListViewAdapterWhoFollowEvent : BaseAdapter<AttendingDB>
    {
        public static MobileServiceClient Client =
        new MobileServiceClient("https://proevents.azurewebsites.net");

        private List<AttendingDB> WFEItems;
        private Context WFEContext;
        private int[] WFEChangeColour;
        private WhoFollowEvent whoFollowEvent;
        private IMobileServiceTable<AttendingDB> tableWFE;

        public ListViewAdapterWhoFollowEvent(Context context, List<AttendingDB> items)
        {
            WFEItems = items;
            WFEContext = context;
            WFEChangeColour = new int[] { 0xF2F2F2, 0x00A0DC };
        }

        public ListViewAdapterWhoFollowEvent(WhoFollowEvent whoFollowEvent,
IMobileServiceTable<AttendingDB> tableWFE)
        {
            this.whoFollowEvent = whoFollowEvent;
            this.tableWFE = tableWFE;
        }

        public override int Count
        {
            get { return WFEItems.Count; }
        }
        public override long GetItemId(int position)
        {
            return position;
        }
        public override AttendingDB this[int position]
        {
            get { return WFEItems[position]; }
        }
        public override View GetView(int position, View convertView, ViewGroup parent)
        {
            View row = convertView;

            if (row == null)
            {
                row =
LayoutInflater.From(WFEContext).Inflate(Resource.Layout.ListView_WhoFollowEvent, null,
false);
            }

            row.SetBackgroundColor(GetColorFromInteger(WFEChangeColour[position %
WFEChangeColour.Length]));

            TextView txtUserNameWhoEvent =
row.FindViewById<TextView>(Resource.Id.txtUserNameWhoFollowEvent);
            txtUserNameWhoEvent.Text = WFEItems[position].AUsername;
            TextView txtNameWhoEvent =
row.FindViewById<TextView>(Resource.Id.txtNameWhoFollowEvent);
            txtNameWhoEvent.Text = WFEItems[position].AUsername;
```

```
            TextView txtEmailWhoEvent =
row.FindViewById<TextView>(Resource.Id.txtEmailWhoFollowEvent);
            txtEmailWhoEvent.Text = WFEItems[position].AUsername;
            TextView txtLocationWhoEvent =
row.FindViewById<TextView>(Resource.Id.txtLocationWhoFollowEvent);
            txtLocationWhoEvent.Text = WFEItems[position].AUsername;
            TextView txtCategoryWhoEvent =
row.FindViewById<TextView>(Resource.Id.txtCategoryWhoFollowEvent);
            txtCategoryWhoEvent.Text = WFEItems[position].AUsername;

            if ((position % 2) == 1)
            {
                txtUserNameWhoEvent.SetTextColor(Color.White);
            }
            else
            {
                txtUserNameWhoEvent.SetTextColor(Color.Black);
            }

            return row;
        }

        private Color GetColorFromInteger(int color)
        {
            return Color.Rgb(Color.GetRedComponent(color),
Color.GetGreenComponent(color), Color.GetBlueComponent(color));
        }

    }
}
```

LVMyEvents .cs

```
namespace ProEventsApp
{
    class LVMyEvents : BaseAdapter<EventsDB>
    {
        public static MobileServiceClient Client =
        new MobileServiceClient("https://proevents.azurewebsites.net");

        private List<EventsDB> mItems;
        private Context mContext;
        private int[]  mChangeColour;
        private MyEvents myevents;
        private IMobileServiceTable<EventsDB> mytable;

        public LVMyEvents(Context context, List<EventsDB> items)
        {
            mItems = items;
            mContext = context;
            mChangeColour = new int[] { 0xF2F2F2, 0x00A0DC };
        }

        public LVMyEvents(MyEvents myevents, IMobileServiceTable<EventsDB> mytable)
        {
            this.myevents = myevents;
            this.mytable = mytable;
```

```
        }

        public override int Count
        {
            get { return mItems.Count; }
        }
        public override long GetItemId(int position)
        {
            return position;
        }
        public override EventsDB this[int position]
        {
            get { return mItems[position]; }
        }
        public override View GetView(int position, View convertView, ViewGroup parent)
        {
            View row = convertView;

            if (row == null)
            {
                row = LayoutInflater.From(mContext).Inflate(Resource.Layout.ListView_Row,
null, false);
            }

            row.SetBackgroundColor(GetColorFromInteger(mChangeColour[position %
mChangeColour.Length]));

            TextView txtMyEventName =
row.FindViewById<TextView>(Resource.Id.txtEventNameRow);
            txtMyEventName.Text = "Event Name: " + mItems[position].EventName;
            TextView txtMyEventCreator =
row.FindViewById<TextView>(Resource.Id.txtEventCreatorRow);
            txtMyEventCreator.Text = "Event Creator: " + mItems[position].CUsername;
            TextView txtMyEventLocation =
row.FindViewById<TextView>(Resource.Id.txtEventLocationRow);
            txtMyEventLocation.Text = "Location: " + mItems[position].Location;
            TextView txtMyEventDate =
row.FindViewById<TextView>(Resource.Id.txtEventDateRow);
            txtMyEventDate.Text = "Date: " + mItems[position].Date;
            TextView txtMyEventTime =
row.FindViewById<TextView>(Resource.Id.txtEventTimeRow);
            txtMyEventTime.Text = "Time: " + mItems[position].Time;
            TextView txtMyEventCategory =
row.FindViewById<TextView>(Resource.Id.txtEventCategoryRow);
            txtMyEventCategory.Text = "Category: " + mItems[position].Category;
            TextView txtMyEventDescription =
row.FindViewById<TextView>(Resource.Id.txtEventDescriptionRow);
            txtMyEventDescription.Text = "Addational Details: " +
mItems[position].Description;

            if ((position % 2) == 1)
            {
                txtMyEventName.SetTextColor(Color.White);
                txtMyEventCreator.SetTextColor(Color.White);
                txtMyEventLocation.SetTextColor(Color.White);
                txtMyEventDate.SetTextColor(Color.White);
                txtMyEventTime.SetTextColor(Color.White);
                txtMyEventCategory.SetTextColor(Color.White);
                txtMyEventDescription.SetTextColor(Color.White);
            }
            else
            {
                txtMyEventName.SetTextColor(Color.Black);
```

```
                txtMyEventCreator.SetTextColor(Color.Black);
                txtMyEventLocation.SetTextColor(Color.Black);
                txtMyEventDate.SetTextColor(Color.Black);
                txtMyEventTime.SetTextColor(Color.Black);
                txtMyEventCategory.SetTextColor(Color.Black);
                txtMyEventDescription.SetTextColor(Color.Black);
            }

            return row;
        }

        private Color GetColorFromInteger(int color)
        {
            return Color.Rgb(Color.GetRedComponent(color),
Color.GetGreenComponent(color), Color.GetBlueComponent(color));
        }

    }
}
```

MainActivity.cs

```
namespace ProEventsApp
{
    [Activity(Label = "ProEventsApp", MainLauncher = true)]
    public class MainActivity : Activity
    {
        private Button btnSignIn;
        private Button btnSignUp;

        static ISharedPreferences pref =
Application.Context.GetSharedPreferences("UserInfo", FileCreationMode.Private);
        ISharedPreferencesEditor edit = pref.Edit();
        protected override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);
            SetContentView(Resource.Layout.Main);

            btnSignIn = FindViewById<Button>(Resource.Id.btnSignIn);
            btnSignIn.Click += (object sender, EventArgs args) =>
            {
                Toast.MakeText(this, "Welcome to Login", ToastLength.Short).Show();
                StartActivity(typeof(Dialog_SignIn));
            };

            btnSignUp = FindViewById<Button>(Resource.Id.btnSignUp);
            btnSignUp.Click += (object sender, EventArgs args) =>
            {
                StartActivity(typeof(Dialog_SignUp));
            };
        }
    }
}
```

MainProfile.cs

```
namespace ProEventsApp
{
    [Activity(Label = "MainProfile")]
    public class MainProfile : Activity
    {
        public static MobileServiceClient Client =
        new MobileServiceClient("https://proevents.azurewebsites.net");

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

        string username = pref.GetString("Username", "User");
        UserProf prof;
        protected async override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);
            SetContentView(Resource.Layout.Main_Profile);
            SetProfilePic();

            var homeBTN = FindViewById(Resource.Id.btnHome);
            var eventsBTN = FindViewById(Resource.Id.btnEvents);
            var connectionsBTN = FindViewById(Resource.Id.btnConnections);
            var logout = FindViewById(Resource.Id.btnLogout);
            var updateprofileBTN = FindViewById(Resource.Id.btnUpdateProfile);
            var viewFollowersBTN = FindViewById(Resource.Id.btnViewFollowers);

            CurrentPlatform.Init();
            List<UserProf> ls = await Client.GetTable<UserProf>().ToListAsync();
            prof = ls.FirstOrDefault(x => x.Username == username);
            TextView textName = (TextView)FindViewById(Resource.Id.textName);
            textName.Text = "Hello, my name is " + prof.Firstname;
            textName.Text += " " + prof.Lastname;
            textName.SetBackgroundResource(Resource.Drawable.round_style);
            TextView textCounty = (TextView)FindViewById(Resource.Id.textCounty);
            textCounty.Text = "I am from " + prof.County;
            textCounty.SetBackgroundResource(Resource.Drawable.round_style);
            TextView textProfession = (TextView)FindViewById(Resource.Id.textProfession);
            textProfession.Text = "My main Interest is " + prof.Profession;
            textProfession.SetBackgroundResource(Resource.Drawable.round_style);
            TextView textDescription =
(TextView)FindViewById(Resource.Id.textDescription);
            textDescription.Text = prof.Description;
            textDescription.SetBackgroundResource(Resource.Drawable.round_style);

            string userid = prof.ID;

            homeBTN.Click += HomeProcess;
            eventsBTN.Click += EventProcess;
            connectionsBTN.Click += ConnectionsProcess;
            logout.Click += LogoutProcess;
            updateprofileBTN.Click += UpdateUser;
            viewFollowersBTN.Click += ViewFollowers;
        }

        private void ViewFollowers(object sender, EventArgs e)
        {
            edit.PutString("Username", prof.Username);
            edit.PutString("UserID", prof.userID);
            edit.Commit();
            StartActivity(typeof(ViewFollowers));
        }
```

```
        private void UpdateUser(object sender, EventArgs e)
        {
            Toast.MakeText(this, "Update Profile", ToastLength.Short).Show();
            edit.PutString("Username", prof.Username);
            edit.PutString("UserID", prof.userID);
            edit.PutString("ID", prof.ID);
            edit.Commit();
            StartActivity(typeof(UpdateProfile));
        }

        private async void SetProfilePic()
        {
            List<UserProf> ls = await Client.GetTable<UserProf>().ToListAsync();
            UserProf prof = ls.FirstOrDefault(x => x.Username == username);
            string cat = prof.Profession;
            string pPic = GetPic(cat);
            ImageView proPic = FindViewById<ImageView>(Resource.Id.ivProPic);
            Stream i = Assets.Open(pPic);
            Drawable draw = Drawable.CreateFromStream(i, null);
            proPic.SetImageDrawable(draw);
        }

        private string GetPic(string cat)
        {
            string userPic = cat;
            return "profilePic/" + userPic + ".png";
        }
        private void HomeProcess(object sender, EventArgs e)
        {
            edit.PutString("Username", "User");
            StartActivity(typeof(MainProfile));
        }
        private void EventProcess(object sender, EventArgs e)
        {
            edit.PutString("Username", "User");
            StartActivity(typeof(Events));
        }
        private void ConnectionsProcess(object sender, EventArgs e)
        {
            edit.PutString("Username", "User");
            StartActivity(typeof(Connections));
        }
        private void LogoutProcess(object sender, EventArgs e)
        {
            edit.Clear();
            edit.Apply();
            Toast.MakeText(this, "Logged Out", ToastLength.Short).Show();
            StartActivity(typeof(MainActivity));
        }
    }
}
```

MyEvents.cs

```
namespace ProEventsApp
{
    [Activity(Label = "MyEvents")]
```

```
    public class MyEvents : Activity
    {
        public static MobileServiceClient Client =
        new MobileServiceClient("https://proevents.azurewebsites.net");

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

        string username = pref.GetString("Username", "User");

        private List<EventsDB> mytable = new List<EventsDB>();
        private ListView mListView;
        List<EventsDB> mItems;
        string emailsAtt;
        //private List<EventsDB> sendEmails;

        protected async override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);
            SetContentView(Resource.Layout.My_Events);

            mListView = FindViewById<ListView>(Resource.Id.myListViewEP);

            var myEventsBTN = FindViewById(Resource.Id.myEvents);
            var homeBTN = FindViewById(Resource.Id.btnHome);
            var eventsBTN = FindViewById(Resource.Id.btnEvents);
            var connectionsBTN = FindViewById(Resource.Id.btnConnections);
            var logout = FindViewById(Resource.Id.btnLogout);

            homeBTN.Click += HomeProcess;
            eventsBTN.Click += EventProcess;
            connectionsBTN.Click += ConnectionsProcess;
            logout.Click += LogoutProcess;
            CurrentPlatform.Init();
            IMobileServiceTable<EventsDB> mytable = Client.GetTable<EventsDB>();
            mItems = await mytable
                .Where(x => x.CUsername == username)
                .ToListAsync();
            LVMyEvents adapterLV = new LVMyEvents(this, mItems);
            mListView.Adapter = adapterLV;
            mListView.ItemClick += MListView_ItemClick;
        }

        public void MListView_ItemClick(object sender, AdapterView.ItemClickEventArgs e)
        {
            var popup = new PopupMenu(this, mListView.GetChildAt(e.Position -
mListView.FirstVisiblePosition));
            CurrentPlatform.Init();
            EventsDB thisEvent = mItems[e.Position];
            string myEventID = mItems[e.Position].ID;
            string myEventCreator = mItems[e.Position].CUsername;
            string myEventName = mItems[e.Position].EventName;
            string myEventCat = mItems[e.Position].Category;
            string myEventDate = mItems[e.Position].Date;
            string myEventTime = mItems[e.Position].Time;
            string myEventDes = mItems[e.Position].Description;

            /*IMobileServiceTable<AttendingDB> alll = Client.GetTable<AttendingDB>();
            List<AttendingDB> usersEvent = await alll
                .Where(x => x.AEventName == myEventName)
                .ToListAsync();*/
```

```
            popup.Inflate(Resource.Layout.MyMenu);
            popup.Show();
            popup.MenuItemClick += async (s, a) =>
            {
                switch (a.Item.ItemId)
                {
                    case Resource.Id.EditEvent:
                        edit.PutString("Username", username);
                        edit.PutString("Event", myEventName);
                        edit.PutString("ID", myEventID);
                        edit.Commit();
                        StartActivity(typeof(UpdateEvent));
                        Toast.MakeText(this, "Success", ToastLength.Short).Show();
                        break;
                    case Resource.Id.SendNotification:
                        IMobileServiceTable<AttendingDB> alll =
Client.GetTable<AttendingDB>();
                        List<AttendingDB> usersacc = await alll
                            .Where(x => x.AEventName == myEventName)
                            .ToListAsync();
                        int y = 0;
                        while(y < usersacc.Count)
                        {
                            await
EmailContext.SendEmailNoticeAsync(usersacc[y].AUserEmail, thisEvent);
                            Toast.MakeText(this, "TEST " + usersacc[y].AUserEmail,
ToastLength.Short).Show();
                            y++;
                        }
                        Toast.MakeText(this, "Successfully sent",
ToastLength.Short).Show();
                        break;
                    case Resource.Id.CancelEvent:
                        EventsDB cancelEvent = new EventsDB
                        {
                            ID = myEventID

                        };
                        await Client.GetTable<EventsDB>().DeleteAsync(cancelEvent);
                        IMobileServiceTable<AttendingDB> all =
Client.GetTable<AttendingDB>();
                        List<AttendingDB> list = await all
                            .Where(x => x.AEventName == myEventName)
                            .ToListAsync();
                        AttendingDB cancelFollow = new AttendingDB
                        {
                            ID = list[0].ID
                        };
                        await Client.GetTable<AttendingDB>().DeleteAsync(cancelFollow);

                        Toast.MakeText(this, "Event Cancelled",
ToastLength.Short).Show();
                        break;
                    default:
                        {
                            Toast.MakeText(this, "An error has occured, please try
again.", ToastLength.Short).Show();
                        }
                        break;
                }
            };
        }
```

```
        private void AllEventsSearch(object sender, EventArgs e)
        {
            ISharedPreferences prefs =
Application.Context.GetSharedPreferences("UserInfo", FileCreationMode.Private);
            ISharedPreferencesEditor edit = prefs.Edit();
            StartActivity(typeof(SearchEvent));
        }

        private void CreateEvent(object sender, EventArgs e)
        {
            ISharedPreferences prefs =
Application.Context.GetSharedPreferences("UserInfo", FileCreationMode.Private);
            ISharedPreferencesEditor edit = prefs.Edit();
            StartActivity(typeof(CreateEventForm));
        }

        private void HomeProcess(object sender, EventArgs e)
        {
            ISharedPreferences prefs =
Application.Context.GetSharedPreferences("UserInfo", FileCreationMode.Private);
            ISharedPreferencesEditor edit = prefs.Edit();
            StartActivity(typeof(MainProfile));
        }
        private void EventProcess(object sender, EventArgs e)
        {
            ISharedPreferences prefs =
Application.Context.GetSharedPreferences("UserInfo", FileCreationMode.Private);
            ISharedPreferencesEditor edit = prefs.Edit();
            StartActivity(typeof(Events));
        }
        private void ConnectionsProcess(object sender, EventArgs e)
        {
            ISharedPreferences prefs =
Application.Context.GetSharedPreferences("UserInfo", FileCreationMode.Private);
            ISharedPreferencesEditor edit = prefs.Edit();
            StartActivity(typeof(Connections));
        }
        private void LogoutProcess(object sender, EventArgs e)
        {
            ISharedPreferences prefs =
Application.Context.GetSharedPreferences("UserInfo", FileCreationMode.Private);
            ISharedPreferencesEditor edit = prefs.Edit();
            edit.Clear();
            edit.Commit();
            Toast.MakeText(this, "Logged Out", ToastLength.Short).Show();
            StartActivity(typeof(MainActivity));
        }
    }
}
```

NearbyConnections.cs

```
namespace ProEventsApp
{
    [Activity(Label = "NearbyConnections")]
    public class NearbyConnections : Activity
    {
        public static MobileServiceClient Client =
        new MobileServiceClient("https://proevents.azurewebsites.net");
```

```
        private List<UserProf> nItems;
        private List<UserProf> nItemsSP = new List<UserProf>();
        private ListView nListView;

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

        string username = pref.GetString("Username", "User");

        protected async override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);

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

            SetContentView(Resource.Layout.NearbyConn);

            var homeBTN = FindViewById(Resource.Id.btnHome);
            var eventsBTN = FindViewById(Resource.Id.btnEvents);
            var connectionsBTN = FindViewById(Resource.Id.btnConnections);
            var logout = FindViewById(Resource.Id.btnLogout);
            var widenLocationBTN = FindViewById(Resource.Id.btnWidenLocation);

            homeBTN.Click += HomeProcess;
            eventsBTN.Click += EventProcess;
            connectionsBTN.Click += ConnectionsProcess;
            logout.Click += LogoutProcess;
            widenLocationBTN.Click += ConnSearchProcess;

            nListView = FindViewById<ListView>(Resource.Id.myListViewNC);

            #region setUpDropdowns
            Spinner spinner = FindViewById<Spinner>(Resource.Id.spnUserCat);

            spinner.ItemSelected += new
EventHandler<AdapterView.ItemSelectedEventArgs>(spinner_ItemSelected);
            var adapter = ArrayAdapter.CreateFromResource(
                    this, Resource.Array.professions,
Android.Resource.Layout.SimpleSpinnerItem);

            adapter.SetDropDownViewResource(Android.Resource.Layout.SimpleSpinnerDropDownIt
em);
            spinner.Adapter = adapter;
            #endregion

            nListView.ItemClick += NListView_ItemClick;
        }

        private async void spinner_ItemSelected(object sender,
AdapterView.ItemSelectedEventArgs e)
        {
            Spinner spinner = (Spinner)sender;
            string spnCat = string.Format("{0}", spinner.GetItemAtPosition(e.Position));
            nItems = await Client.GetTable<UserProf>().ToListAsync();

            IMobileServiceTable<UserProf> tableFollow = Client.GetTable<UserProf>();
            List<UserProf> userList = await tableFollow
```

```
                .Where(x => x.Username == username)
                .ToListAsync();

            IMobileServiceTable<UserProf> table = Client.GetTable<UserProf>();
            nItems = await table
                .Where(x => x.Profession == spnCat && x.County == userList[0].County)
                .ToListAsync();

            ListViewAdapterDisplayUser adapterLV = new ListViewAdapterDisplayUser(this,
nItems);
            nListView.Adapter = adapterLV;

        }

        private void NListView_ItemClick(object sender, AdapterView.ItemClickEventArgs e)
        {
            var popup = new PopupMenu(this, nListView.GetChildAt(e.Position -
nListView.FirstVisiblePosition));
            CurrentPlatform.Init();
            string followingID = nItems[e.Position].userID;
            string followingUserName = nItems[e.Position].Username;
            string followingFirst = nItems[e.Position].Firstname;
            string followingSecond = nItems[e.Position].Lastname;
            string followingProf = nItems[e.Position].Profession;
            string followingLocation = nItems[e.Position].County;
            string followingDes = nItems[e.Position].Description;
            popup.Inflate(Resource.Layout.WFE);
            popup.Show();
            popup.MenuItemClick += async (s, a) =>
            {
                switch (a.Item.ItemId)
                {
                    case Resource.Id.FollowWFE:
                        FollowingUser followUser = new FollowingUser
                        {
                            Username = username,
                            FuserID = followingID,
                            FUsername = followingUserName,
                            FFirstname = followingFirst,
                            FLastname = followingSecond,
                            FProfession = followingProf,
                            FCounty = followingLocation,
                            FDescription = followingDes
                        };
                        await Client.GetTable<FollowingUser>().InsertAsync(followUser);
                        Toast.MakeText(this, "Success", ToastLength.Short).Show();
                        break;
                    case Resource.Id.CancelWFE:
                        Toast.MakeText(this, "Cancelled", ToastLength.Short).Show();
                        break;
                    default:
                        {
                            Toast.MakeText(this, "An error has occured, please try again.",
ToastLength.Short).Show();
                        }
                        break;
                }
            };
        }

        private void ConnSearchProcess(object sender, EventArgs e)
        {
            edit.PutString("Username", "User");
```

```
            StartActivity(typeof(ConnSearch));
        }

        private void HomeProcess(object sender, EventArgs e)
        {
            Toast.MakeText(this, "Home Page", ToastLength.Short).Show();
            edit.PutString("Username", "User");
            StartActivity(typeof(MainProfile));
        }
        private void EventProcess(object sender, EventArgs e)
        {
            Toast.MakeText(this, "Events(soon)", ToastLength.Short).Show();
            edit.PutString("Username", "User");
            StartActivity(typeof(Events));
        }
        private void ConnectionsProcess(object sender, EventArgs e)
        {
            Toast.MakeText(this, "Your Connections", ToastLength.Short).Show();
            edit.PutString("Username", "User");
            StartActivity(typeof(Connections));
        }

        private void LogoutProcess(object sender, EventArgs e)
        {
            edit.Clear();
            edit.Apply();
            Toast.MakeText(this, "Logged Out", ToastLength.Short).Show();
            StartActivity(typeof(MainActivity));
        }
    }
}
```

NearbyEvents .cs

```
namespace ProEventsApp
{
    [Activity(Label = "NearbyEvents")]
    public class NearbyEvents : Activity
    {
        public static MobileServiceClient Client =
        new MobileServiceClient("https://proevents.azurewebsites.net");

        private List<EventsDB> nItems;
        private List<EventsDB> nItemsSP = new List<EventsDB>();
        private ListView nListView;

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

        string username = pref.GetString("Username", "User");

        protected async override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);

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

```
            SetContentView(Resource.Layout.Nearby_Events);

            var homeBTN = FindViewById(Resource.Id.btnHome);
            var eventsBTN = FindViewById(Resource.Id.btnEvents);
            var connectionsBTN = FindViewById(Resource.Id.btnConnections);
            var logout = FindViewById(Resource.Id.btnLogout);
            var eventLocationBTN = FindViewById(Resource.Id.btnSearchEv);

            homeBTN.Click += HomeProcess;
            eventsBTN.Click += EventProcess;
            connectionsBTN.Click += ConnectionsProcess;
            logout.Click += LogoutProcess;
            eventLocationBTN.Click += BackSearchEvProcess;

            nListView = FindViewById<ListView>(Resource.Id.myListViewNE);

            #region setUpDropdowns
            Spinner spinner = FindViewById<Spinner>(Resource.Id.spnSearchEvents);

            spinner.ItemSelected += new
EventHandler<AdapterView.ItemSelectedEventArgs>(spinner_ItemSelected);
            var adapter = ArrayAdapter.CreateFromResource(
                    this, Resource.Array.professions,
Android.Resource.Layout.SimpleSpinnerItem);

            adapter.SetDropDownViewResource(Android.Resource.Layout.SimpleSpinnerDropDownIt
em);
            spinner.Adapter = adapter;
            #endregion

            nListView.ItemClick += EListView_ItemClick;
        }

        private async void spinner_ItemSelected(object sender,
AdapterView.ItemSelectedEventArgs e)
        {
            Spinner spinner = (Spinner)sender;
            string spnCat = string.Format("{0}", spinner.GetItemAtPosition(e.Position));

            IMobileServiceTable<UserProf> tableFollow = Client.GetTable<UserProf>();
            List<UserProf> userList = await tableFollow
                .Where(x => x.Username == username)
                .ToListAsync();

            IMobileServiceTable<EventsDB> table = Client.GetTable<EventsDB>();
            nItems = await table
                .Where(x => x.Category == spnCat && x.Location == userList[0].County)
                .ToListAsync();

            ListViewAdapter adapterLV = new ListViewAdapter(this, nItems);
            nListView.Adapter = adapterLV;
        }

        public async void EListView_ItemClick(object sender, AdapterView.ItemClickEventArgs
e)
        {
            var popup = new PopupMenu(this, nListView.GetChildAt(e.Position -
nListView.FirstVisiblePosition));
            CurrentPlatform.Init();
            string eventcreator = nItems[e.Position].CUsername;
```

```
            string eventname = nItems[e.Position].EventName;
            string eventid = nItems[e.Position].ID;
            string eventlocation = nItems[e.Position].Location;
            string eventdate = nItems[e.Position].Date;
            string eventcategory = nItems[e.Position].Category;
            string eventdescription = nItems[e.Position].Description;

            IMobileServiceTable<UserProf> tableFollow = Client.GetTable<UserProf>();
            List<UserProf> userList = await tableFollow
                .Where(x => x.Username == username)
                .ToListAsync();

            IMobileServiceTable<UserProf> alll = Client.GetTable<UserProf>();
            List<UserProf> userss = await alll
                .Where(x => x.Username == username && x.County == userList[0].County)
                .ToListAsync();
            string Auseremail = userss[0].Email;
            popup.Inflate(Resource.Layout.SEMenu);
            popup.Show();
            popup.MenuItemClick += async (s, a) =>
            {
                switch (a.Item.ItemId)
                {
                    case Resource.Id.Follow:
                        AttendingDB attendingEvent = new AttendingDB
                        {
                            ACreator = eventcreator,
                            AUsername = username,
                            AEventName = eventname,
                            AUserEmail = Auseremail,
                            AEventID = eventid,
                            ALocation = eventlocation,
                            ADate = eventdate,
                            ACategory = eventcategory,
                            ADescription = eventdescription
                        };
                        await Client.GetTable<AttendingDB>().InsertAsync(attendingEvent);
                        Toast.MakeText(this, "Success", ToastLength.Short).Show();
                        break;
                    case Resource.Id.Details:
                        edit.PutString("Username", username);
                        edit.PutString("EventName", eventname);
                        edit.Commit();
                        StartActivity(typeof(EventDetails));
                        break;
                    case Resource.Id.Cancel:
                        Toast.MakeText(this, "Cancelled", ToastLength.Short).Show();
                        break;
                    default:
                        {
                            Toast.MakeText(this, "An error has occured, please try again.",
ToastLength.Short).Show();
                        }
                        break;
                }
            };
        }

        private void BackSearchEvProcess(object sender, EventArgs e)
        {
            edit.PutString("Username", "User");
            StartActivity(typeof(SearchEvent));
        }
```

```
        private void HomeProcess(object sender, EventArgs e)
        {
            Toast.MakeText(this, "Home Page", ToastLength.Short).Show();
            edit.PutString("Username", "User");
            StartActivity(typeof(MainProfile));
        }
        private void EventProcess(object sender, EventArgs e)
        {
            Toast.MakeText(this, "Events(soon)", ToastLength.Short).Show();
            edit.PutString("Username", "User");
            StartActivity(typeof(Events));
        }
        private void ConnectionsProcess(object sender, EventArgs e)
        {
            Toast.MakeText(this, "Your Connections", ToastLength.Short).Show();
            edit.PutString("Username", "User");
            StartActivity(typeof(Connections));
        }

        private void LogoutProcess(object sender, EventArgs e)
        {
            edit.Clear();
            edit.Apply();
            Toast.MakeText(this, "Logged Out", ToastLength.Short).Show();
            StartActivity(typeof(MainActivity));
        }
    }
}
```

PasswordSaltHash.cs

```
namespace ProEventsApp
{
    class InvalidHashException : Exception
    {
        public InvalidHashException() { }
        public InvalidHashException(string message)
            : base(message) { }
        public InvalidHashException(string message, Exception inner)
            : base(message, inner) { }
    }

    class CannotPerformOperationException : Exception
    {
        public CannotPerformOperationException() { }
        public CannotPerformOperationException(string message)
            : base(message) { }
        public CannotPerformOperationException(string message, Exception inner)
            : base(message, inner) { }
    }

    class PasswordSaltHash
    {
        // These constants may be changed without breaking existing hashes.
        public const int SALT_BYTES = 24;
        public const int HASH_BYTES = 18;
        public const int PBKDF2_ITERATIONS = 64000;
```

```
        // These constants define the encoding and may not be changed.
        public const int HASH_SECTIONS = 5;
        public const int HASH_ALGORITHM_INDEX = 0;
        public const int ITERATION_INDEX = 1;
        public const int HASH_SIZE_INDEX = 2;
        public const int SALT_INDEX = 3;
        public const int PBKDF2_INDEX = 4;

        public static string CreateHash(string password)
        {
            // Generate a random salt
            byte[] salt = new byte[SALT_BYTES];
            try
            {
                using (RNGCryptoServiceProvider csprng = new RNGCryptoServiceProvider())
                {
                    csprng.GetBytes(salt);
                }
            }
            catch (CryptographicException ex)
            {
                throw new CannotPerformOperationException(
                    "Random number generator not available.",
                    ex
                );
            }
            catch (ArgumentNullException ex)
            {
                throw new CannotPerformOperationException(
                    "Invalid argument given to random number generator.",
                    ex
                );
            }

            byte[] hash = PBKDF2(password, salt, PBKDF2_ITERATIONS, HASH_BYTES);

            // format: algorithm:iterations:hashSize:salt:hash
            String parts = "sha1:" +
                PBKDF2_ITERATIONS +
                ":" +
                hash.Length +
                ":" +
                Convert.ToBase64String(salt) +
                ":" +
                Convert.ToBase64String(hash);
            return parts;
        }

        public static bool VerifyPassword(string password, string goodHash)
        {
            char[] delimiter = { ':' };
            string[] split = goodHash.Split(delimiter);

            if (split.Length != HASH_SECTIONS)
            {
                throw new InvalidHashException(
                    "Fields are missing from the password hash."
                );
            }

            // We only support SHA1 with C#.
            if (split[HASH_ALGORITHM_INDEX] != "sha1")
            {
```

```
            throw new CannotPerformOperationException(
                "Unsupported hash type."
            );
        }

        int iterations = 0;
        try
        {
            iterations = Int32.Parse(split[ITERATION_INDEX]);
        }
        catch (ArgumentNullException ex)
        {
            throw new CannotPerformOperationException(
                "Invalid argument given to Int32.Parse",
                ex
            );
        }
        catch (FormatException ex)
        {
            throw new InvalidHashException(
                "Could not parse the iteration count as an integer.",
                ex
            );
        }
        catch (OverflowException ex)
        {
            throw new InvalidHashException(
                "The iteration count is too large to be represented.",
                ex
            );
        }

        if (iterations < 1)
        {
            throw new InvalidHashException(
                "Invalid number of iterations. Must be >= 1."
            );
        }

        byte[] salt = null;
        try
        {
            salt = Convert.FromBase64String(split[SALT_INDEX]);
        }
        catch (ArgumentNullException ex)
        {
            throw new CannotPerformOperationException(
                "Invalid argument given to Convert.FromBase64String",
                ex
            );
        }
        catch (FormatException ex)
        {
            throw new InvalidHashException(
                "Base64 decoding of salt failed.",
                ex
            );
        }

        byte[] hash = null;
        try
        {
            hash = Convert.FromBase64String(split[PBKDF2_INDEX]);
```

```
            }
            catch (ArgumentNullException ex)
            {
                throw new CannotPerformOperationException(
                    "Invalid argument given to Convert.FromBase64String",
                    ex
                );
            }
            catch (FormatException ex)
            {
                throw new InvalidHashException(
                    "Base64 decoding of pbkdf2 output failed.",
                    ex
                );
            }

            int storedHashSize = 0;
            try
            {
                storedHashSize = Int32.Parse(split[HASH_SIZE_INDEX]);
            }
            catch (ArgumentNullException ex)
            {
                throw new CannotPerformOperationException(
                    "Invalid argument given to Int32.Parse",
                    ex
                );
            }
            catch (FormatException ex)
            {
                throw new InvalidHashException(
                    "Could not parse the hash size as an integer.",
                    ex
                );
            }
            catch (OverflowException ex)
            {
                throw new InvalidHashException(
                    "The hash size is too large to be represented.",
                    ex
                );
            }

            if (storedHashSize != hash.Length)
            {
                throw new InvalidHashException(
                    "Hash length doesn't match stored hash length."
                );
            }

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

        private static bool SlowEquals(byte[] a, byte[] b)
        {
            uint diff = (uint)a.Length ^ (uint)b.Length;
            for (int i = 0; i < a.Length && i < b.Length; i++)
            {
                diff |= (uint)(a[i] ^ b[i]);
            }
            return diff == 0;
        }
```

```
        private static byte[] PBKDF2(string password, byte[] salt, int iterations, int
outputBytes)
        {
            using (Rfc2898DeriveBytes pbkdf2 = new Rfc2898DeriveBytes(password, salt))
            {
                pbkdf2.IterationCount = iterations;
                return pbkdf2.GetBytes(outputBytes);
            }
        }
    }
}
```

SearchEvent .cs

```
namespace ProEventsApp
{
    [Activity(Label = "SearchEvent")]
    public class SearchEvent : Activity
    {
        public static MobileServiceClient Client =
        new MobileServiceClient("https://proevents.azurewebsites.net");

        private List<EventsDB> eItems;
        private List<EventsDB> eItemsSP = new List<EventsDB>();
        private ListView eListView;

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

        string username = pref.GetString("Username", "User");

        protected async override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);

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

            SetContentView(Resource.Layout.Search_Event);

            var homeBTN = FindViewById(Resource.Id.btnHome);
            var eventsBTN = FindViewById(Resource.Id.btnEvents);
            var connectionsBTN = FindViewById(Resource.Id.btnConnections);
            var logout = FindViewById(Resource.Id.btnLogout);
            var nearbyBTN = FindViewById(Resource.Id.btnWidenEvent);

            homeBTN.Click += HomeProcess;
            eventsBTN.Click += EventProcess;
            connectionsBTN.Click += ConnectionsProcess;
            logout.Click += LogoutProcess;
            nearbyBTN.Click += NearbyEventsProcess;

            eListView = FindViewById<ListView>(Resource.Id.myListView);

            #region setUpDropdowns
```

```
            Spinner spinner = FindViewById<Spinner>(Resource.Id.spnSearchEvents);

            spinner.ItemSelected += new
EventHandler<AdapterView.ItemSelectedEventArgs>(spinner_ItemSelected);
            var adapter = ArrayAdapter.CreateFromResource(
                    this, Resource.Array.professions,
Android.Resource.Layout.SimpleSpinnerItem);

            adapter.SetDropDownViewResource(Android.Resource.Layout.SimpleSpinnerDropDownIt
em);
            spinner.Adapter = adapter;
            #endregion

            eListView.ItemClick += EListView_ItemClick;
        }

        private async void spinner_ItemSelected(object sender,
AdapterView.ItemSelectedEventArgs e)
        {
            Spinner spinner = (Spinner)sender;
            string spnCat = string.Format("{0}", spinner.GetItemAtPosition(e.Position));
            //string formatDate = "{d, 0:MMMM, yyyy}";
            IMobileServiceTable<EventsDB> table = Client.GetTable<EventsDB>();
            eItems = await table
                .Where(x => x.Category == spnCat)
                .ToListAsync();

            ListViewAdapter adapterLV = new ListViewAdapter(this, eItems);
            eListView.Adapter = adapterLV;
        }

        public async void EListView_ItemClick(object sender, AdapterView.ItemClickEventArgs
e)
        {
            var popup = new PopupMenu(this, eListView.GetChildAt(e.Position -
eListView.FirstVisiblePosition));
            CurrentPlatform.Init();
            string eventcreator = eItems[e.Position].CUsername;
            string eventname = eItems[e.Position].EventName;
            string eventid = eItems[e.Position].ID;
            string eventlocation = eItems[e.Position].Location;
            string eventdate = eItems[e.Position].Date;
            string eventcategory = eItems[e.Position].Category;
            string eventdescription = eItems[e.Position].Description;

            IMobileServiceTable<UserProf> alll = Client.GetTable<UserProf>();
            List<UserProf> userss = await alll
                .Where(x => x.Username == username)
                .ToListAsync();
            string Auseremail = userss[0].Email;
            popup.Inflate(Resource.Layout.SEMenu);
            popup.Show();
            popup.MenuItemClick += async (s, a) =>
            {
                switch (a.Item.ItemId)
                {
                    case Resource.Id.Follow:
                        AttendingDB attendingEvent = new AttendingDB
                        {
                            ACreator = eventcreator,
                            AUsername = username,
                            AEventName = eventname,
```

```
                            AUserEmail = Auseremail,
                            AEventID = eventid,
                            ALocation = eventlocation,
                            ADate = eventdate,
                            ACategory = eventcategory,
                            ADescription = eventdescription
                        };
                        await Client.GetTable<AttendingDB>().InsertAsync(attendingEvent);
                        Toast.MakeText(this, "Success", ToastLength.Short).Show();
                        break;
                    case Resource.Id.Details:
                        edit.PutString("Username", username);
                        edit.PutString("EventName", eventname);
                        edit.Commit();
                        StartActivity(typeof(EventDetails));
                        break;
                    case Resource.Id.Cancel:
                        Toast.MakeText(this, "Cancelled", ToastLength.Short).Show();
                        break;
                    default:
                        {
                            Toast.MakeText(this, "An error has occured, please try again.",
ToastLength.Short).Show();
                        }
                        break;
                }
            };
        }

        private void NearbyEventsProcess(object sender, EventArgs e)
        {
            edit.PutString("Username", "User");
            StartActivity(typeof(NearbyEvents));
        }

        private void HomeProcess(object sender, EventArgs e)
        {
            Toast.MakeText(this, "Home Page", ToastLength.Short).Show();
            edit.PutString("Username", "User");
            StartActivity(typeof(MainProfile));
        }
        private void EventProcess(object sender, EventArgs e)
        {
            Toast.MakeText(this, "Events(soon)", ToastLength.Short).Show();
            edit.PutString("Username", "User");
            StartActivity(typeof(Events));
        }
        private void ConnectionsProcess(object sender, EventArgs e)
        {
            Toast.MakeText(this, "Your Connections", ToastLength.Short).Show();
            edit.PutString("Username", "User");
            StartActivity(typeof(Connections));
        }

        private void LogoutProcess(object sender, EventArgs e)
        {
            edit.Clear();
            edit.Apply();
            Toast.MakeText(this, "Logged Out", ToastLength.Short).Show();
            StartActivity(typeof(MainActivity));
        }
    }
}
```

SetUpProfile.cs

```
namespace ProEventsApp
{
    [Activity(Label = "SetUpProfile")]
    public class SetUpProfile : Activity
    {
        public static MobileServiceClient Client =
        new MobileServiceClient("https://proevents.azurewebsites.net");

        string username = pref.GetString("Username", "User");

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

        protected override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);
            SetContentView(Resource.Layout.Profile_Set_Up);
            #region setUpDropdowns
            Spinner countySelect = FindViewById<Spinner>(Resource.Id.county);
            var countyAdapter = ArrayAdapter.CreateFromResource(
            this, Resource.Array.county, Android.Resource.Layout.SimpleSpinnerItem);

            countyAdapter.SetDropDownViewResource(Android.Resource.Layout.SimpleSpinnerDrop
DownItem);
            countySelect.Adapter = countyAdapter;

            Spinner eventSelect = FindViewById<Spinner>(Resource.Id.professions);
            var eventAdapter = ArrayAdapter.CreateFromResource(
            this, Resource.Array.professions, Android.Resource.Layout.SimpleSpinnerItem);

            eventAdapter.SetDropDownViewResource(Android.Resource.Layout.SimpleSpinnerDropD
ownItem);
            eventSelect.Adapter = eventAdapter;
            #endregion

            var saveButton = FindViewById(Resource.Id.btnSave);
            saveButton.Click += ValidateForm;
        }
        private async void TrySave()
        {
            CurrentPlatform.Init();
            EditText firstName = (EditText)FindViewById(Resource.Id.txtFirstName);
            EditText lastName = (EditText)FindViewById(Resource.Id.txtSecondName);
            EditText email = (EditText)FindViewById(Resource.Id.txtEmail);
            Spinner profession = (Spinner)FindViewById(Resource.Id.professions);
            Spinner county = (Spinner)FindViewById(Resource.Id.county);
            EditText description = (EditText)FindViewById(Resource.Id.txtDescripton);
            UserProf newUserInfo = new UserProf
            {
                userID = pref.GetString("UserID", "NULL"),
                Username = pref.GetString("Username", "NULL"),
                Firstname = firstName.Text,
                Lastname = lastName.Text,
```

```
                Email = email.Text,
                Profession = profession.SelectedItem.ToString(),
                Description = description.Text,
                County = county.SelectedItem.ToString()
            };
            await Client.GetTable<UserProf>().InsertAsync(newUserInfo);
            Toast.MakeText(ApplicationContext, "User " + username + " info created!",
ToastLength.Short).Show();
        }
        private void ValidateForm(object sender, EventArgs e)
        {
            EditText firstName = (EditText)FindViewById(Resource.Id.txtFirstName);
            EditText lastName = (EditText)FindViewById(Resource.Id.txtSecondName);
            EditText email = (EditText)FindViewById(Resource.Id.txtEmail);
            Spinner county = (Spinner)FindViewById(Resource.Id.county);
            EditText description = (EditText)FindViewById(Resource.Id.txtDescripton);
            string namePattern = "[^a-zA-Z]";
            bool validInput = false;
            #region fieldValidation
            if (firstName.Text == "")
            {
                firstName.Error = "Cannot be Empty";
                firstName.RequestFocus();
            }
            else if (lastName.Text == "")
            {
                lastName.Error = "Cannot be Empty";
                lastName.RequestFocus();
            }
            else if (description.Text == "")
            {
                description.Error = "Cannot be Empty";
                description.RequestFocus();
            }
            else if (Regex.IsMatch(firstName.Text, namePattern))
            {
                firstName.Error = "Invalid characters in name";
                firstName.RequestFocus();
            }
            else if (Regex.IsMatch(lastName.Text, namePattern))
            {
                lastName.Error = "Invalid characters in name";
                lastName.RequestFocus();
            }
            else
            {
                validInput = true;
            }
            #endregion
            if (validInput)
            {
                //edit.PutString("Username", username);
                //edit.Commit();
                TrySave();
                Toast.MakeText(this, "Account Created Successfully!",
ToastLength.Short).Show();
                StartActivity(typeof(Dialog_SignIn));
            }
        }
    }
}
```

UpdateEvent.cs

```
namespace ProEventsApp
{
    [Activity(Label = "EventEvent")]
    public class UpdateEvent : Activity
    {
        public static MobileServiceClient Client =
        new MobileServiceClient("https://proevents.azurewebsites.net");

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

        string username = pref.GetString("Username", "User");
        string eventid = pref.GetString("Event", "");
        string id = pref.GetString("ID", "");

        protected async override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);
            SetContentView(Resource.Layout.Update_Event);

            Button dateBTN = FindViewById<Button>(Resource.Id.btnDateU);
            dateBTN.Click += DateOnClick;
            Button timeBTN = FindViewById<Button>(Resource.Id.btnTimeU);
            timeBTN.Click += TimeOnClick;

            #region setUpDropdowns
            Spinner eventSelect = FindViewById<Spinner>(Resource.Id.spnEventsU);
            var eventAdapter = ArrayAdapter.CreateFromResource(
            this, Resource.Array.professions, Android.Resource.Layout.SimpleSpinnerItem);

            eventAdapter.SetDropDownViewResource(Android.Resource.Layout.SimpleSpinnerDropD
ownItem);
            eventSelect.Adapter = eventAdapter;

            Spinner locationSelect = FindViewById<Spinner>(Resource.Id.spnLocationU);
            var locationAdapter = ArrayAdapter.CreateFromResource(
            this, Resource.Array.county, Android.Resource.Layout.SimpleSpinnerItem);

            locationAdapter.SetDropDownViewResource(Android.Resource.Layout.SimpleSpinnerDr
opDownItem);
            locationSelect.Adapter = locationAdapter;
            #endregion

            var saveButton = FindViewById(Resource.Id.btnConfirmUE);
            saveButton.Click += ValidateForm;
            //Create your application here
        }
        private async void TrySave()
        {
            CurrentPlatform.Init();
            EditText eventName = (EditText)FindViewById(Resource.Id.txtEventNameU);
            EditText address = (EditText)FindViewById(Resource.Id.txtEventAddessU);
            Spinner location = (Spinner)FindViewById(Resource.Id.spnLocationU);
            TextView date = (TextView)FindViewById(Resource.Id.txtDateU);
            Button dateBTN = FindViewById<Button>(Resource.Id.btnDateU);
            dateBTN.Click += DateOnClick;
            Button timeBTN = FindViewById<Button>(Resource.Id.btnTimeU);
```

```
            timeBTN.Click += TimeOnClick;
            TextView time = (TextView)FindViewById(Resource.Id.txtTimeU);
            Spinner category = (Spinner)FindViewById(Resource.Id.spnEventsU);
            EditText description = (EditText)FindViewById(Resource.Id.txtEventDescriptonU);
            EventsDB UpdateEvent = new EventsDB
            {
                ID = id,
                CUsername = username,
                EventName = eventName.Text,
                Location = location.SelectedItem.ToString(),
                Address = address.Text,
                Time = time.Text,
                Date = date.Text,
                Category = category.SelectedItem.ToString(),
                Description = description.Text
            };
            await Client.GetTable<EventsDB>().UpdateAsync(UpdateEvent);

            IMobileServiceTable<AttendingDB> all = Client.GetTable<AttendingDB>();
            List<AttendingDB> list = await all
                .Where(x => x.AEventID == id)
                .ToListAsync();
            //string attendid = list[0].ID;
            AttendingDB UpdateAttendEvent = new AttendingDB
            {
                ID = list[0].ID,
                ACreator = username,
                AUsername = username,
                AEventName = eventName.Text,
                AEventID = eventid,
                AAdress = address.Text,
                ALocation = location.SelectedItem.ToString(),
                ADate = date.Text,
                ATime = time.Text,
                ACategory = category.SelectedItem.ToString(),
                ADescription = description.Text
            };
            await Client.GetTable<AttendingDB>().UpdateAsync(UpdateAttendEvent);
            //Toast.MakeText(ApplicationContext, "User " + pref.GetString("UserName",
"NULL") + " created an event!", ToastLength.Short).Show();
        }

        private void DateOnClick(object sender, EventArgs e)
        {
            DatePickerFragmentUp DPicker = DatePickerFragmentUp.NewInstance(delegate
(DateTime time)
            {
                TextView datetxt = (TextView)FindViewById(Resource.Id.txtDateU);
                DateTime eventDate;
                datetxt.Text = time.ToLongDateString();
                eventDate = time;
            });
            DPicker.Show(FragmentManager, DatePickerFragmentUp.TAG);
        }
        void TimeOnClick(object sender, EventArgs e)
        {
            TimePickerFragmentUp TPicker = TimePickerFragmentUp.NewInstance(delegate
(DateTime time)
            {
                TextView timetxt = (TextView)FindViewById(Resource.Id.txtTimeU);
                DateTime eventTime;
                timetxt.Text = time.ToShortTimeString();
                eventTime = time;
            });
```

```
            TPicker.Show(FragmentManager, TimePickerFragmentUp.TAG);
        }

        private void ValidateForm(object sender, EventArgs e)
        {
            EditText eventName = (EditText)FindViewById(Resource.Id.txtEventNameU);
            TextView datetxt = (TextView)FindViewById(Resource.Id.txtDateU);
            Spinner category = (Spinner)FindViewById(Resource.Id.spnEventsU);
            EditText description = (EditText)FindViewById(Resource.Id.txtEventDescriptonU);
            bool validInput = false;
            #region fieldValidation
            if (eventName.Text == "")
            {
                eventName.Error = "Cannot be Empty";
                eventName.RequestFocus();
            }
            else if (datetxt.Text == "")
            {
                datetxt.Error = "Cannot be Empty";
                datetxt.RequestFocus();
            }
            else if (description.Text == "")
            {
                description.Error = "Cannot be Empty";
                description.RequestFocus();
            }
            else
            {
                validInput = true;
            }
            #endregion
            if (validInput)
            {
                TrySave();
                Toast.MakeText(this, "Event Updated Successfully",
ToastLength.Short).Show();
                edit.PutString("Username", username);
                StartActivity(typeof(MyEvents));
            }
        }
    }

    public class DatePickerFragmentUp : DialogFragment, DatePickerDialog.IOnDateSetListener
    {
        // TAG can be any string of your choice.
        public static readonly string TAG = "X:" +
typeof(DatePickerFragmentUp).Name.ToUpper();

        // Initialize this value to prevent NullReferenceExceptions.
        Action<DateTime> dateSelectedU = delegate { };

        public static DatePickerFragmentUp NewInstance(Action<DateTime> date)
        {
            DatePickerFragmentUp DPicker = new DatePickerFragmentUp();
            DPicker.dateSelectedU = date;
            return DPicker;
        }

        public override Dialog OnCreateDialog(Bundle savedInstanceState)
        {
            DateTime currently = DateTime.Now;
            DatePickerDialog DPicker = new DatePickerDialog(Activity, this, currently.Year,
currently.Month - 1, currently.Day);
```

```
            return DPicker;
        }

        public void OnDateSet(DatePicker view, int year, int month, int day)
        {
            DateTime selectedDate = new DateTime(year, month + 1, day);
            Log.Debug(TAG, selectedDate.ToLongDateString());
            dateSelectedU(selectedDate);
        }
    }

    public class TimePickerFragmentUp : DialogFragment, TimePickerDialog.IOnTimeSetListener
    {
        public static readonly string TAG = "MyTimePickerFragment";
        Action<DateTime> timeSelectedU = delegate { };

        public static TimePickerFragmentUp NewInstance(Action<DateTime> time)
        {
            TimePickerFragmentUp TPicker = new TimePickerFragmentUp();
            TPicker.timeSelectedU = time;
            return TPicker;
        }

        public override Dialog OnCreateDialog(Bundle savedInstanceState)
        {
            DateTime currentTime = DateTime.Now;
            bool is24HourFormat = DateFormat.Is24HourFormat(Activity);
            TimePickerDialog TPicker = new TimePickerDialog(Activity, this,
currentTime.Hour, currentTime.Minute, is24HourFormat);
            return TPicker;
        }

        public void OnTimeSet(TimePicker view, int hour, int minute)
        {
            DateTime currentTime = DateTime.Now;
            DateTime selectedTime = new DateTime(currentTime.Year, currentTime.Month,
currentTime.Day, hour, minute, 0);
            Log.Debug(TAG, selectedTime.ToLongTimeString());
            timeSelectedU(selectedTime);
        }
    }
}
```

UpdateProfile.cs

```
namespace ProEventsApp
{
    [Activity(Label = "UpdateProfile")]
    public class UpdateProfile : Activity
    {
        public static MobileServiceClient Client =
        new MobileServiceClient("https://proevents.azurewebsites.net");

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

```
        string username = pref.GetString("Username", "username");
        string userid = pref.GetString("UserID", "user");
        string id = pref.GetString("ID", "id");

        protected async override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);
            SetContentView(Resource.Layout.Update_Profile);
            #region setUpDropdowns
            Spinner countySelect = FindViewById<Spinner>(Resource.Id.countyU);
            var countyAdapter = ArrayAdapter.CreateFromResource(
            this, Resource.Array.county, Android.Resource.Layout.SimpleSpinnerItem);

            countyAdapter.SetDropDownViewResource(Android.Resource.Layout.SimpleSpinnerDrop
DownItem);
            countySelect.Adapter = countyAdapter;

            Spinner eventSelect = FindViewById<Spinner>(Resource.Id.professionU);
            var eventAdapter = ArrayAdapter.CreateFromResource(
            this, Resource.Array.professions, Android.Resource.Layout.SimpleSpinnerItem);

            eventAdapter.SetDropDownViewResource(Android.Resource.Layout.SimpleSpinnerDropD
ownItem);
            eventSelect.Adapter = eventAdapter;
            #endregion

            var saveButton = FindViewById(Resource.Id.btnSaveU);
            saveButton.Click += ValidateForm;
        }
        private async void TrySave()
        {
            CurrentPlatform.Init();
            EditText firstName = (EditText)FindViewById(Resource.Id.txtFirstNameU);
            EditText lastName = (EditText)FindViewById(Resource.Id.txtSecondNameU);
            EditText email = (EditText)FindViewById(Resource.Id.txtEmailU);
            Spinner professions = (Spinner)FindViewById(Resource.Id.professionU);
            Spinner county = (Spinner)FindViewById(Resource.Id.countyU);
            EditText description = (EditText)FindViewById(Resource.Id.txtDescriptonU);
            UserProf UpdateUserInfo = new UserProf
            {
                //userID = userid,
                ID = pref.GetString("ID", "id"),
                userID = pref.GetString("UserID", "user"),
                Username = pref.GetString("Username", "username"),
                //Username = username,
                Firstname = firstName.Text,
                Lastname = lastName.Text,
                Email = email.Text,
                Profession = professions.SelectedItem.ToString(),
                Description = description.Text,
                County = county.SelectedItem.ToString()
            };
           // Toast.MakeText(this, )
            await Client.GetTable<UserProf>().UpdateAsync(UpdateUserInfo);

            Toast.MakeText(ApplicationContext, "User " + username + " info Updated!",
ToastLength.Short).Show();
        }
        private void ValidateForm(object sender, EventArgs e)
        {
            EditText firstName = (EditText)FindViewById(Resource.Id.txtFirstNameU);
            EditText lastName = (EditText)FindViewById(Resource.Id.txtSecondNameU);
            EditText email = (EditText)FindViewById(Resource.Id.txtEmailU);
```

```
            Spinner profession = (Spinner)FindViewById(Resource.Id.professionU);
            Spinner county = (Spinner)FindViewById(Resource.Id.countyU);
            EditText description = (EditText)FindViewById(Resource.Id.txtDescriptonU);
            string namePattern = "[^a-zA-Z]";
            bool validInput = false;
            #region fieldValidation
            if (firstName.Text == "")
            {
                firstName.Error = "Cannot be Empty";
                firstName.RequestFocus();
            }
            else if (lastName.Text == "")
            {
                lastName.Error = "Cannot be Empty";
                lastName.RequestFocus();
            }
            else if (description.Text == "")
            {
                description.Error = "Cannot be Empty";
                description.RequestFocus();
            }
            else if (Regex.IsMatch(firstName.Text, namePattern))
            {
                firstName.Error = "Invalid characters in name";
                firstName.RequestFocus();
            }
            else if (Regex.IsMatch(lastName.Text, namePattern))
            {
                lastName.Error = "Invalid characters in name";
                lastName.RequestFocus();
            }
            else
            {
                validInput = true;
            }
            #endregion
            if (validInput)
            {
                TrySave();
                edit.PutString("Username", username);
                edit.Commit();
                Finish();
            }
        }
    }
}
```

UserProf.cs

```
namespace ProEventsApp
{
    public class UserProf
    {
        public string ID { get; set; }
        public string userID { get; set; }
        public string Username { get; set; }
        public string Firstname { get; set; }
        public string Lastname { get; set; }
        public string Email { get; set; }
```

```
        public string Profession { get; set; }
        public string County { get; set; }
        public string Description { get; set; }
    }
}
```

users.cs

```
namespace ProEventsApp
{
    public class users
    {
        public string ID { get; set; }
        public string username { get; set; }
        public string password { get; set; }
    }
}
```

ViewEventsFollow.cs

```
namespace ProEventsApp
{
    [Activity(Label = "ViewEventsFollow")]
    public class ViewEventsFollow : Activity
    {
        public static MobileServiceClient Client =
        new MobileServiceClient("https://proevents.azurewebsites.net");

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

        private List<AttendingDB> table = new List<AttendingDB>();
        private ListView aListView;

        string username = pref.GetString("Username", "User");
        string userprofile = pref.GetString("Userprofile", "userprofile");

        protected async override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);
            SetContentView(Resource.Layout.ViewEvents_Follow);

            aListView = FindViewById<ListView>(Resource.Id.myListViewEP);

            var backBTN = FindViewById(Resource.Id.btnBackViewEvents);
            var homeBTN = FindViewById(Resource.Id.btnHome);
            var eventsBTN = FindViewById(Resource.Id.btnEvents);
            var connectionsBTN = FindViewById(Resource.Id.btnConnections);
            var logout = FindViewById(Resource.Id.btnLogout);

            homeBTN.Click += HomeProcess;
            eventsBTN.Click += EventProcess;
```

```
            connectionsBTN.Click += ConnectionsProcess;
            logout.Click += LogoutProcess;
            backBTN.Click += BackProcess;
            CurrentPlatform.Init();
            IMobileServiceTable<AttendingDB> table = Client.GetTable<AttendingDB>();
            List<AttendingDB> eItems = await table
                .Where(x => x.AUsername == userprofile)
                .ToListAsync();
            ListViewAdapterAtt adapterLV = new ListViewAdapterAtt(this, eItems);
            aListView.Adapter = adapterLV;
        }

        private void BackProcess(object sender, EventArgs e)
        {
            ISharedPreferences prefs =
Application.Context.GetSharedPreferences("UserInfo", FileCreationMode.Private);
            ISharedPreferencesEditor edit = prefs.Edit();
            edit.Clear();
            edit.Commit();
            Finish();
        }

        private void HomeProcess(object sender, EventArgs e)
        {
            ISharedPreferences prefs =
Application.Context.GetSharedPreferences("UserInfo", FileCreationMode.Private);
            ISharedPreferencesEditor edit = prefs.Edit();
            Toast.MakeText(this, "Home Page", ToastLength.Short).Show();
            StartActivity(typeof(MainProfile));
        }
        private void EventProcess(object sender, EventArgs e)
        {
            ISharedPreferences prefs =
Application.Context.GetSharedPreferences("UserInfo", FileCreationMode.Private);
            ISharedPreferencesEditor edit = prefs.Edit();
            Toast.MakeText(this, "Events(soon)", ToastLength.Short).Show();
            StartActivity(typeof(Events));
        }
        private void ConnectionsProcess(object sender, EventArgs e)
        {
            ISharedPreferences prefs =
Application.Context.GetSharedPreferences("UserInfo", FileCreationMode.Private);
            ISharedPreferencesEditor edit = prefs.Edit();
            Toast.MakeText(this, "Your Connections", ToastLength.Short).Show();
            //StartActivity(typeof(Connections));
        }
        private void LogoutProcess(object sender, EventArgs e)
        {
            ISharedPreferences prefs =
Application.Context.GetSharedPreferences("UserInfo", FileCreationMode.Private);
            ISharedPreferencesEditor edit = prefs.Edit();
            edit.Clear();
            edit.Commit();
            Toast.MakeText(this, "Logged Out", ToastLength.Short).Show();
            StartActivity(typeof(MainActivity));
        }
    }
}
```

ViewFollowers.cs

```
namespace ProEventsApp
{
    [Activity(Label = "ViewFollowers")]
    public class ViewFollowers : Activity
    {
        public static MobileServiceClient Client =
        new MobileServiceClient("https://proevents.azurewebsites.net");

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

        string username = pref.GetString("Username", "User");

        private List<FollowingUser> tableFollow = new List<FollowingUser>();
        List<UserProf> prof = new List<UserProf>();
        private ListView fListView;
        private List<FollowingUser> fItems;

        protected async override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);
            SetContentView(Resource.Layout.Connections_Page);

            fListView = FindViewById<ListView>(Resource.Id.myListViewConn);

            var searchBTN = FindViewById(Resource.Id.btnSeachUsers);
            var homeBTN = FindViewById(Resource.Id.btnHome);
            var eventsBTN = FindViewById(Resource.Id.btnEvents);
            var connectionsBTN = FindViewById(Resource.Id.btnConnections);
            var logout = FindViewById(Resource.Id.btnLogout);

            homeBTN.Click += HomeProcess;
            eventsBTN.Click += EventProcess;
            connectionsBTN.Click += ConnectionsProcess;
            logout.Click += LogoutProcess;
            searchBTN.Click += AllUsersSearch;

            CurrentPlatform.Init();
            IMobileServiceTable<FollowingUser> allfollows =
Client.GetTable<FollowingUser>();
            fItems = await allfollows
                .Where(x => x.FUsername == username)
                .ToListAsync();
            int y = 0;
            while(y < fItems.Count)
            {
                string fListItem = fItems[y].Username;


                IMobileServiceTable<UserProf> allusers = Client.GetTable<UserProf>();
                List<UserProf> res = await allusers
                    .Where(x => x.Username.Contains(fListItem))
                    .ToListAsync();
                prof.Add(res[0]);
                y++;
            }
            ListViewAdapterDisplayUser adapterLV = new ListViewAdapterDisplayUser(this,
prof);
```

```
            fListView.Adapter = adapterLV;

            fListView.ItemClick += FListView_ItemClick;
        }

        private void FListView_ItemClick(object sender, AdapterView.ItemClickEventArgs e)
        {
            var popup = new PopupMenu(this, fListView.GetChildAt(e.Position));
            CurrentPlatform.Init();
            string followingID = fItems[e.Position].FuserID;
            string followingUserName = fItems[e.Position].Username;
            string followingFirst = fItems[e.Position].FFirstname;
            string followingSecond = fItems[e.Position].FLastname;
            string followingProf = fItems[e.Position].FProfession;
            string followingLocation = fItems[e.Position].FCounty;
            string followingDes = fItems[e.Position].FDescription;
            popup.Inflate(Resource.Layout.ViewProfileMenu);
            popup.Show();
            popup.MenuItemClick += (s, a) =>
            {
                switch (a.Item.ItemId)
                {
                    case Resource.Id.ViewVPM:
                        ISharedPreferences prefs =
Application.Context.GetSharedPreferences("UserInfo", FileCreationMode.Private);
                        ISharedPreferencesEditor edit = prefs.Edit();
                        string userprofile = prof[e.Position].Username;
                        username = pref.GetString("Username", "User");
                        edit.PutString("Username", username);
                        edit.PutString("Userprofile", userprofile);
                        edit.Commit();
                        StartActivity(typeof(ViewProfile));
                        break;
                    case Resource.Id.FollowVPM:
                        FollowingUser followUser = new FollowingUser
                        {
                            Username = username,
                            FuserID = followingID,
                            FUsername = followingUserName,
                            FFirstname = followingFirst,
                            FLastname = followingSecond,
                            FProfession = followingProf,
                            FCounty = followingLocation,
                            FDescription = followingDes
                        };
                        Toast.MakeText(this, "Followed", ToastLength.Short).Show();
break;
                    default:
                        {
                            Toast.MakeText(this, "An error has occured, please try
again.", ToastLength.Short).Show();
                        }
                        break;
                }
            };
        }

        private void AllUsersSearch(object sender, EventArgs e)
        {
            ISharedPreferences prefs =
Application.Context.GetSharedPreferences("UserInfo", FileCreationMode.Private);
            ISharedPreferencesEditor edit = prefs.Edit();
            Toast.MakeText(this, "Search for an Event", ToastLength.Short).Show();
            StartActivity(typeof(ConnSearch));
```

```
        }
        private void HomeProcess(object sender, EventArgs e)
        {
            ISharedPreferences prefs =
Application.Context.GetSharedPreferences("UserInfo", FileCreationMode.Private);
            ISharedPreferencesEditor edit = prefs.Edit();
            Toast.MakeText(this, "Home Page", ToastLength.Short).Show();
            StartActivity(typeof(MainProfile));
        }
        private void EventProcess(object sender, EventArgs e)
        {
            ISharedPreferences prefs =
Application.Context.GetSharedPreferences("UserInfo", FileCreationMode.Private);
            ISharedPreferencesEditor edit = prefs.Edit();
            Toast.MakeText(this, "Events(soon)", ToastLength.Short).Show();
            StartActivity(typeof(Events));
        }
        private void ConnectionsProcess(object sender, EventArgs e)
        {
            ISharedPreferences prefs =
Application.Context.GetSharedPreferences("UserInfo", FileCreationMode.Private);
            ISharedPreferencesEditor edit = prefs.Edit();
            Toast.MakeText(this, "Your Connections", ToastLength.Short).Show();
            StartActivity(typeof(Connections));
        }
        private void LogoutProcess(object sender, EventArgs e)
        {
            ISharedPreferences prefs =
Application.Context.GetSharedPreferences("UserInfo", FileCreationMode.Private);
            ISharedPreferencesEditor edit = prefs.Edit();
            edit.Clear();
            edit.Commit();
            Toast.MakeText(this, "Logged Out", ToastLength.Short).Show();
            StartActivity(typeof(MainActivity));
        }
    }
}
```

ViewProfile.cs

```
namespace ProEventsApp
{
    [Activity(Label = "ViewProfile")]
    public class ViewProfile : Activity
    {
        public static MobileServiceClient Client =
        new MobileServiceClient("https://proevents.azurewebsites.net");

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

        private List<AttendingDB> table = new List<AttendingDB>();
        private ListView aListView;

        string username = pref.GetString("Username", "User");
        string userprofile = pref.GetString("Userprofile", "");
        UserProf prof;
        protected async override void OnCreate(Bundle savedInstanceState)
        {
```

```
            base.OnCreate(savedInstanceState);
            SetContentView(Resource.Layout.View_Profile);
            Toast.MakeText(this, "TEST " + userprofile, ToastLength.Short).Show();
            SetProfilePic();

            aListView = FindViewById<ListView>(Resource.Id.myListViewEP);

            var homeBTN = FindViewById(Resource.Id.btnHome);
            var eventsBTN = FindViewById(Resource.Id.btnEvents);
            var connectionsBTN = FindViewById(Resource.Id.btnConnections);
            var logout = FindViewById(Resource.Id.btnLogout);
            var viewBackBTN = FindViewById(Resource.Id.btnViewBack);
            var viewEventsBTN = FindViewById(Resource.Id.btnViewEvents);

            CurrentPlatform.Init();
            List<UserProf> ls = await Client.GetTable<UserProf>().ToListAsync();
            prof = ls.FirstOrDefault(x => x.Username == userprofile);
            TextView textName = (TextView)FindViewById(Resource.Id.txtViewName);
            textName.Text = "Hello, my name is " + prof.Firstname;
            textName.Text += " " + prof.Lastname;
            textName.SetBackgroundResource(Resource.Drawable.round_style);
            TextView textEmail = (TextView)FindViewById(Resource.Id.txtViewEmail);
            textEmail.Text = "You can contact me at " + prof.Email;
            textEmail.SetBackgroundResource(Resource.Drawable.round_style);
            TextView textCounty = (TextView)FindViewById(Resource.Id.txtViewCounty);
            textCounty.Text = "I am from " + prof.County;
            textCounty.SetBackgroundResource(Resource.Drawable.round_style);
            TextView textProfession =
(TextView)FindViewById(Resource.Id.txtViewProfession);
            textProfession.Text = "My main Interest is " + prof.Profession;
            textProfession.SetBackgroundResource(Resource.Drawable.round_style);
            TextView textDescription =
(TextView)FindViewById(Resource.Id.txtViewDescription);
            textDescription.Text = prof.Description;
            textDescription.SetBackgroundResource(Resource.Drawable.round_style);

            //string userid = prof.ID;

            homeBTN.Click += HomeProcess;
            eventsBTN.Click += EventProcess;
            connectionsBTN.Click += ConnectionsProcess;
            logout.Click += LogoutProcess;
            viewEventsBTN.Click += ViewEventsProcess;
            viewBackBTN.Click += BackViewProfile;
        }

        private void BackViewProfile(object sender, EventArgs e)
        {
            ISharedPreferences prefs =
Application.Context.GetSharedPreferences("UserInfo", FileCreationMode.Private);
            ISharedPreferencesEditor edit = prefs.Edit();
            edit.PutString("Username", username);
            edit.PutString("Userprofile", userprofile);
            edit.Commit();
            Finish();
        }

        private void ViewEventsProcess(object sender, EventArgs e)
        {
            ISharedPreferences prefs =
Application.Context.GetSharedPreferences("UserInfo", FileCreationMode.Private);
            ISharedPreferencesEditor edit = prefs.Edit();
```

```
            edit.PutString("Username", username);
            edit.PutString("Userprofile", userprofile);
            edit.Commit();
            //Toast.MakeText(this, "TEST " + userprofile, ToastLength.Short).Show();
            StartActivity(typeof(ViewEventsFollow));
        }

        private async void SetProfilePic()
        {
            List<FollowingUser> ls = await
Client.GetTable<FollowingUser>().ToListAsync();
            FollowingUser prof = ls.FirstOrDefault(x => x.FUsername == userprofile);
            string cat = prof.FProfession;
            string pPic = GetPic(cat);
            ImageView proPic = FindViewById<ImageView>(Resource.Id.ivViewProPic);
            Stream i = Assets.Open(pPic);
            Drawable draw = Drawable.CreateFromStream(i, null);
            proPic.SetImageDrawable(draw);
        }

        private string GetPic(string cat)
        {
            string userPic = cat;
            return "profilePic/" + userPic + ".png";
        }
        private void HomeProcess(object sender, EventArgs e)
        {
            Toast.MakeText(this, "Home Page", ToastLength.Short).Show();
            edit.PutString("Username", "User");
            StartActivity(typeof(MainProfile));
        }
        private void EventProcess(object sender, EventArgs e)
        {
            Toast.MakeText(this, "Events(soon)", ToastLength.Short).Show();
            edit.PutString("Username", "User");
            StartActivity(typeof(Events));
        }
        private void ConnectionsProcess(object sender, EventArgs e)
        {
            Toast.MakeText(this, "Your Connections", ToastLength.Short).Show();
            edit.PutString("Username", "User");
            StartActivity(typeof(Connections));
        }
        private void LogoutProcess(object sender, EventArgs e)
        {
            edit.Clear();
            edit.Apply();
            Toast.MakeText(this, "Logged Out", ToastLength.Short).Show();
            StartActivity(typeof(MainActivity));
        }
    }
}
```

WhoFollowEvent.cs

```
namespace ProEventsApp
{
    [Activity(Label = "WhoFollowEvent")]
    public class WhoFollowEvent : Activity
    {
```

```
        public static MobileServiceClient Client =
        new MobileServiceClient("https://proevents.azurewebsites.net");

        private List<UserProf> list;
        private List<AttendingDB> WFEItems;
        private List<AttendingDB> WFEItemsSP = new List<AttendingDB>();
        private ListView WFEListView;
        List<UserProf> prof = new List<UserProf>();

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

        string username = pref.GetString("Username", "User");
        string eventname = pref.GetString("EventName", "");

        protected async override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);

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

            SetContentView(Resource.Layout.WhoFollow_Event);

            WFEListView = FindViewById<ListView>(Resource.Id.myListViewWFE);

            var homeBTN = FindViewById(Resource.Id.btnHome);
            var eventsBTN = FindViewById(Resource.Id.btnEvents);
            var connectionsBTN = FindViewById(Resource.Id.btnConnections);
            var logout = FindViewById(Resource.Id.btnLogout);

            homeBTN.Click += HomeProcess;
            eventsBTN.Click += EventProcess;
            connectionsBTN.Click += ConnectionsProcess;
            logout.Click += LogoutProcess;
            CurrentPlatform.Init();
            IMobileServiceTable<AttendingDB> tableWFE = Client.GetTable<AttendingDB>();
            WFEItems = await tableWFE
                .Where(x => x.AEventName == eventname)
                .ToListAsync();

            int y = 0;
            while (y < WFEItems.Count)
            {
                string WFEListItem = WFEItems[y].AUsername;


                IMobileServiceTable<UserProf> allusers = Client.GetTable<UserProf>();
                List<UserProf> res = await allusers
                    .Where(x => x.Username.Contains(WFEListItem))
                    .ToListAsync();
                prof.Add(res[0]);
                y++;
            }
            ListViewAdapterDisplayUser adapterLV = new ListViewAdapterDisplayUser(this,
prof);
            WFEListView.Adapter = adapterLV;
            WFEListView.ItemClick += WFEListView_ItemClick;
```

```
            //sListView.ItemClick += EListView_ItemClick;
        }

        private async void WFEListView_ItemClick(object sender,
AdapterView.ItemClickEventArgs e)
        {
            var popup = new PopupMenu(this, WFEListView.GetChildAt(e.Position));
            CurrentPlatform.Init();
            //string UserNameFollow = WFEItems[e.Position].AUsername;
            IMobileServiceTable<UserProf> tableWFE = Client.GetTable<UserProf>();
            try {
            list = await tableWFE
                .Where(x => x.Username == WFEItems[e.Position].AUsername)//error here
                .ToListAsync();
            string followingUsername = list[e.Position].Username;
            string followingID = list[e.Position].userID;
            string followingFirst = list[e.Position].Firstname;
            string followingSecond = list[e.Position].Lastname;
            string followingProf = list[e.Position].Profession;
            string followingLocation = list[e.Position].County;
            string followingDes = list[e.Position].Description;
            popup.Inflate(Resource.Layout.WFE);
            popup.Show();
            popup.MenuItemClick += async (s, a) =>
            {
                switch (a.Item.ItemId)
                {
                    case Resource.Id.Follow:
                        FollowingUser followUser = new FollowingUser
                        {
                            Username = username,
                            FuserID = followingID,
                            FUsername = followingUsername,
                            FFirstname = followingFirst,
                            FLastname = followingSecond,
                            FProfession = followingProf,
                            FCounty = followingLocation,
                            FDescription = followingDes
                        };
                        await Client.GetTable<FollowingUser>().InsertAsync(followUser);
                        Toast.MakeText(this, "Success", ToastLength.Short).Show();
                        break;
                    case Resource.Id.Cancel:
                        Toast.MakeText(this, "Cancelled", ToastLength.Short).Show();
                        break;
                    default:
                        {
                            Toast.MakeText(this, "An error has occured, please try
again.", ToastLength.Short).Show();
                        }
                        break;
                }
            };
                }
            catch
            {
                Toast.MakeText(this, "TEST ", ToastLength.Short).Show();
            }

        }
        private void HomeProcess(object sender, EventArgs e)
        {
            Toast.MakeText(this, "Home Page", ToastLength.Short).Show();
            edit.PutString("Username", "User");
```

```
            StartActivity(typeof(MainProfile));
        }
        private void EventProcess(object sender, EventArgs e)
        {
            Toast.MakeText(this, "Events(soon)", ToastLength.Short).Show();
            edit.PutString("Username", "User");
            StartActivity(typeof(Events));
        }
        private void ConnectionsProcess(object sender, EventArgs e)
        {
            Toast.MakeText(this, "Your Connections", ToastLength.Short).Show();
            edit.PutString("Username", "User");
            StartActivity(typeof(Connections));
        }

        private void LogoutProcess(object sender, EventArgs e)
        {
            edit.Clear();
            edit.Apply();
            Toast.MakeText(this, "Logged Out", ToastLength.Short).Show();
            StartActivity(typeof(MainActivity));
        }
    }
}
```

Strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="app_name">ProEventsApp</string>
  <string name="county_prompt">Please select your county</string>
  <string-array name="county">
    <item>Antrim</item>
    <item>Armagh</item>
    <item>Carlow</item>
    <item>Cavan</item>
    <item>Clare</item>
    <item>Cork</item>
    <item>Derry</item>
    <item>Donegal</item>
    <item>Down</item>
    <item>Dublin</item>
    <item>Fermanagh</item>
    <item>Galway</item>
    <item>Kerry</item>
    <item>Kildare</item>
    <item>Kilkenny</item>
    <item>Laois</item>
    <item>Leitrim</item>
    <item>Limerick</item>
    <item>Longford</item>
    <item>Louth</item>
    <item>Mayo</item>
    <item>Meath</item>
    <item>Monaghan</item>
    <item>Offaly</item>
    <item>Roscommon</item>
    <item>Sligo</item>
    <item>Tipperary</item>
    <item>Tyrone</item>
    <item>Waterford</item>
    <item>Westmeath</item>
```

```
    <item>Wexford</item>
    <item>Wicklow</item>
  </string-array>
  <string name="event_prompt">Select Event Category</string>
  <string-array name="professions">
    <item>Computing</item>
    <item>Music</item>
    <item>Business</item>
    <item>Teaching</item>
    <item>Design</item>
    <item>Science</item>
    <item>Sports</item>
  </string-array>
</resources>
```

ConnMenu.cs

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
  <item android:id="@+id/ViewProfile" android:title="View" showAsAction="always"/>
  <item android:id="@+id/Unfollow" android:title="Unfollow" showAsAction="always"/>
</menu>
```

MyMenu.cs

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
  <item android:id="@+id/EditEvent" android:title="Edit Event" showAsAction="always"/>
  <item android:id="@+id/SendNotification" android:title="Send Notification(One Time
Only)" showAsAction="always"/>
  <item android:id="@+id/CancelEvent" android:title="Cancel Event"
showAsAction="always"/>
</menu>
```

SEMenu.cs

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
  <item android:id="@+id/Follow" android:title="Follow" showAsAction="always"/>
  <item android:id="@+id/Details" android:title="Details" showAsAction="always"/>
  <item android:id="@+id/Cancel" android:title="Cancel" showAsAction="always"/>
</menu>
```