

FrogFit

Version 3.0



DEVELOPER GUIDE

Texas Christian University

May 5, 2015

Revision History

The following is a history of document revisions.

Version	Changes	Edited
Version 1.0	Initial Draft	April 26, 2015
Version 2.0	Rewording and reformatting fixes	May 5, 2015
Version 3.0	Remove .0, wording grammar	May 5, 2015

Revision Sign Off

By signing the following, the team member asserts that he has read the entire document and has, to the best of knowledge, found the information contained herein to be accurate, relevant, and free of typographical errors.

Name	Signature	Date
Bryan Kribbs		
Geoffrey Adams		
Matt Ratliff		

Table of Contents

Revision History	I
Revision Sign Off	II
1 Introduction	1
1.1 Purpose.....	1
1.2 Section Overview	1
2 System Overview	2
2.1 System Components	2
2.2 Desktop Administration Portal	2
2.3 Mobile Application	2
2.4 REST API.....	2
2.5 Database.....	2
3 Getting Started	3
3.1 Web Administration Portal	3
3.2 Mobile Application	3
3.3 Database.....	3
4 Web Portal Development	4
4.1 Basic Layout.....	4
4.2 Home Screen	5
4.3 Create Profile	6
4.4 Athletes	6
4.5 Athlete Details	7
4.6 Gym FitScore	7
4.7 History	8
4.8 Leaderboards	8
4.9 Profile	9
4.10 Profile History	9
4.11 Settings	10
4.12 Login	10
4.13 Controllers.....	11
4.13.1 Home Controller	11
4.13.2 Data Controller	12
4.13.3 DB Controller.....	12
4.13.4 Math Controller.....	13
5 Mobile Application Development	14
5.1 Installation.....	14
5.1.1 Eclipse	14
5.1.2 Android SDK	14
5.1.3 Android SDK Packages	14
5.1.4 Project Setup.....	14
5.2 Activity\Fragment Layout	15
5.3 List Views \ Adapters.....	16
5.4 Layouts	16
5.5 Custom Drawables and Resources	17

5.6 MPCharts	18
6 Database Development	19
6.1 Basic Layout.....	19
6.2 Database Tables	19
6.2.1 Personnel Table	19
6.2.2 Coaches Table	19
6.2.3 Gym Table.....	19
6.2.4 History Table	19
6.2.5 Workouts Table	20
7 Definition of Terms	21

1 Introduction

1.1 Purpose

The purpose of this document is to guide new developers in installing and setting up FrogFit to expand features and continue development. Installation for each component of the system will be listed in their own individual sections.

1.2 Section Overview

Section 2 – System Overview – Defines the constraints associated with the design.

Section 3 – Getting Started – Resources needed to develop FrogFit.

Section 4 – Web Portal Development – Describes each item of the web portal.

Section 5 – Mobile Application Development – Describes each portion of the mobile applications development.

Section 6 – Database Development – Describes each portion of the database's development.

Section 7 – Glossary of Terms – Defines all the project-specific terms that are located throughout this document.

2 System Overview

2.1 System Components

FrogFit consists of four components: a desktop administrative portal, mobile application, REST API, and database. Read each component's description below.

2.2 Desktop Administration Portal

The desktop administration portal is built on an ASP.NET 5 framework with a responsive bootstrap front end. The administration portal is to be used only by coaches. The portal allows for coaches to easily manage and control their specific gym.

2.3 Mobile Application

The mobile application is made for Android devices only. It is built for Android operating systems Ice Cream Sandwich (API 15) and higher. Athletes are the only users of the Android app. The primary functions of the mobile application are to log workouts, view FitScores, view previous workouts, view gym leaderboards, and keep profiles up to date.

2.4 REST API

The REST API is the entry point for the communication between the Android application and the database. The API allows for information to be passed from the mobile application to the database in JSON form.

2.5 Database

Our database is the center of the entire system. It contains information for all the athletes within the gym, all of the workout data, and history data for all workouts completed by the gym. The database must be populated in order for the mobile application and web portal to function.

3 Getting Started

3.1 Web Administration Portal

In order to start development on the web portal, installation of Visual Studio 2013 or later is required. Visual Studio packages ASP.NET 5 framework, Chart.js, and Date Picker HTML Helper for MVC needs to be installed. All development work takes place within Visual Studio. Testing can be carried out over the web or through the debug provided by Visual Studio.

3.2 Mobile Application

Beginning development of the mobile application requires Eclipse to be installed onto a workstation. The Android plugin must be downloaded and installed from the Android development website. Development takes place within Eclipse and for testing purposes can be loaded onto the smart phone or on an emulator that is provided by Eclipse.

3.3 Database

To build the database, MSSQL Server 2012 needs to be installed onto a workstation. Also, install MSSQL Server Management Studio 2012 or later and refer to the database schema to begin database design. To receive any data needed for the database refer to Chalkbucket Labs.

4 Web Portal Development

4.1 Basic Layout

The Basic Layout of the website is responsive, and is made up of three parts. There is a top navigation bar that contains a home button on the left side and logout button on the right. The next portion is the side navigation bar, which is located on the left side of the webpage and contains two multi-level dropdown boxes and a single button. Each dropdown box contains links to the various pages and the single button links to the settings page. The final component to the layout is the page wrapper located in the center of the page, which contains each page's dynamic data. The layout can be accessed through the "Layout_Test" link under the "Shared Views" section.

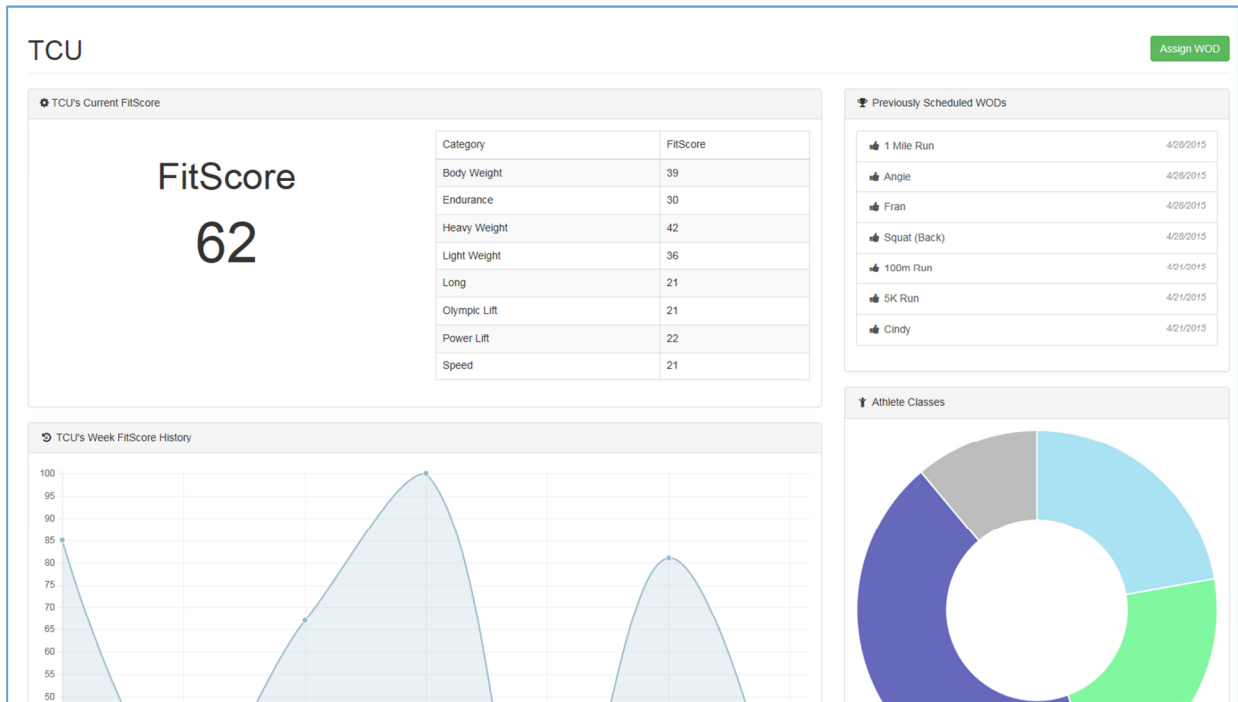
```

        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
    </button>
    @Html.ActionLink("FrogFit", "Index", "Home", new { area = "" }, new { @class = "navbar-brand" })
</div>
<ul class="nav navbar-top-links navbar-right">
    <li>
        Welcome, @Session["LoggedInUser"]
    </li>
    <li>
        @Html.ActionLink("Logout", "Login", "Home", new { area = "" }, new { @class = "navbar-link" })
    </li>
</ul>
<div class="navbar default sidebar" role="navigation" style="background-color:#60A7F9">
    <div class="sidebar-nav navbar-collapse">
        <ul class="nav" id="side-menu">
            <li>
                <a href="@Url.Action("Index","Home", new{ id = UrlParameter.Optional})">My Gym<span class="fa arrow"></span></a>
                <ul class="nav nav-second-level">
                    <li>
                        <a href="@Url.Action("CreateProfile","Home", new{ id = UrlParameter.Optional})"><i class="fa fa-user fa-fw"></i>Create/Edit Profile</a>
                    </li>
                    <li>
                        <a href="@Url.Action("LogWOD", "Home", new { id = UrlParameter.Optional })"><i class="fa fa-plus fa-fw"></i>Log A WOD</a>
                    </li>
                    <li>
                        <a href="@Url.Action("Athletes","Home", new{ id = UrlParameter.Optional})"><i class="fa fa-child fa-fw"></i>Athletes</a>
                    </li>
                    <li>
                        <a href="@Url.Action("GymFitScore","Home", new{ id = UrlParameter.Optional})"><i class="fa fa-check-circle-o fa-fw"></i>Gym's FitScore</a>
                    </li>
                    <li>
                        <a href="@Url.Action("History","Home", new{ id = UrlParameter.Optional})"><i class="fa fa-history fa-fw"></i>History</a>
                    </li>
                    <li>
                        <a href="@Url.Action("Leaderboards","Home", new{ id = UrlParameter.Optional})"><i class="fa fa-trophy fa-fw"></i>Leaderboards</a>
                    </li>
                </ul>
            </li>
            <li>
                <a href="@Url.Action("Profile","Home", new{ id = UrlParameter.Optional})">My Profile<span class="fa arrow"></span></a>
                <ul class="nav nav-second-level">
                    <li>
                        <a href="@Url.Action("Profile", "Home", new { id = UrlParameter.Optional })"><i class="fa fa-user fa-fw"></i>Profile</a>
                    </li>
                    <li>
                        <a href="@Url.Action("aHistory", "Home", new { id = UrlParameter.Optional })"><i class="fa fa-history fa-fw"></i>History</a>
                    </li>
                </ul>
            </li>
        </ul>
    </div>
</div>

```

4.2 Home Screen

The Home screen gives an overview of the gym information and statistics. It is made up of 4 different parts. There are four separate panels laid out in the page wrapper. The first shows the overall gym FitScores, the second shows the completed WODs for the week, the third chart is the week's FitScores, and the final panel shows the number of gym athletes in each class. To modify this page, use index.cshtml in the project solution.



4.3 Create Profile

The Create Profile screen is a basic form allowing gym owners to create athlete profiles. There are 13 different fields of basic personal information for the owners to complete. At the bottom of the page, there is an option to “Edit an Athlete”, which allows you to select a current athlete from a list and change their current information. To modify this page, use CreateProfile.cshtml in the project solution.

Athlete Profiles

First Name

Last Name

Email

Date of Birth

Gender Male Female

Address

City

State

Zip

Class

Home Phone

Cell Phone

Password

[Edit An Athlete](#)

4.4 Athletes

The Athletes page consists of one panel displaying a table of all the athletes currently working out in the gym. The name of each athlete is a link to the athlete’s detail page. The athletes table also displays the class, age and current FitScore of the athletes as well. To modify the athletes page use the athletes.cshtml page in the solution panel.

Athletes


⌵ Gym Athletes

Name	Class	Age	FitScore
Bryan Kribbs	Intermediate	23	57
Collin Duncan	Advanced	22	59
Geoff Adams	Beginner	39	67
Jim Owner	Beginner	31	60
John Doe	Masters	23	85
Kyle Johnson	Beginner	31	56
Marcus Beal	Beginner	24	60
Matt Ratliff	Advanced	27	64
Peter Day	Intermediate	28	52

4.5 Athlete Details

The Athlete Details page is a detailed page about each individual athlete. The details page contains personal information, FitScores (categorical and overall), and history of the selected athlete. This page is primarily for information only about the athlete, but this is the page where deleting the athlete will occur. To modify the information on this page use the AthleteDetails.cshtml file.

Bryan Kribbs
Delete Athlete

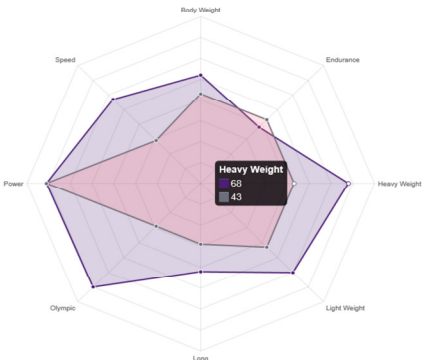


Age: 23 Last Workout Date: 4/28/2015

Gender: Male Last WOD Completed: Fran

Class: Intermediate Best Overall FitScore: 67

Bryan's FitScore



Overall FitScore: 57

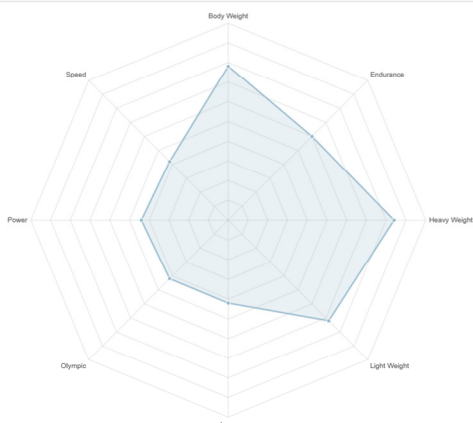
Category	FitScore	Confidence
Body Weight	52	43
Endurance	38	43
Heavy Weight	68	43
Light Weight	60	43
Long	42	29
Olympic Lift	70	29
Power Lift	71	71
Speed	57	29

4.6 Gym FitScore

The Gym FitScore page is a quick overview of the entire gym's FitScore. The gym has its own FitScore and this page illustrates their score. To modify this page, use GymFitScore.cshtml in the project solution.

TCU's FitScore

TCU's FitScores



TCU's Overall FitScore: 62

Category	FitScore
Body Weight	39
Endurance	30
Heavy Weight	42
Light Weight	36
Long	21
Olympic Lift	21
Power Lift	22
Speed	21

4.7 History

The history page allows gym owners to see all of the workouts that have ever been assigned. This page shows the WOD name, average, high and low scores of each workout. To modify this page, use History.cshtml in the project solution.

Gym WOD History				
Gym History				
Date	WOD	Average Score	High Score	Low Score
4/28/2015	1 Mile Run	59	59	59
4/28/2015	Angie	52	52	52
4/28/2015	Fran	57	57	57
4/28/2015	Fran	67	67	67
4/28/2015	Squat (Back)	55	55	55
4/21/2015	100m Run	68	68	68
4/21/2015	5K Run	96	96	96
4/21/2015	Cindy	95	95	95
4/20/2015	Pheezy	39	39	39
4/19/2015	Nate	55	55	55
4/16/2015	1 Mile Run	56	56	56
4/16/2015	10K Run	56	56	56
4/16/2015	10K Run	57	57	57
4/16/2015	2 Mile Run	60	60	60
4/16/2015	800m Run	46	46	46
4/16/2015	800m Run	60	60	60
4/16/2015	800m Run	61	61	61
4/16/2015	Angie	58	58	58
4/16/2015	Angie	60	60	60
4/16/2015	Annie	29	29	29

4.8 Leaderboards


The leaderboards page allows gym owners to see how all the athletes within the gym stack up against one another. The leaderboard ranks the athletes by their overall FitScore. To modify this page, use leaderboards.cshtml in the project solution.

Leaderboards		
Leaders of the Day		
Name	Date	FitScore
John Doe	4/14/2015	85
Geoff Adams	4/28/2015	67
Matt Ratliff	4/16/2015	64
Collin Duncan	4/28/2015	59
Bryan Kribbs	4/28/2015	57
Kyle Johnson	4/8/2015	56
Peter Day	4/28/2015	52

4.9 Profile

The profile page allows gym owners to view their own information. Gym owners may be athletes themselves, and this page allows them to see how they are performing. To modify this page, use profile.cshtml in the project solution.

Jim Owner



Age: 31

Gender: Male


Class: Beginner

Last Workout Date: 1/1/2001

Last WOD Completed: Get to work!

Best FitScore: 60

Jim's FitScore



Overall FitScore: 60

Category	FitScore	Confidence
Body Weight	0	0
Endurance	0	0
Heavy Weight	0	0
Light Weight	0	0
Long	0	0
Olympic Lift	0	0
Power Lift	0	0
Speed	0	0

4.10 Profile History

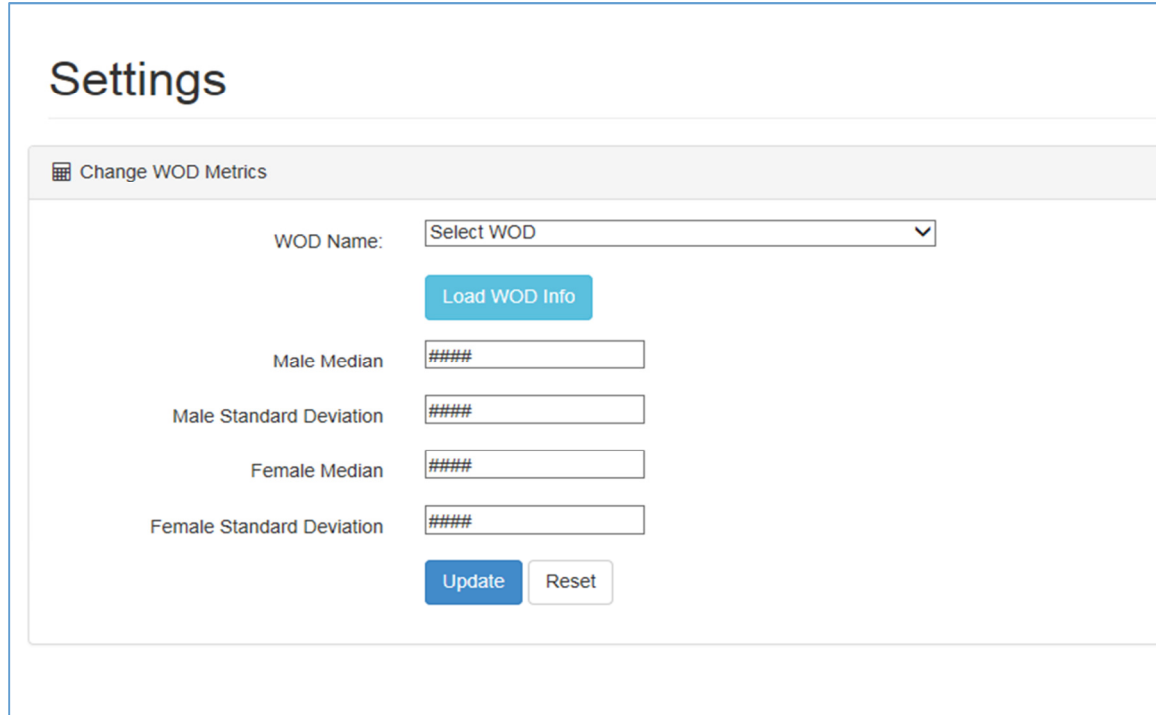
Profile History allows for gym owners to view the workouts they have performed themselves. To modify this page, use aHistory.cshtml in the project solution.

Jim's History

Date	WOD	Fitscore	Change in Score
1/1/2001	Get to work!	60	60

4.11 Settings

The settings page allows gym owners to modify the median and standard deviations for any WOD that can be performed. Changing the metrics to WODs allow owners to customize their gym's metrics. To modify this page, use settings.cshtml in the project solution.

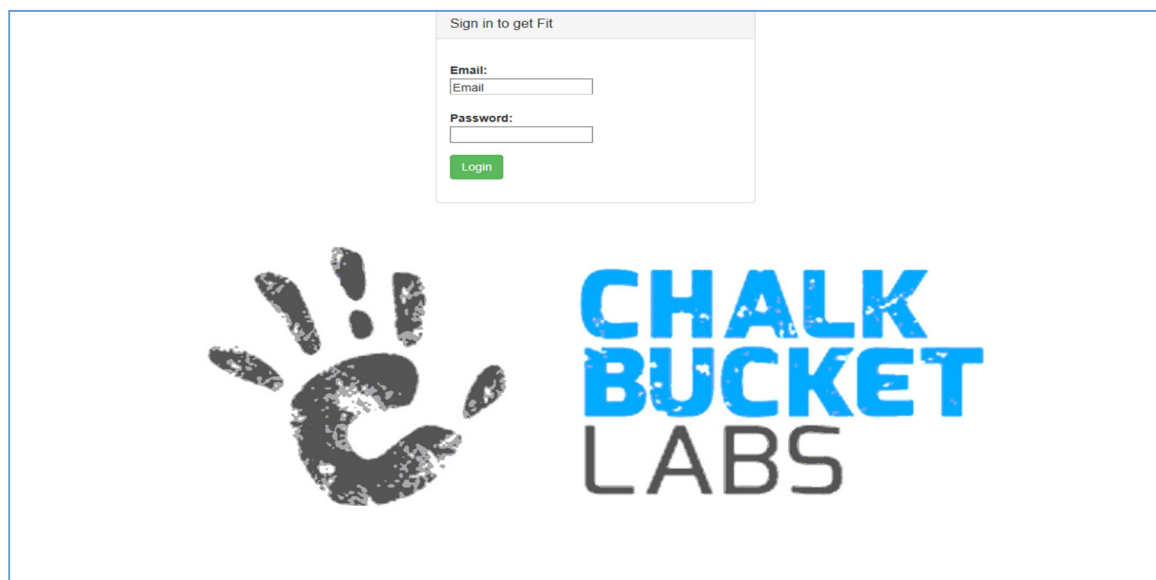


The screenshot shows a web page titled "Settings". Below the title is a section header "Change WOD Metrics" with a grid icon. The form contains the following elements:

- A "WOD Name:" label followed by a dropdown menu with the text "Select WOD" and a downward arrow.
- A blue button labeled "Load WOD Info" positioned below the dropdown.
- A "Male Median" label followed by a text input field containing "####".
- A "Male Standard Deviation" label followed by a text input field containing "####".
- A "Female Median" label followed by a text input field containing "####".
- A "Female Standard Deviation" label followed by a text input field containing "####".
- At the bottom of the form are two buttons: a blue "Update" button and a white "Reset" button.

4.12 Login

The login page allows gym owners to access the site. All gym owners need to do is insert their credentials correctly and they will gain access. To modify this page, use login.cshtml in the project solution.



The screenshot shows a login page with the following elements:

- A form titled "Sign in to get Fit" containing:
 - An "Email:" label followed by a text input field with "Email" inside.
 - A "Password:" label followed by a text input field.
 - A green "Login" button.
- Below the form is a large graphic featuring a black handprint on the left and the text "CHALK BUCKET LABS" on the right. "CHALK" and "BUCKET" are in blue, and "LABS" is in black.

4.13 Controllers

4.13.1 Home Controller

There are four main controllers to the site. The “HomeController” is the main controller that loads each page and is the hub for all the functionality within the site. Each page has its own method within the “HomeController”, and the name of the method correlates to the page.

```

{
  Oreferences
  public class HomeController : Controller
  {
    Oreferences
    public ActionResult Login()
    {
      return View();
    }
    Oreferences
    public ActionResult LogUserIn(Athletes ath)
    {
      if (ath.Email != null && ath.Password != null )
      {
        DBController db = new DBController();
        DataTable dt = db.LogUserIn(ath.Email);
        if (dt.Rows.Count == 0)
        {
          ModelState.AddModelError("", "The Login was Invalid, Try Again");
          return RedirectToAction("Login");
        }
        for (int i = 0; i < dt.Rows.Count; i++)
        {
          if(ath.Email.Equals(dt.Rows[i]["Email"].ToString()) && ath.Password.Equals(dt.Rows[i]["Password"].ToString())){
            Session["LoggedUserID"] = dt.Rows[i]["ID"].ToString();
            Session["LoggedInUser"] = dt.Rows[i]["Email"].ToString();
            return RedirectToAction("Index");
          }else{
          }
        }
        ModelState.AddModelError("", "The Login Credentials were incorrect");
        return View("Login", ath);
      }
      else
      {
        ModelState.AddModelError("", "The Login was Invalid, Try Again");
        return View("Login", ath);
      }
    }
    Oreferences
    public ActionResult Index()
    {
      DBController d = new DBController();
      MathController m = new MathController();
      int beginner = 0;
      int intermediate = 0;
      int advanced = 0;
      int master = 0;
      int[] gymCatScores = m.makeAllAverage();
      int GymFit = m.makeAverage();
      DataTable dt = d.getPreviousWODS();
      DataTable C = d.getCategoryNums();
    }
  }
}

```


4.13.2 Data Controller

The second controller is the “DataController”. The “DataController” handles all of the traffic from the mobile application to the database. Each method is either an HttpGet, or HttpPost method. The method type describes the action taking place.

```

References
public class DataController : ApiController
{
    [HttpGet]
    [ActionName("AllAthletes")]
    References
    public HttpResponseMessage GetAllAthletes()
    {
        var json = new JavaScriptSerializer();
        HttpResponseMessage response = Request.CreateResponse(HttpStatusCode.OK, "value");
        response.Content = new StringContent("Get all athletes method", Encoding.Unicode);
        return response;
    }

    [HttpGet]
    [ActionName("Personnel")]
    References
    public IEnumerable<string> GetAllPersonnel()
    {
        var json = new JavaScriptSerializer();
        DBController d = new DBController();
        DataTable dt = d.getPersonnel();
        List<string> data = new List<string>();
        for (int i = 0; i < dt.Rows.Count; i++)
        {
            data.Add(dt.Rows[i]["FName"].ToString() + " " + dt.Rows[i]["LName"].ToString());
        }
        return data;
    }

    [HttpGet]
    [ActionName("FitScore")]
    References
    public List<Scores> GetFitScore(string id)
    {
        JavaScriptSerializer serializer = new JavaScriptSerializer();

        DBController d = new DBController();
        DataTable dt = d.getFitScore(id);
        List<Scores> scoreData = new List<Scores>();
        DateTime date;
        foreach (DataRow row in dt.Rows)
        {
            //Console.WriteLine(row["DateCompleted"].ToString());
            //Debug.WriteLine(row["DateCompleted"].ToString());
            //Debug.WriteLine(row["DateCompleted"].ToString());
            date = (DateTime)row["DateCompleted"];
            scoreData.Add(new Scores
            {
                DateCompleted = date.ToShortDateString(),
                PID = Convert.ToInt32(row["PID"]),
                WID = (row["WID"].ToString()),
            });
        }
    }
}

```

4.13.3 DB Controller

The “DBController” handles all of the methods for the web portal. The methods within the “DBController” control the interactions between the database and the web portal. An example is the getAllWods method, which returns all the WODs within the database.

```

References
public class DBController
{
    //GET SECTION
    //
    //All SQL queries used to retrieve and return data from the
    //database are included in this block.

    //getAllWods
    //
    //input: none
    //output: none
    //
    //when called, returns a data table filled with all workout
    //names and IDs.
    References
    public DataTable getAllWods()
    {
        DataTable dt = new DataTable();
        using (SqlDataAdapter da = new SqlDataAdapter("SELECT DISTINCT CONVERT(VARCHAR(MAX), Name) AS Name, WorkoutID FROM workouts", new SqlConnection(ConfigurationManager.ConnectionStrings["bConnection"].ToString())))
        {
            da.Fill(dt);
        }
        return dt;
    }

    //getAthleteData
    //
    //input: athlete ID number
    //output: returns the entire row belonging to that athlete in
    //the database
    References
    public DataTable getAthleteData(int id)
    {
        DataTable dt = new DataTable();
        using (SqlDataAdapter da = new SqlDataAdapter(@"
        SELECT *, DATEDIFF(VV, DOB, GETDATE()) AS Age
        FROM Personnel
        WHERE ID = " + id + ""
        ", new SqlConnection(ConfigurationManager.ConnectionStrings["bConnection"].ToString())))
        {
            da.Fill(dt);
        }
        return dt;
    }

    //getAthleteHistory
    //
    //input: athlete ID number
    //output: returns a data table filled with the athlete's
    //complete history
    References
}

```

4.13.4 Math Controller

The math controller handles the math portion for both the mobile application and web portal. All of the FitScore calculations and confidence score calculations are performed in this class.

```

103 //
104 //Creates a new instance of the DBController to retrieve multiple
105 //DataTables containing various athlete, workout and history
106 //information. Calculations are then performed based on workout
107 //score, number of entries in history, and workout data.
108 1 reference
109 public Scores doWork()
110 {
111     //initialization of the DBController and DataTables
112     DBController d = new DBController();
113     DataTable dt1 = new DataTable();
114     DataTable dt2 = new DataTable();
115     DataTable dt3 = new DataTable();
116
117     //retrieving benchmark workout data
118     int gender;
119     gender = String.Compare(_w, "Male", true);
120     if (gender == 0)
121     {
122         dt2 = d.getValues(Convert.ToInt32(_v));
123         _median = dt2.Rows[0]["Median"].ToString();
124         _std = dt2.Rows[0]["StdDev"].ToString();
125     }
126     else
127     {
128         dt2 = d.getValues(Convert.ToInt32(_v));
129         _median = dt2.Rows[0]["FMed"].ToString();
130         _std = dt2.Rows[0]["FSD"].ToString();
131     }
132
133     //call fixScore and makeScore methods
134     fixScore();
135     makeScore();
136
137     //retrieving benchmark workout category
138     dt3 = d.getCategory(Convert.ToInt32(_v));
139     _category = dt3.Rows[0]["Category"].ToString();
140
141     //retrieving most recent instance of history based
142     //on athlete PID
143     dt1 = d.getScore(_pid);
144
145     //initialization of local variables
146     int Confidence;
147     int Fitness = 0;
148     int count = 0;
149     int body, endurance, heavy, light, longw, olympic, power, speed;
150     int min = 7;
151     int place = 0;
152     float tconf;
153
154     //basic string conversion to determine category

```

5 Mobile Application Development

5.1 Installation

5.1.1 Eclipse

Download Java JRE and JDK from java.com. Double click the Double-click on the downloaded file and follow the installation instructions. You do not need to register the JDK. Download Eclipse Luna from eclipse.org. There is no installation required for Eclipse. Just double click the eclipse.exe file.

5.1.2 Android SDK

For Windows, your download package is an executable file that starts an installer. The installer checks your machine for required tools, such as the proper Java SE Development Kit (JDK) and installs it if necessary. The installer then saves the Android SDK Tools to a specified the location outside of the Android Studio directories. Double-click the executable (.exe file) to start the install. Make a note of the name and location where you save the SDK on your system—you will need to refer to the SDK directory later when using the SDK tools from the command line. Once the installation completes, the installer starts the Android SDK Manager.

5.1.3 Android SDK Packages

As a minimum when setting up the Android SDK, you should download the latest tools and Android platform. Open the Tools directory and select: Android SDK Tools, Android SDK Platform-tools, Build-tools (highest version). Open the first Android X.X folder (the latest version) and select: SDK Platform, A system image for the emulator, such as ARM EABI v7a System Image. Once you've selected all the desired packages, continue to install by clicking Install Packages. Do not exit the SDK Manager.

5.1.4 Project Setup

To Install FrogFit in Eclipse, copy the FrogFit Mobile Application directory to a location on your computer (desktop).

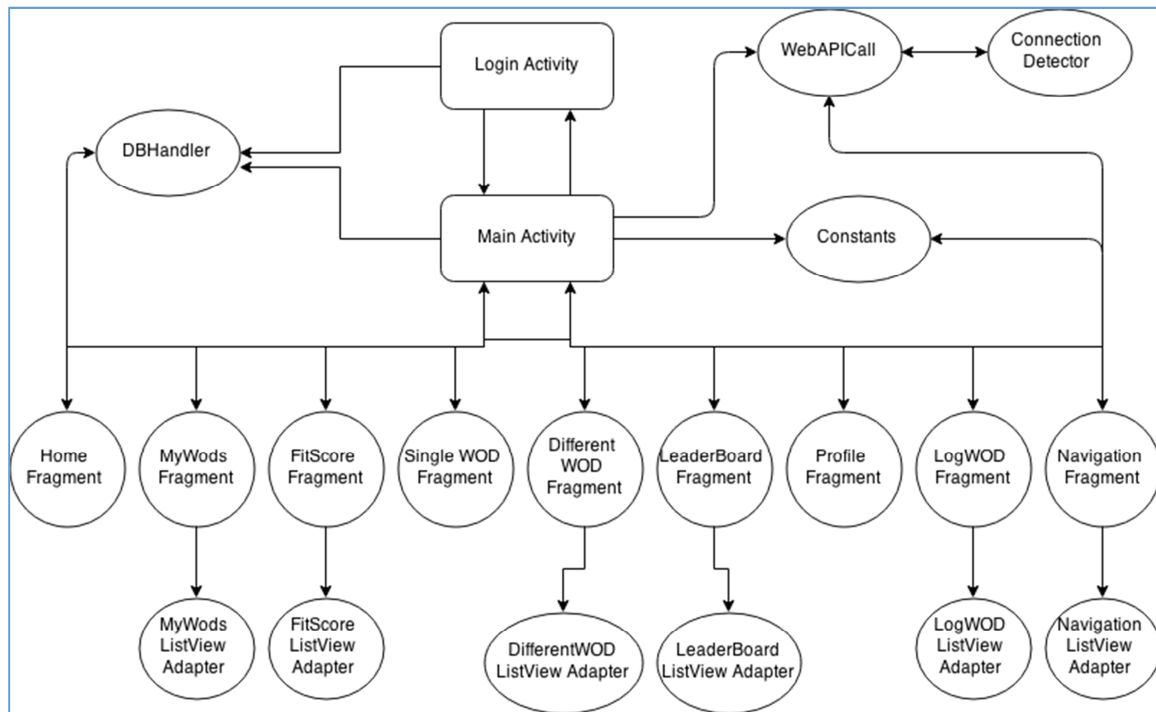
- Open Eclipse
- Click import > Android > Existing Android Code Into Workspace
- Be sure to click “Copy projects into workspace”
- Enjoy

5.2 Activity/Fragment Layout

The FrogFit application has two activities; Login activity which starts with the app and the Main Activity which controls interaction with all of the fragments. Upon login authentication, the user information is saved in the DB and upon logout, the DB entry is erased.

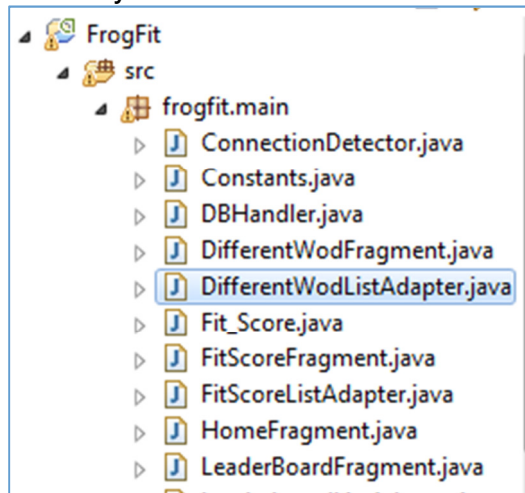
All screens are inflated by their respective fragments with the required information. This information comes from one of the static classes; DBHandler, WebAPICall, or Constants.

- The DBHandler contains the personal information for the logged in athlete.
- The Constants class contains information for the web address and column name constants used in the custom list view adapters.
- The WebAPICall gets information from the REST API.
- The Connection Detector is just used to make sure the web is available to help determine which error screen is used.



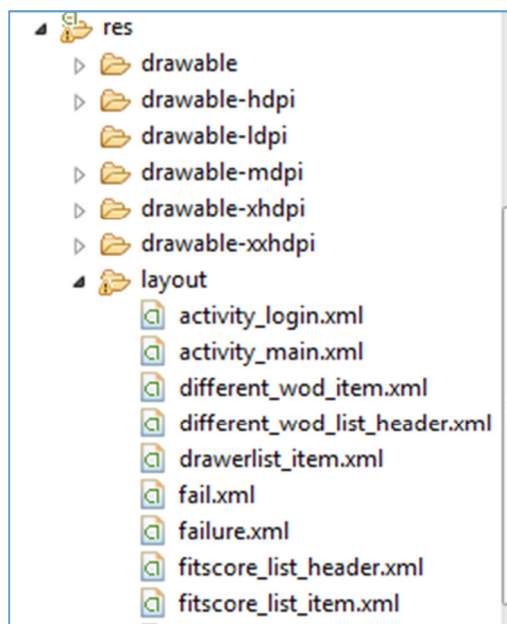
5.3 List Views \ Adapters

ListView is a view group that displays a list of scrollable items. The list items are automatically inserted to the list using an Adapter that pulls content from a WebAPICall and converts each item result into a view that's placed into the list. Each of the ListViews has its own adapter since each has a different formatting or number of columns which is controlled by the layout which is called from the ListView adapter. The ListView adapters are in the FrogFit/src/frogfit.main/ directory.



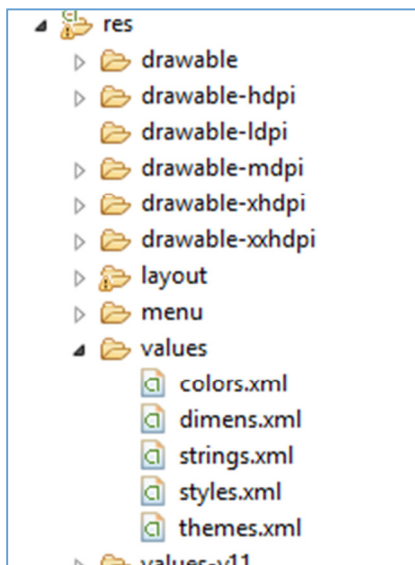
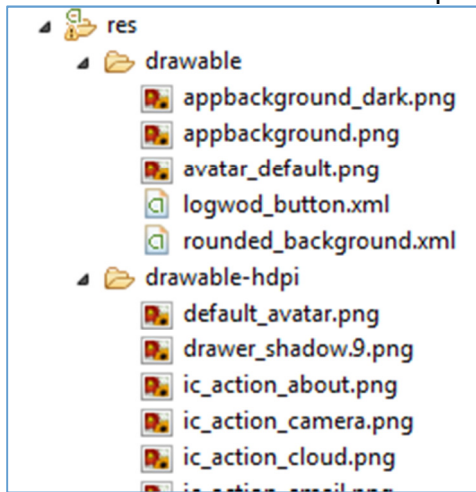
5.4 Layouts

A layout defines the visual structure for a user interface of each screen and each list. Each activity or fragments inflates a layout. The layouts are listed in FrogFit/res/layout/ directory.



5.5 Custom Drawables and Resources

FrogFit has custom icons, buttons, values (colors), and custom layouts. Each of these can be found in their respective directories in FrogFit/res/.

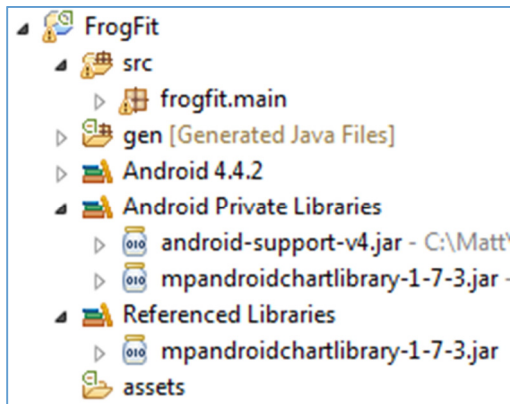


5.6 MPCharts

FrogFit uses a graphing library developed by PhilJay (<https://github.com/PhilJay>) called MPAndroidChart. MPAndroidChart is “MPAndroidChart is a powerful & easy to use chart library for Android, supporting line-, bar-, scatter-, candlestick-, bubble-, pie- and radarcharts (spider web), as well as scaling, dragging (panning), selecting and animations. Works on **Android 2.2 (API level 8)** and upwards.” FrogFit is just using leveraging the radar chart and populating the athletes categorical FitScore and categorical confidence scores.

To install MPAndroidChart,

- Download the latest .jar file from the releases section
- Copy the mpandroidchartlibrary-1-7-3.jar file into the libs folder of your Android application project
- Start using the library



```
// Create variables to populate or change data.
listView = (ListView) rootView.findViewById(R.id.fitscore_listView);
mChart = (RadarChart) rootView.findViewById(R.id.radar_chart);
fitScoreNum = (TextView) rootView.findViewById(R.id.fitscore_fitscore_num);
scoreVals = new ArrayList<Entry>();
confVals = new ArrayList<Entry>();
db = new DBHandler(getActivity(), null, null, Constants.DB_VERSION);
int tempnum;
```

6 Database Development

6.1 Basic Layout

The FrogFit CFData database is built and designed on a Microsoft SQL Server using typical table and column structure. The CFData database has also been optimized and refined to provide the simple and fast interaction between the REST API and itself.

6.2 Database Tables

6.2.1 Personnel Table

The Personnel table holds all relevant information for an individual at a gym, whether an athlete or a coach. This includes basic information that contains, but is not limited to, address, city, date of birth, name, gender, etc. The personnel table also contains the auto-incrementing primary key ID which is referenced in several other tables such as History and Coaches.

6.2.2 Coaches Table

The Coaches table holds all information relevant to being a coach at a gym. This includes start date, coach title and the coaches ID. Having an entry in this table enables access to the coach's web portal.

6.2.3 Gym Table

The Gym table holds all relevant information for a given gym. This table closely mimics the Personnel table in that it treats the gym as it were an individual athlete. The columns in this table include, but are not limited to, address, email, fax, phone, name, etc. This table is used to populate any gym-relevant information found on the website or mobile application.

6.2.4 History Table

The History table handles and holds each instance of every completed workout. Any time an athlete completes a workout, a new row is inserted into the History table with the athlete's ID number and a unique ID tag (track) for that workout. By referencing this table, any information about a workout done in the past can be retrieved and displayed for the user. This table also holds instances of each categorical score, confidence scores and FitScore which updates each time a new workout is performed.

6.2.5 Workouts Table

The Workouts table holds all relevant workout information. This includes but is not limited to, workout name, category, median and standard deviation. The workouts table also contains columns which take care of much smaller details. The WODD column is updated to contain a specific date for which that workout is to be performed. The workout table also contains unique male/female median and standard deviation data as well as an index to keep track of whether the workout is scored by time or repetition.

7 Definition of Terms

Administrative Web Portal - Internet web page that gym owners\managers use to update the content on their FrogFit site.

Android - An open-source operating system used on mobile devices.

ASP.NET MVC – Open source web application framework used to create applications.

Benchmark Workout – Workout designed to measure an athlete’s performance and improvements through repeated and irregular appearances in a workout of the day. Benchmark workouts can be in one of the categories; Girls, Heroes, Notables.

Categories – There are eight fitness categories that are considered when creating an overall fitness score.

- Body Weight
- Endurance
- Heavy Weight
- Light Weight
- Long
- Olympic Lift
- Power Lift
- Speed

Class – Classification system for athletes. The classifications include Beginner, Intermediate, Advanced, and Master.

CoRD – A remote desktop application used on Macs.

Crossfit - Strength and conditioning program that is by design broad, general, and inclusive.

Microsoft SQL Server - A relational database system designed for business or enterprise environments.

Remote Desktop Connection – One of the components of Microsoft Windows that allows users to take control of a remote computer or virtual device over a network connection.

REST – (REpresentational State Transfer) is an architectural style that is used in the development of light-weight Web services.

Rx – When a workout is completed “as prescribed” by the coach. This includes doing the correct movements, weight, and rounds. Also known as, Rx’d.

Web API – An application programming interface for both the webserver and web browser that allows retrieval of structured data.

WinSCP – A secure FTP client for Windows with features that include FTP, SSL/TLS, SFTP via SSH, and HTTP/HTTPS support.

WOD – (Workout of the Day) A set of movements that athletes will complete for a given day.