Irbid National University
Faculty of Information Technology
Computer Science

VIRTUAL COACH

Simulation Project to Attainment a degree in Computer Science

Students:

Hasan Mahmoud Eskandarani
201120048

Ali Yousef Nawashy
201120160

Supervised by: Dr.Motassem Abu Dawwas

Semester: 2015/2014 - 2

ACKNOWLEDGEMENTS

This project is made possible through the help and support from everyone, including: parents, teachers, family, friends, and in essence, all sentient beings. Especially, please allow us to dedicate my acknowledgment of gratitude toward the following significant advisors and contributors:

First, we would like to thank Dr .Motassem Abu Dawwas for his most support and encouragement. He kindly followed us through this project and offered invaluable detailed advices on many critical aspects.

Second, we would like to thank Dr. Ahmad odat and Dr. Radwan Baitha i for the kind courses in coumpter Scince and Web design.

Finally, we sincerely thank to our parents, family,My Wife , and friends, who provide the advice and financial support.

The application of this project would not be possible without all of them.

ABSTRACT

For this project, we propose and implement a system that can be used to simulate a private coach using the Kinect v2 sensor connected to a windows store application, which allows players to check their exercises just using a computer without the need to be in a gym club, and providing the ability to track his progress history to help the player archiving the most fitness benefits.

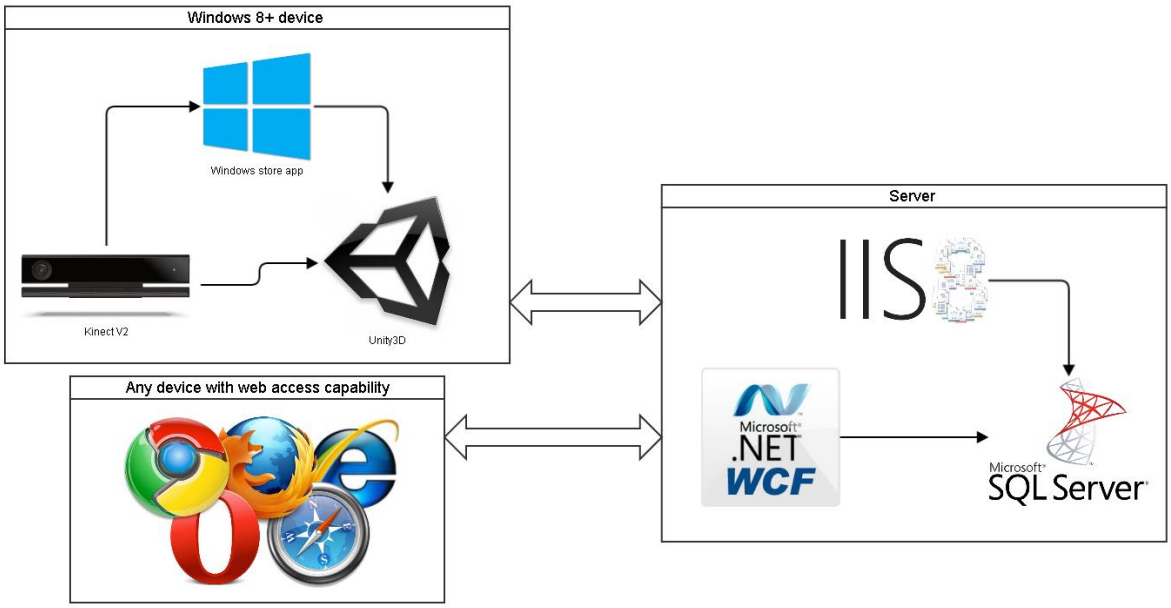
Table of Contents

Contents

Introduction

With the modern and busy life we all have, sometimes we can't commit going to gym on time, for that we are providing a new fitness service that allows the players to play his fitness schedule at any time where ever he is. To accomplish that we considered the needs of players according to the requirements we gathered them previously from many gyms we visited. The main needs was scheduling their fitness programs, tracking and correct the particular movement for each exercise using a special sensor.

Figure 1 system architecture diagram



1 Hardware section

1.1 The Kinect Device

1.1.1 Definition:

is a line of motion sensing input devices by Microsoft for Xbox 360 and Xbox One video game consoles and Windows PCs. Based around a webcam-style add-on peripheral, it enables users to control and interact with their console/computer without the need for a game controller, through a natural user interface using gestures and spoken commands. The first-generation Kinect was first introduced in November 2010 in an attempt to broaden Xbox 360's audience beyond its typical gamer base. A version for Windows was released on February 1, 2012. Kinect competes with several motion controllers on other home consoles, such as Wii Remote Plus for Wii and Wii U, PlayStation Move/PlayStation Eye for PlayStation 3, and PlayStation Camera for PlayStation 4.

The Kinect History:

Kinect was first announced on June 1, 2009 at E3 2009 under the code name "Project Natal". Following in Microsoft's tradition of using cities as code names, "Project Natal" was named after the Brazilian city of Natal as a tribute to the country by Brazilian-born Microsoft director Alex Kipman, who incubated the project. The name Natal was also chosen because the word natal means "of or relating to birth", reflecting Microsoft's view of the project as "the birth of the next generation of home entertainment".

Three demos were shown to display Kinect when it was revealed at Microsoft's E3 2009 Media Briefing: Ricochet, Paint Party and Milo & Kate. A demo based on Burnout Paradise was also shown outside of Microsoft's media briefing. The skeletal mapping technology shown at E3 2009 was capable of simultaneously tracking four people, with a feature extraction of 48 skeletal points on a human body at 30 Hz.

It was rumored that the launch of Project Natal would be accompanied with the release of a new Xbox 360 console (as either a new retail configuration, a significant design revision and/or a modest hardware upgrade). Microsoft dismissed the reports in public and repeatedly emphasized that Project Natal would be fully compatible with all Xbox 360 consoles. Microsoft indicated that the company considers it to be a significant

initiative, as fundamental to Xbox brand as Xbox Live, and with a launch akin to that of a new Xbox console platform. Kinect was even referred to as a "new Xbox" by Microsoft CEO Steve Ballmer at a speech for Executives' Club of Chicago. When asked if the introduction will extend the time before the next-generation console platform is launched (historically about 5 years between platforms), Microsoft corporate vice president Shane Kim reaffirmed that the company believes that the life cycle of Xbox 360 will last through 2015 (10 years).

Figure 2 Testing Kinect V1 + Kinect V2 in our Lab

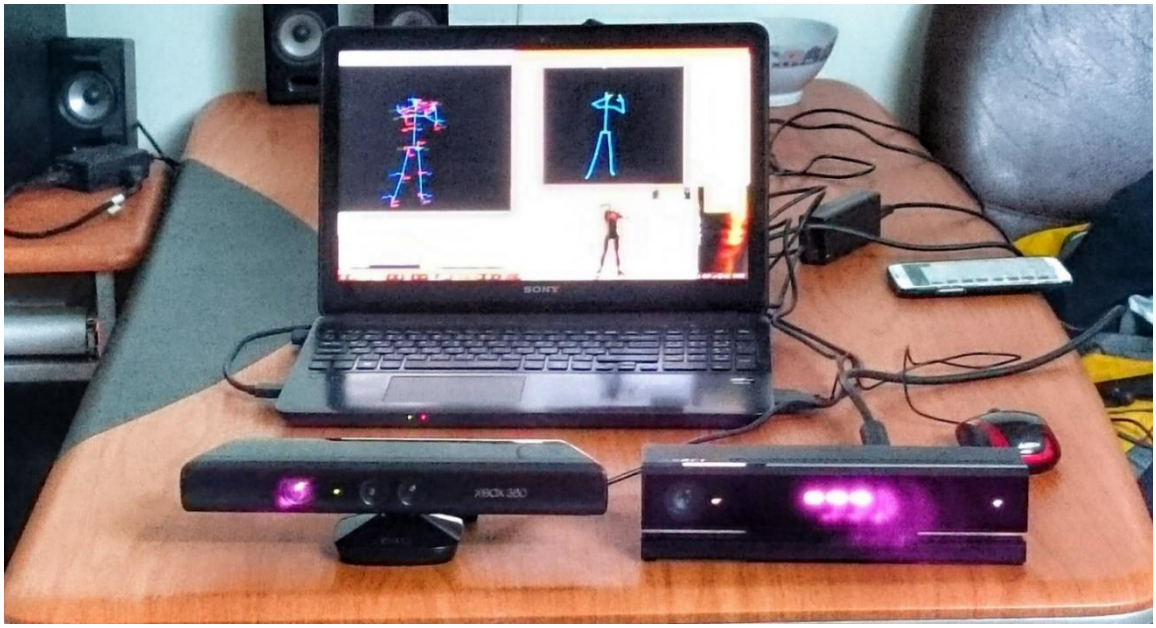
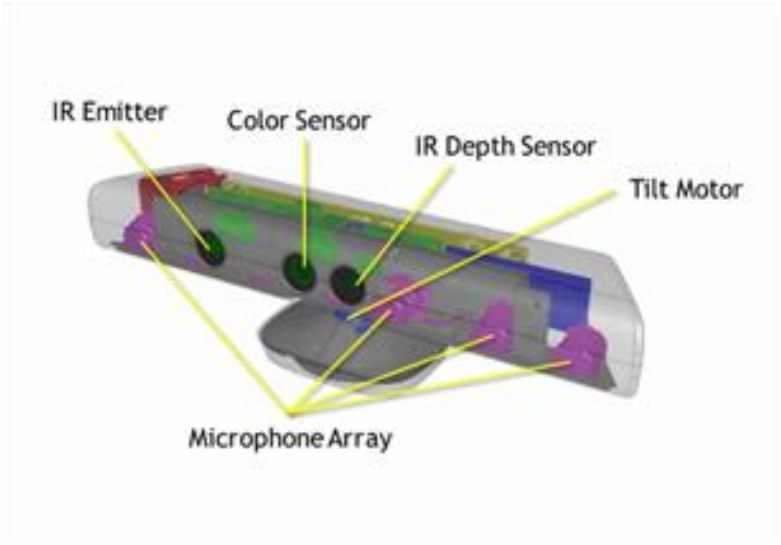


Figure 3 kinect sensor parts



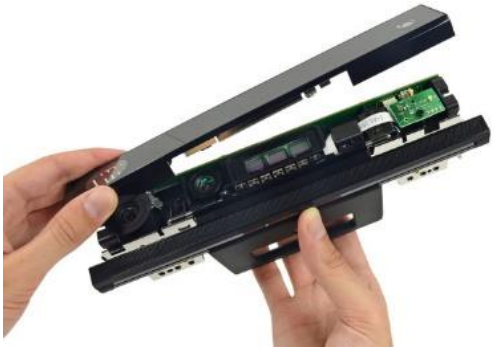
1.2 Kinect v1 VS Kinect V2

	Kinect V1	Kinect V2
Color Camera	640 x 480 @30fps	1920 x 1080 @30fps
Depth Camera	320 x 240	512 x 424
Maxx Depth Distance	~4.5M	~4.5M
Min Depth Distance	40 cm in near mode	50 cm
Horizontal Field of View	57 degrees	70 degrees
Vertical Filed of View	43 degrees	60 degrees
Tilt Motor	yes	No
Skeleton Joints Defined	20 joint	26 joint
Full Skeleton Tracked	2	6
USB Standard	2.0	3.0
Supported OS	Windows 7, Windows 8	Windows 8

Table 1 Kinect versions specification comparison

Figure 5 Kinect V1 disassembly

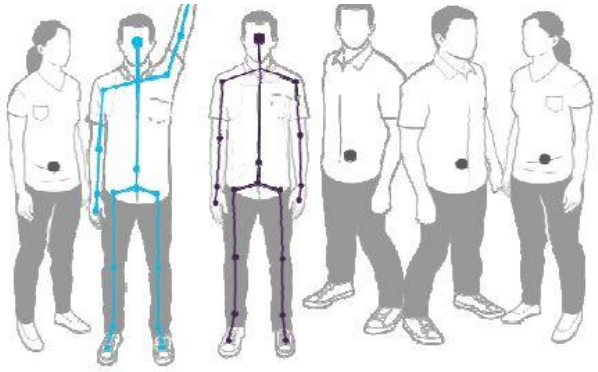
Figure 4 Kinect V2 disassembly



1.3 Kinect SDK

1.3.1 Skeletal Tracking:

Figure 6 Kinect can recognize six people and track two



An application can locate the joints of tracked users in space and track their movements over time.

Skeletal tracking is optimized to recognize users standing or sitting, and facing the Kinect, Sideways poses provide some challenges regarding the part of the user that is not visible to. To be recognized, users simply need to be in front of the sensor, making sure the sensor can see their head and upper body. No specific pose or calibration action needs to be taken for users to be tracked.

1.3.2 Field of View:

Kinect field of view of the users is determined by the settings of the IR camera, which are set with the DepthRange Enumeration.

Depth data ranges, which determine how close to (or far from) the sensor a person can be and still be in the field of view.

Figure 7 Kinect vertical Field of View in default range

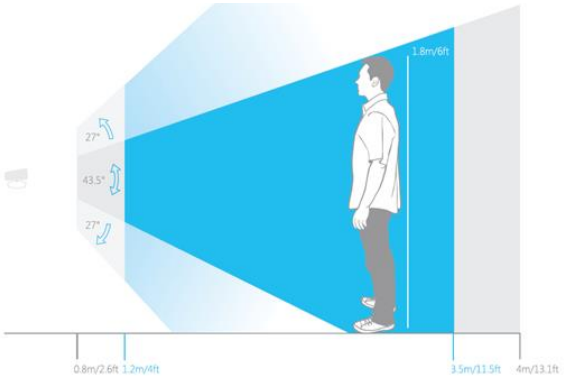
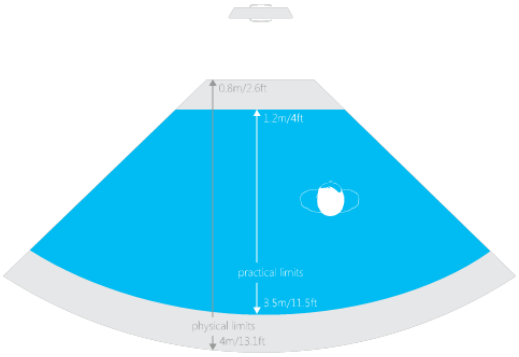


Figure 8 Kinect horizontal Field of View in default range

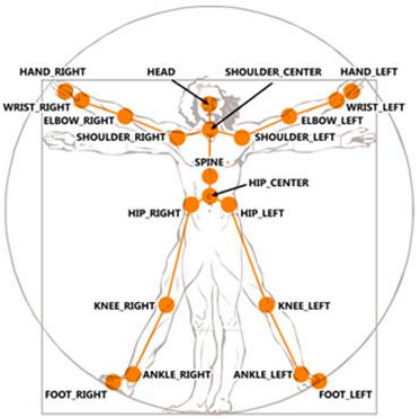


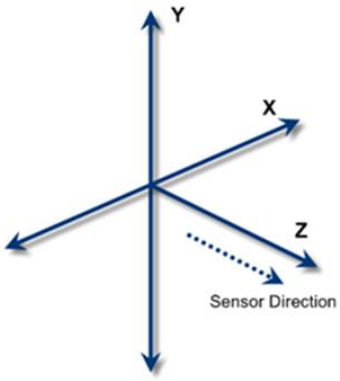
1.3.3 Depth Reference:

- Distance Range: 850mm to 4000mm range
- Depth value 0 means unknown (Shadows, low reflectivity, and high reflectivity among few reasons)
- Player index
    - 0 : no player
    - 1: Skeleton 0
    - 2: Skeleton 1
    - …

1.3.4 Joints:

- Maximum 6 players tracked at once
- Each player with a set of <x,y,z> joints positions
- Each joint has associated state :
    - Tracked.
    - Not Tracked.
    - Inferred.
- Inferred – Occluded, Clipped, or low confidence joints
- Not Tracked – Rare, but the code must check for this state





2 Software Analysis

2.1 Functional Requirements:

This system provide service for each user has logged in:

R1. This system would be able to provide a profile for each player

- R1.1 add exercise to his program
- R1.2 add new program
- R1.3 modify profile details
- R1.4 modify program
- R1.5 play exercise
  - R1.5.1 play exercise in tracking mode
  - R.1.5.2 play exercise without tracking mode
  - R1.5.3 play exercise with 3D view
  - R1.5.4 play exercise without 3D view
- R1.6 Select avatar to player
- R1.7 Select avatar to Coach of player

R2. This system would be able to provide a page for each admin (coach)

R2.1 add new exercise to the system

R2.1.1 add counts of exercise

R2.1.2 add image for exercise

R2.1.3 add details for exercise

R2.1.4 add YouTube link for exercise

R2.2 edit exercise details

R2.3 Delete exercise from the system

R2.4 Add program to the system

R2.5 Delete program from system

R3. Edit user permission

R4: Record the exercise

2.2 Non-Functional Requirements:

2.2.1 Performance Requirement:

The software able to support one user.

2.2.2 Safety Requirements

The system must encrypt the data using some encryption algorithm from the database for transmission from point to point

Data should be backed up every quarter hour

2.2.3 Security Requirements

More than three attempts al login and failure will produce a red flag to the system administrator report.

Data should be secured.

2.2.4 Software Quality Attributes

The system interfaces must follow design conventions, which allow for familiar location of drop down menus, help, report an error, etc.

Input errors will be returned a user-friendly message.

2.3 Use Cases:

2.3.1 Use Case specification:

ID		***S01***
Name		*Login*
Description		*Authentication checking*
Primary actors		*User*
Secondary actors		*Admin*
Flow of event	*Actors*	*System response*
		The system show a login dialog
	The user enter his Username and his Password	
		check the data entered
		show welcome screen to the user
Pre-Condition		-
Post-Condition		*Show Profile of the user*

ID		***S02***
Name		*Sign up*
Description		*Registration For new users*
Primary actors		*User*
Secondary actors		*Admin*
Flow of event	*Actors*	*System response*
		The system show a Sign up dialog
	The user enter his Username and his Password	
		check the data entered
		show Registered message to the user
Pre-Condition		-
Post-Condition		*Show Information Form for new user*

ID	***S03***

Name	*Add Exercise*	
Description	*Allowed to user to add exercises in current days*	
Primary actors	*User*	
Secondary actors	-	
Flow of event	*Actors*	*System response*
	1- Add Plan form	
	2- Choose day	
		3-Show all exercises that can user add it
	4-Add counts of exercises	
		Save exercises
Pre-Condition	*Login*	
Post-Condition	*Show new exercises that user added*	

ID	***S04***

Name	*Add New Program to current user*	
Description	*Allowed to user to add his own program*	
Primary actors	*User*	
Secondary actors	-	
Flow of event	*Actors*	*System response*
	1- Go to Add Plan form	
	2- Choose exercises	
	3- Add program	
		4-Show exercises of new program that user added
		5-New Program saved
Pre-Condition	*Login*	
Post-Condition	*Show new program that user added*	

ID ***S05***

Name		*Logout*
Description		*Allowed user to logout from website*
Primary actors		*User*
Secondary actors		*Admin*
Flow of event	*Actors*	*System response*
	1-logout	
		2- Exit user from system
Pre-Condition		*Login*
Post-Condition		*User logged out*

ID	***S06***

Name	*Edit Profile information*	
Description	*Allowed to user to edit his own information*	
Primary actors	*User*	
Secondary actors	-	
Flow of event	*Actors*	*System response*
	1- Go to setting form	
	2- Edit information	
		3- Save new data
Pre-Condition	*Login*	
Post-Condition	*New information has been saved*	

ID ***S07***

Name		*Add Program*
Description		*Add new Program to system*
Primary actors		*Admin*
Secondary actors		-
Flow of event	*Actors*	*System response*
	1- Add exercises	
	2- Add program	
		3- Save new data
Pre-Condition		*Login*
Post-Condition		*New Program has been saved*

ID	***S09***

Name	*Edit Program for current user*	
Description	*Allowed to user to Edit his own program*	
Primary actors	*User*	
Secondary actors	-	
Flow of event	*Actors*	*System response*
	1- Edit program	
		2- Edit information program
		3- Data saved
Pre-Condition	*Login*	
Post-Condition	*Program of current user has been updated*	

ID ***S10***

Name		*Delete Program*
Description		*Delete Program from system*
Primary actors		*Admin*
Secondary actors		-
Flow of event	*Actors*	*System response*
	1- Choose program	
	2- Delete program	
		3- Save new data
Pre-Condition		*Login*
Post-Condition		*Program has been saved*

ID ***S11***

Name		*Delete exercise*
Description		*Delete exercise from to system*
Primary actors		*Admin*
Secondary actors		-
Flow of event	*Actors*	*System response*
	1- Choose exercise	
	2- Delete exercise	
		3- Save data
Pre-Condition		*Login*
Post-Condition		*Exercise has been removed*

ID		***S12***
Name		*Edit exercises*
Description		*Edit exercise in the system*
Primary actors		*Admin*
Secondary actors		-
Flow of event	*Actors*	*System response*
	1- Choose exercise	
	2- Edit exercise information	
		3- Save data
Pre-Condition		*Login*
Post-Condition		*Exercise has been updated*

ID ***S13***

Name		*Edit user permission*
Description		*Edit user permission in the system*
Primary actors		*Admin*
Secondary actors		-
Flow of event	*Actors*	*System response*
	1- Choose user	
	2- Edit user permission	
		3- Save data
Pre-Condition		*Login*
Post-Condition		*User permission has been updated*

ID ***S14***

Name		*Paly exercise*
Description		*Allow to user to play an exercise with/without 3D and with/without Kinect*
Primary actors		*User*
Secondary actors		*Admin*
Flow of event	*Actors*	*System response*
	1- Select 3D mode (on/off)	
	2-Select Tracking mode (on/off)	
	3- Select exercise	
		4-Invoke Unity 3D and pass the parameter
Pre-Condition		*Login*
Post-Condition		*Assign "Done" to the database*

ID ***S16***

Name		*Stop record exercise*
Description		*Allow to Coach(Admin) to stop record when finished recording section*
Primary actors		*Admin*
Secondary actors		-
Flow of event	*Actors*	*System response*
	1- Stop record	
		2-Create XML file
		3-Submit XML file
Pre-Condition		*Login*
Post-Condition		*Show message box*

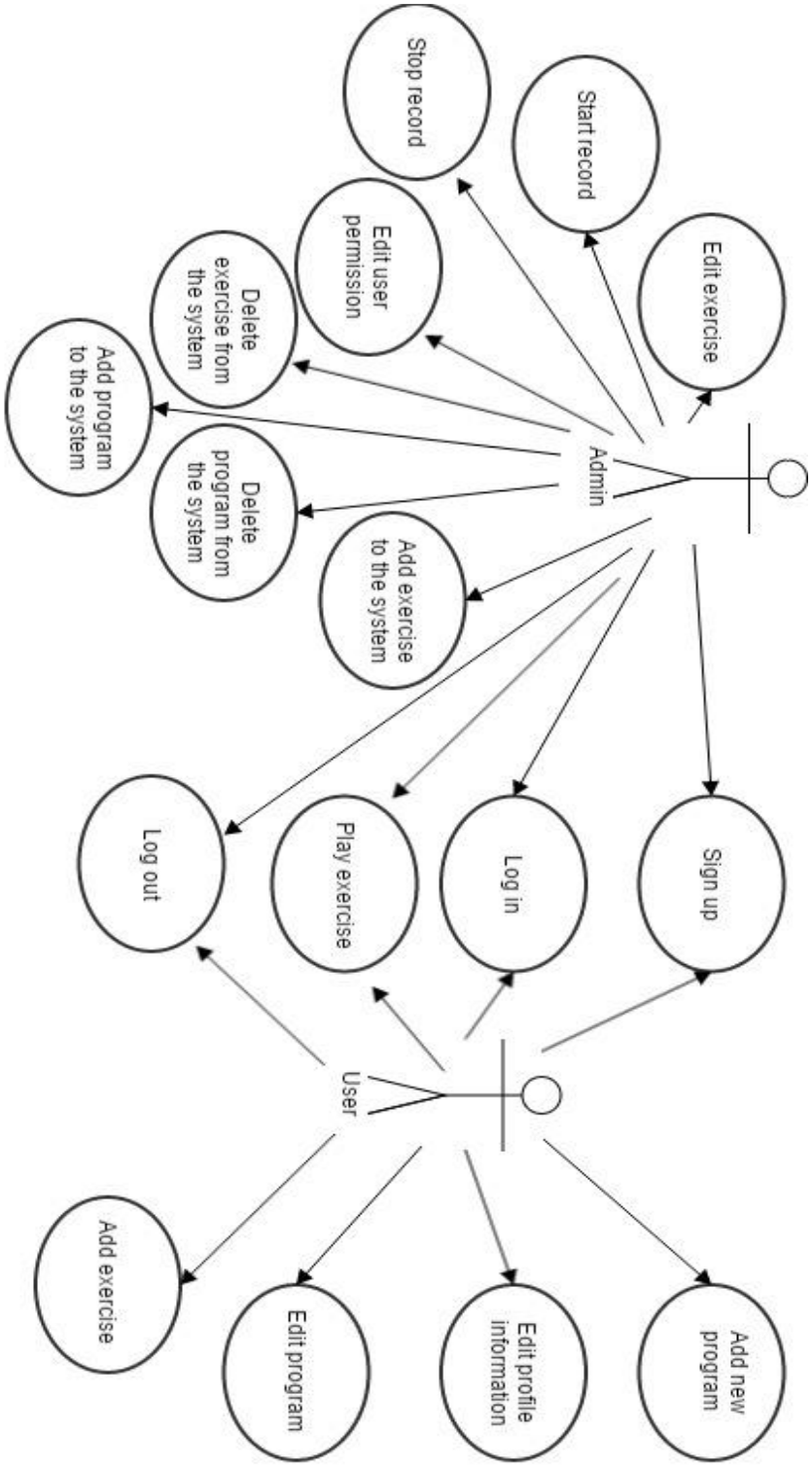
ID		***S08***
Name		*Add exercise*
Description		*Add new exercises to system*
Primary actors		*Admin*
Secondary actors		-
Flow of event	*Actors*	*System response*
	1- Add exercises	
		2- Save new data
Pre-Condition		*Login*
Post-Condition		*New exercises has been saved*

ID ***S15***

Name		*Start record exercise*
Description		*Allow to Coach(Admin) to submit new exercise to the system*
Primary actors		*Admin*
Secondary actors		-
Flow of event	*Actors*	*System response*
	1- Input name	
		2- Check name
	3- Start record	
		4-Show preview
		5-Wait for stop record
Pre-Condition		*Login*
Post-Condition		*Start record and show Stop record*

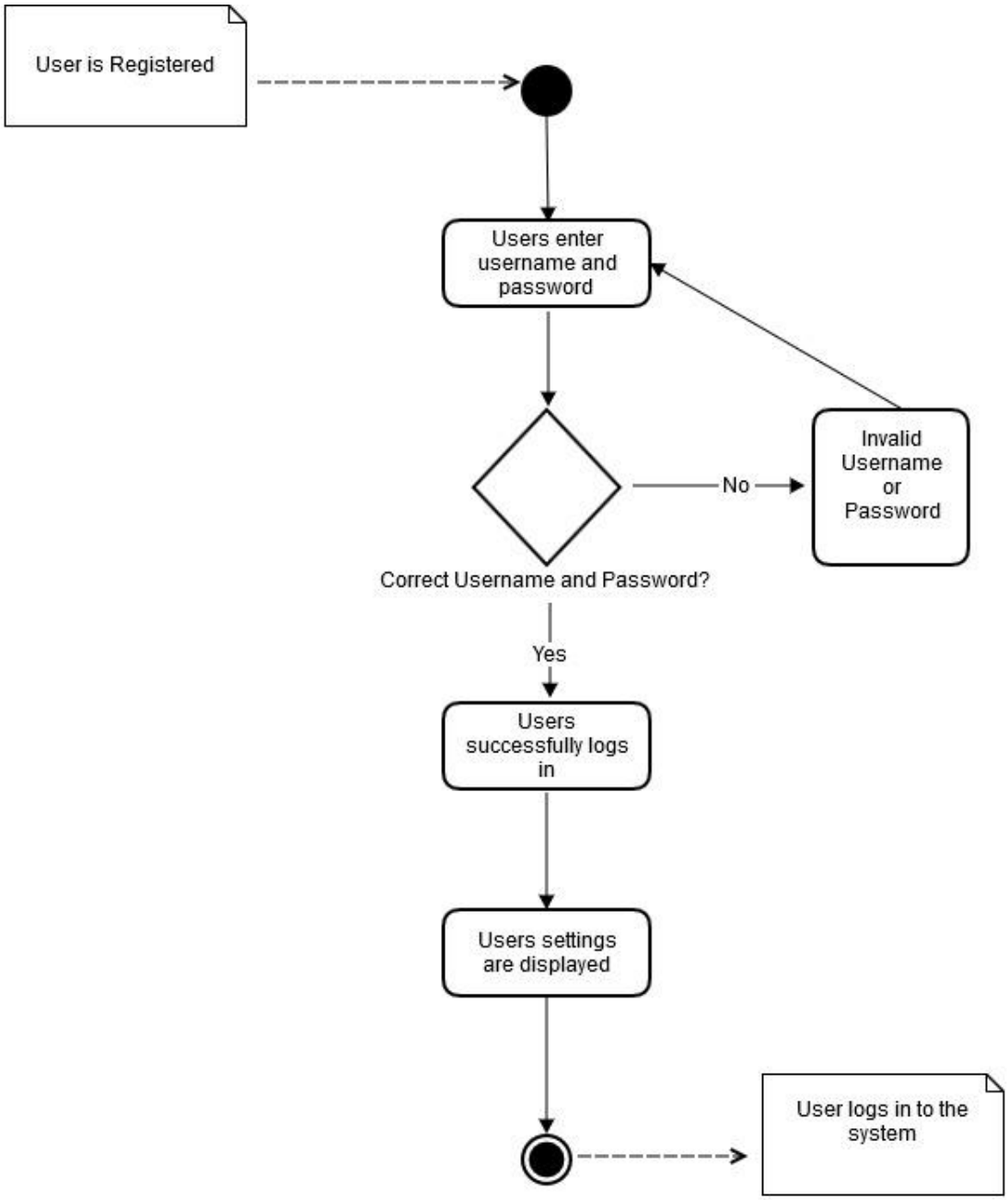
2.4 Use Case Diagram:

Figure 10 UseCase Diagram

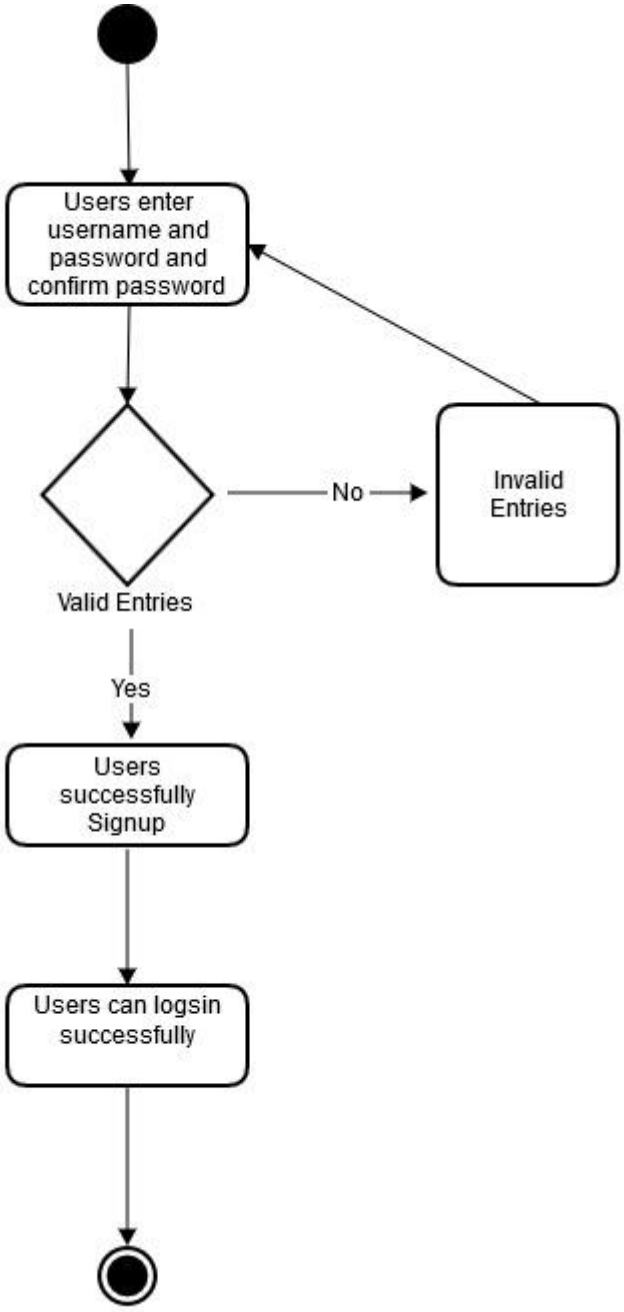


2.4.1 Activity Diagrams:

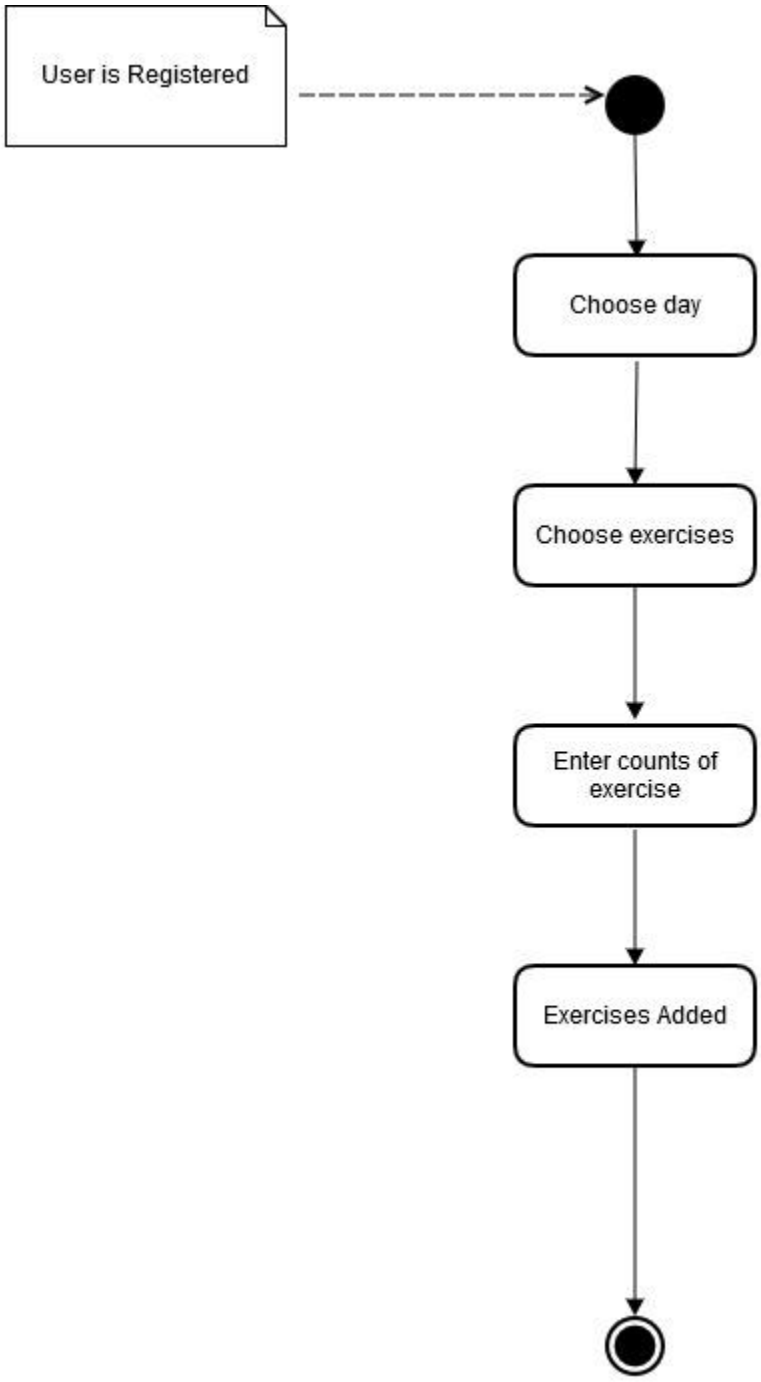
2.4.1.1 UC01:



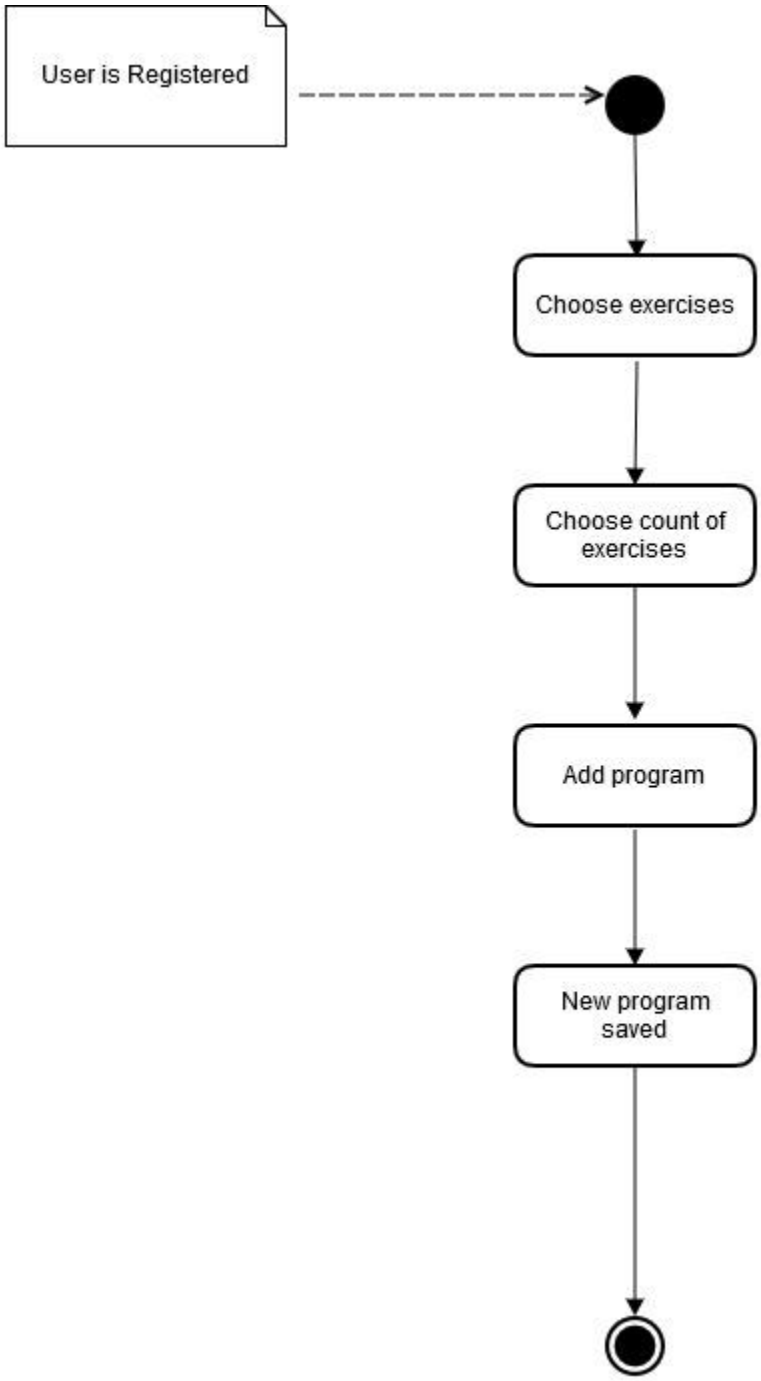
2.4.1.2 UC02:


Users enter username and password and confirm password
Valid Entries
No
Invalid Entries
Yes
Users successfully Signup
Users can logsin successfully

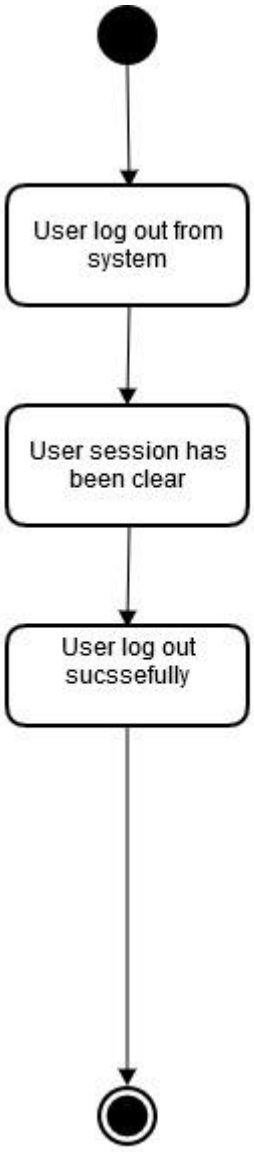
2.4.1.3 UC03:


User is Registered
Choose day
Choose exercises
Enter counts of exercise
Exercises Added

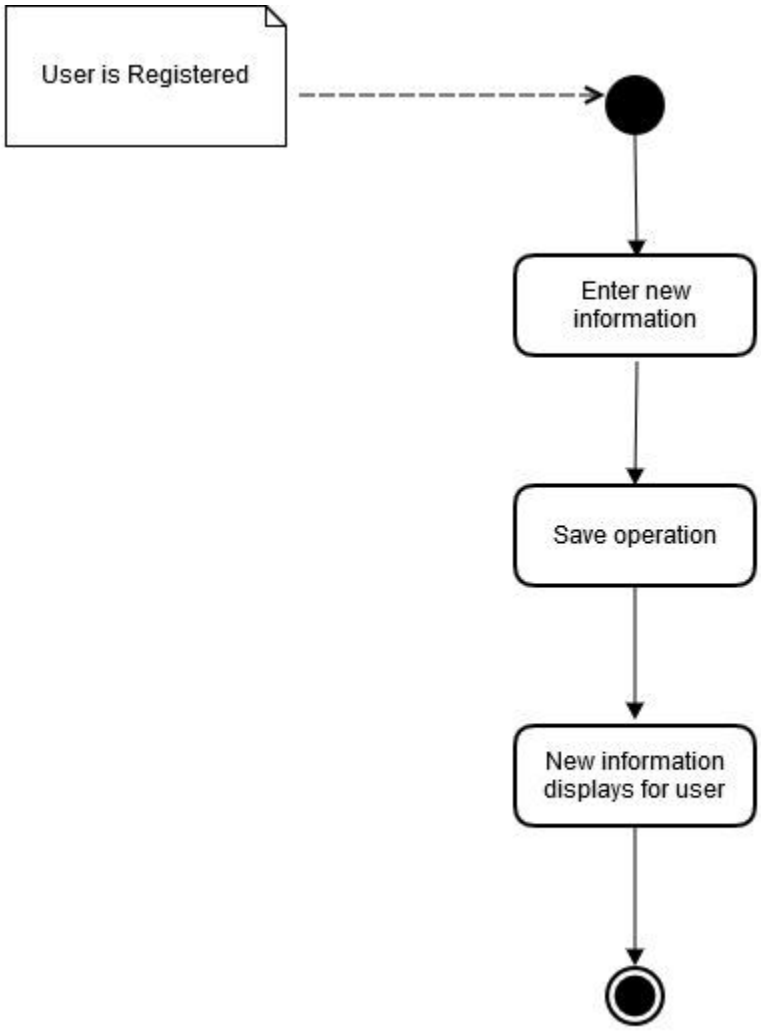
2.4.1.4 UC04:


User is Registered
Choose exercises
Choose count of exercises
Add program
New program saved

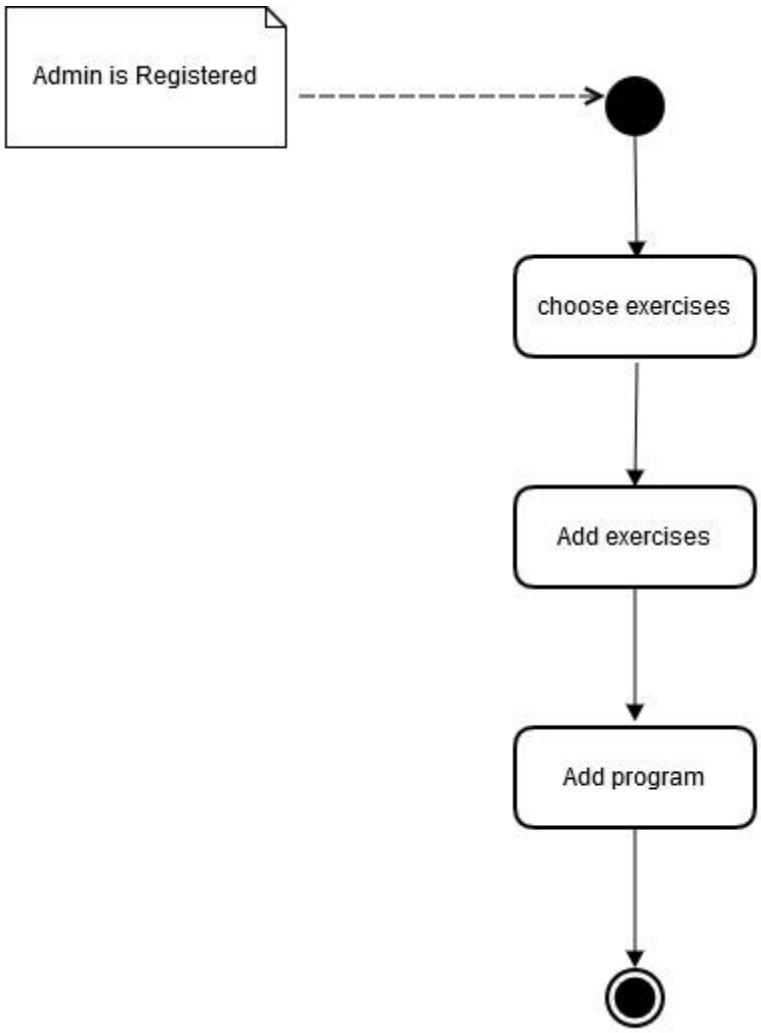
2.4.1.5 UC05:


User log out from system
User session has been clear
User log out sucssefully

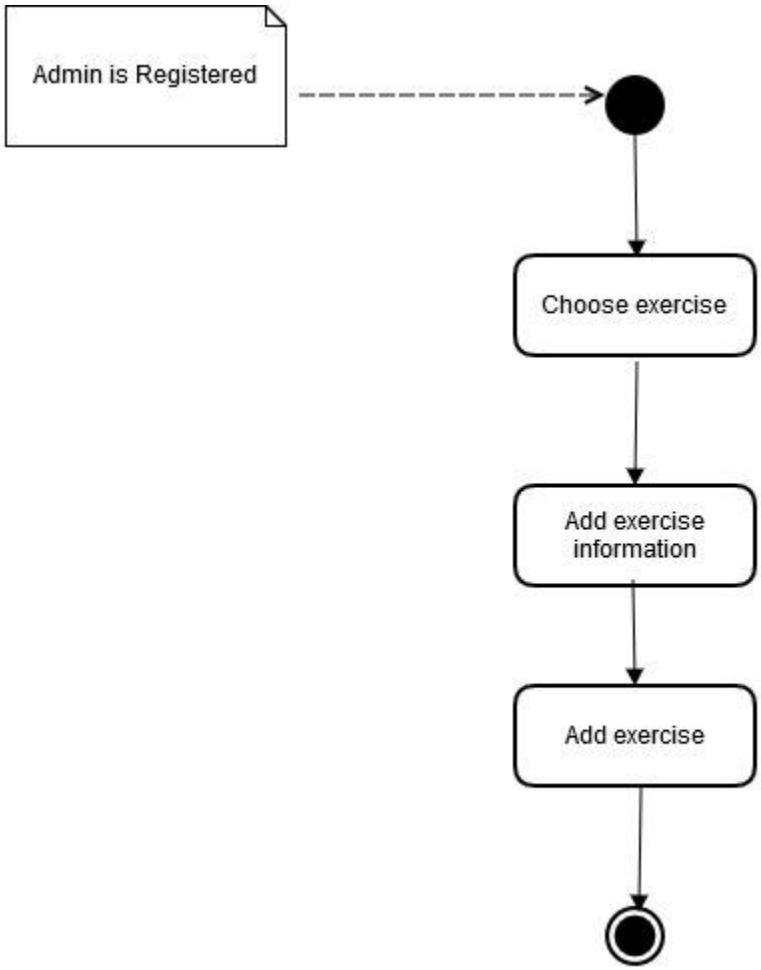
2.4.1.6 UC06:


User is Registered
Enter new information
Save operation
New information displays for user

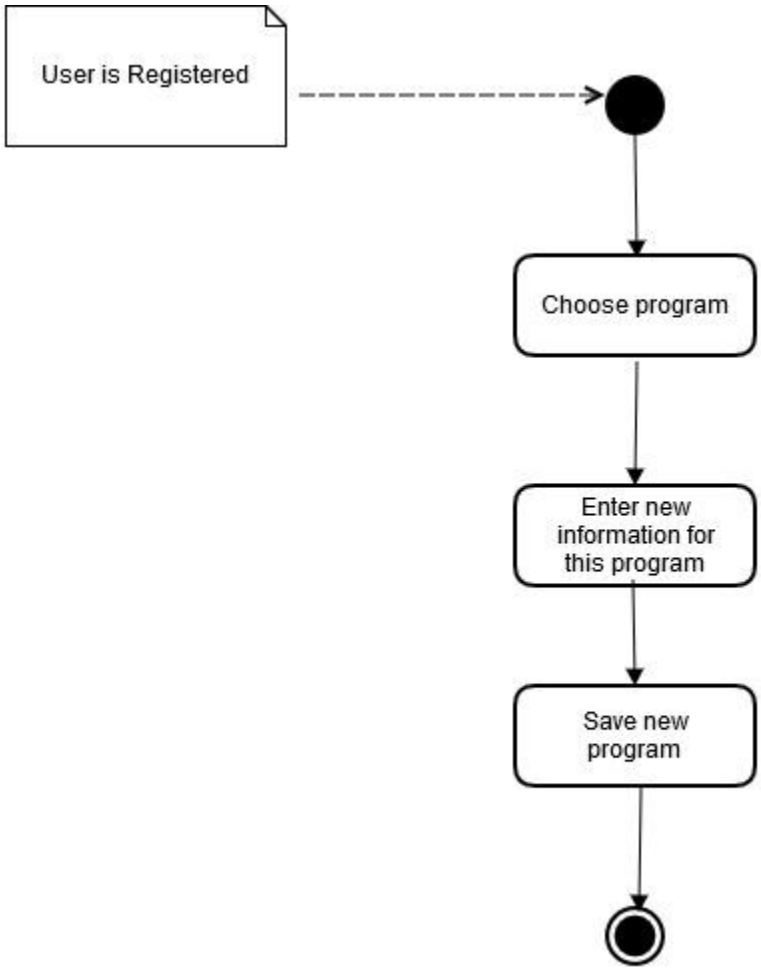
2.4.1.7 UC07:


Admin is Registered
choose exercises
Add exercises
Add program

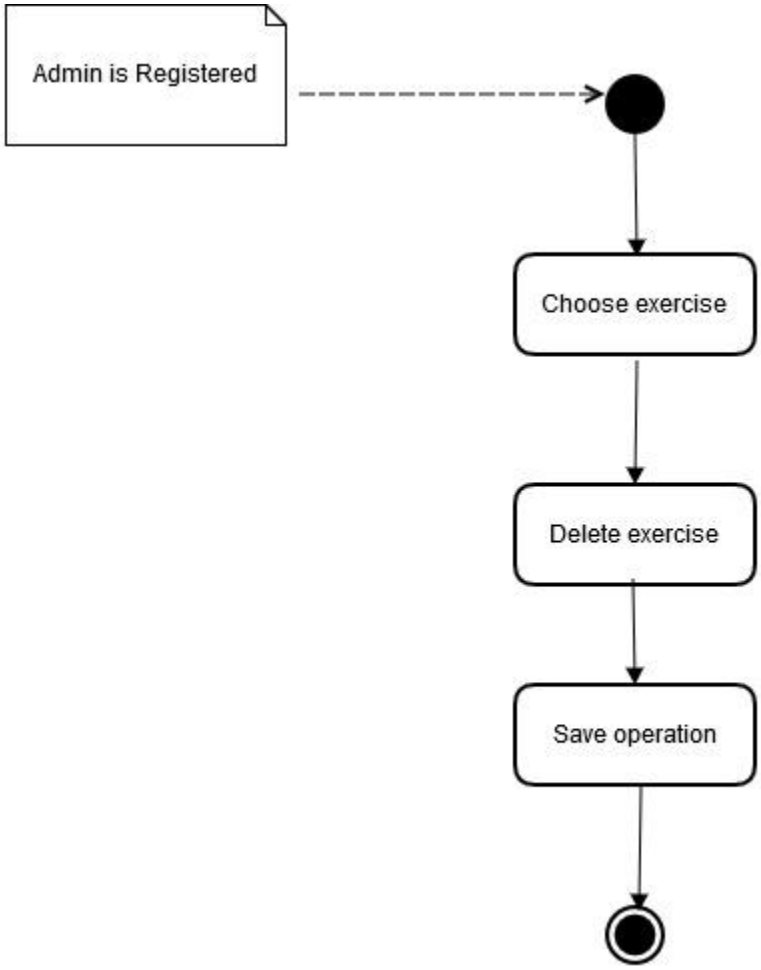
2.4.1.8 UC08:


Admin is Registered
Choose exercise
Add exercise information
Add exercise

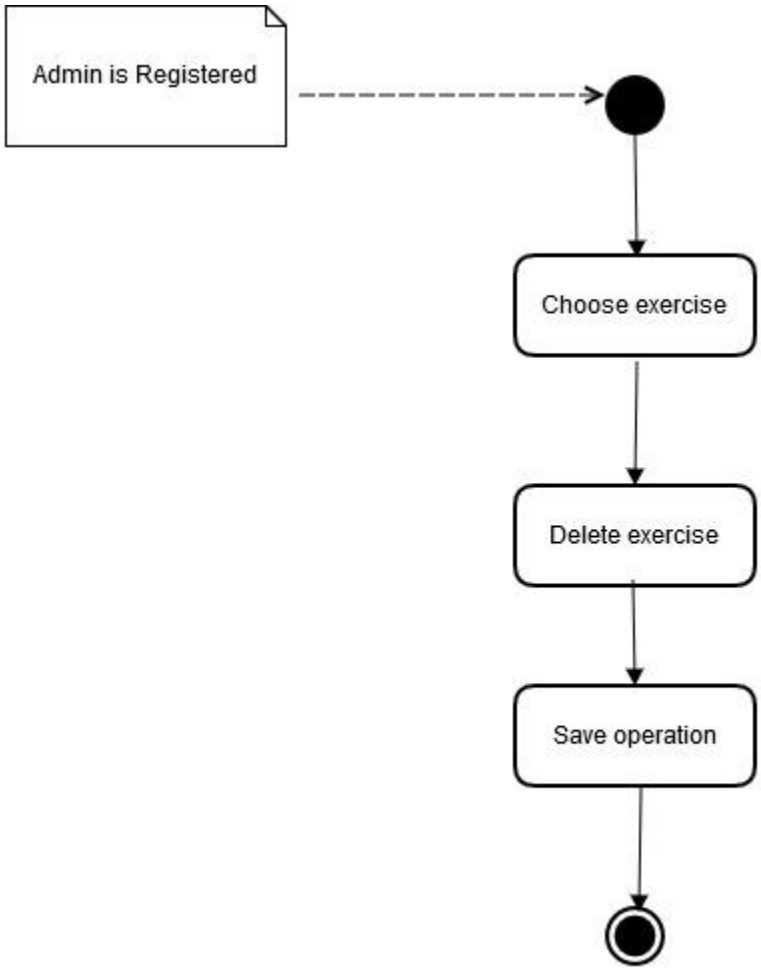
2.4.1.9 UC09:


User is Registered
Choose program
Enter new information for this program
Save new program

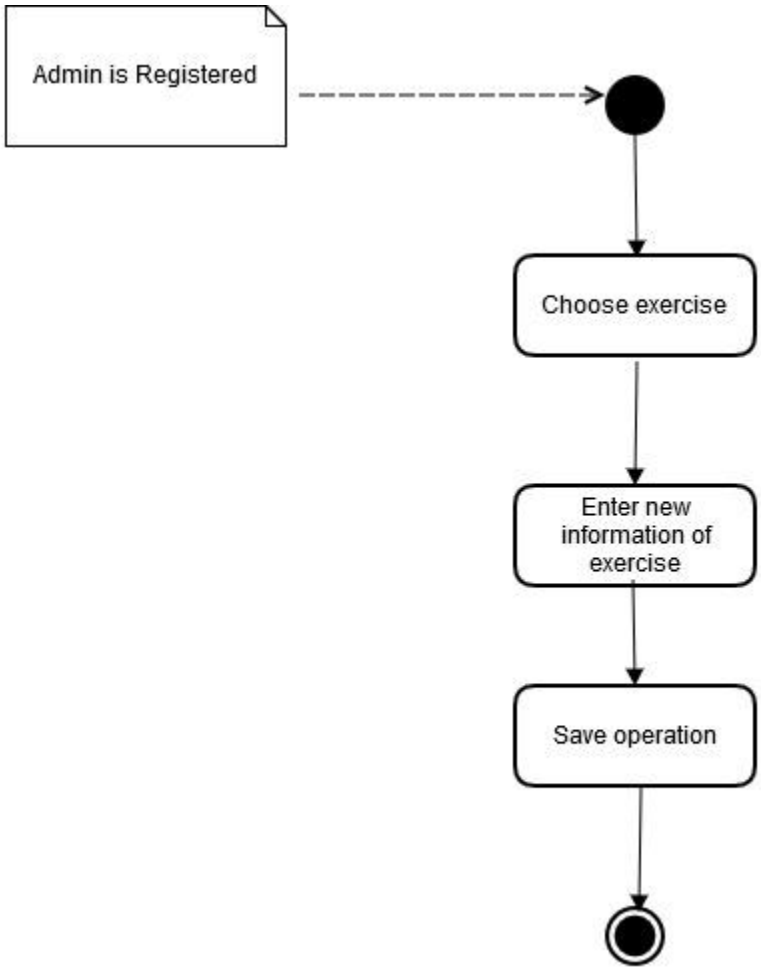
2.4.1.10 UC10:


Admin is Registered
Choose exercise
Delete exercise
Save operation

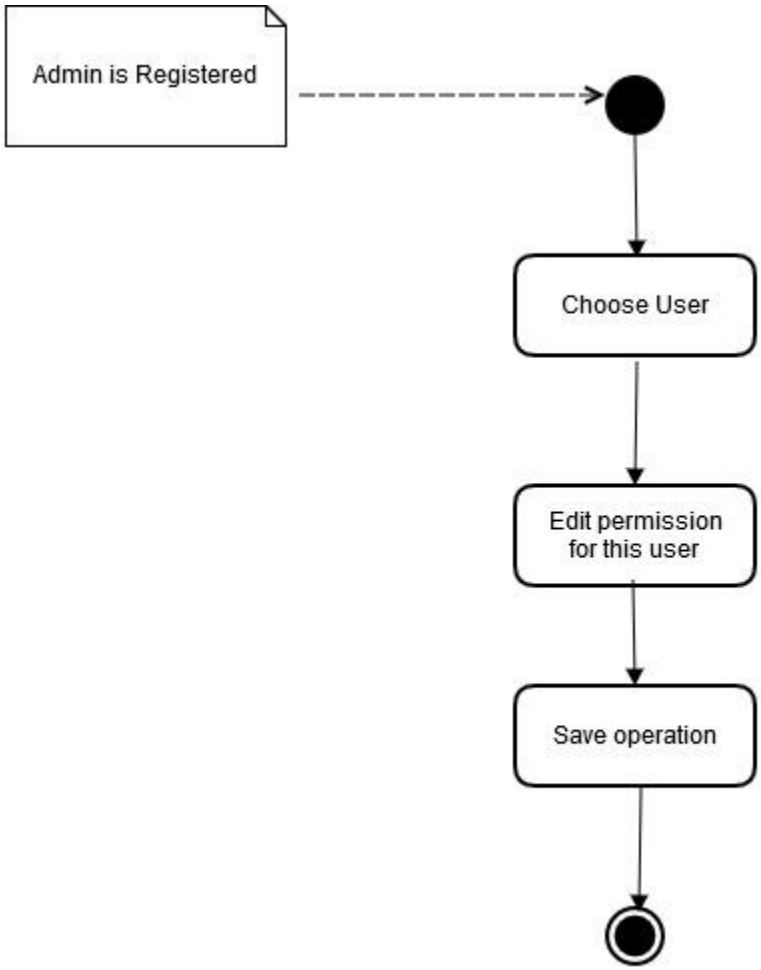
2.4.1.11 UC11:


Admin is Registered
Choose exercise
Delete exercise
Save operation

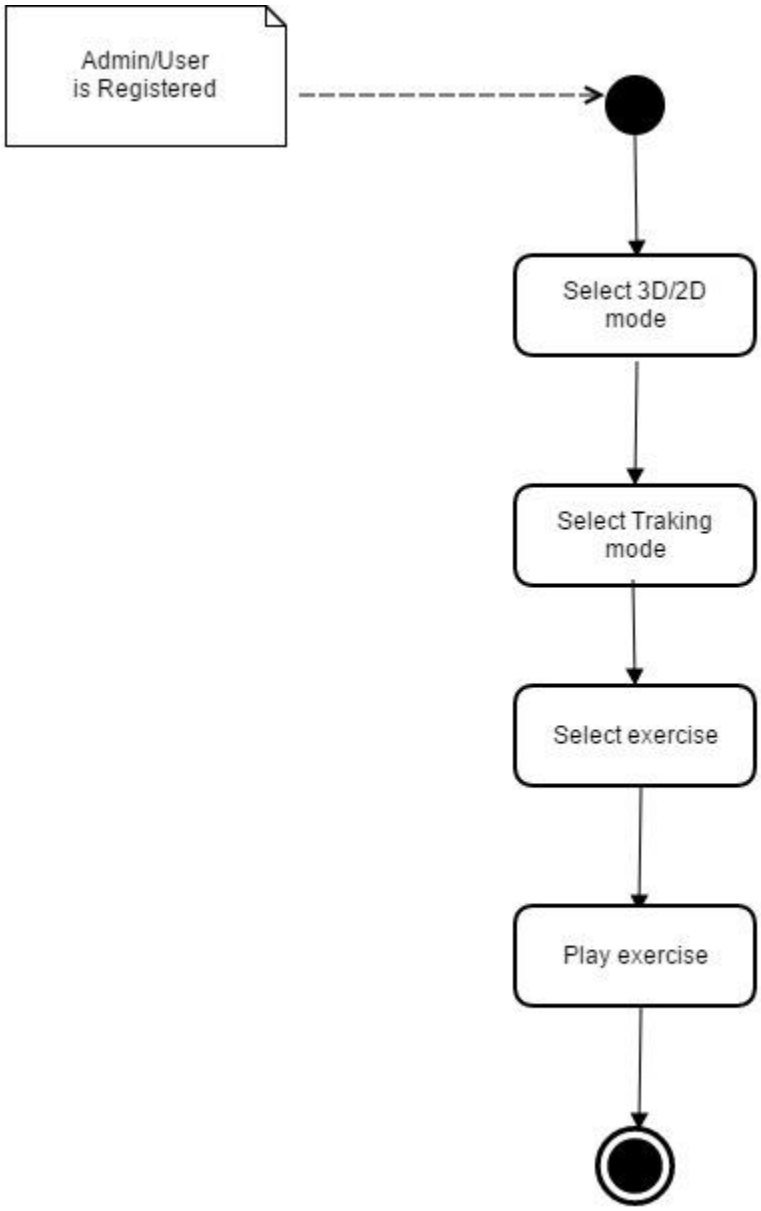
2.4.1.12 UC12:


Admin is Registered
Choose exercise
Enter new information of exercise
Save operation

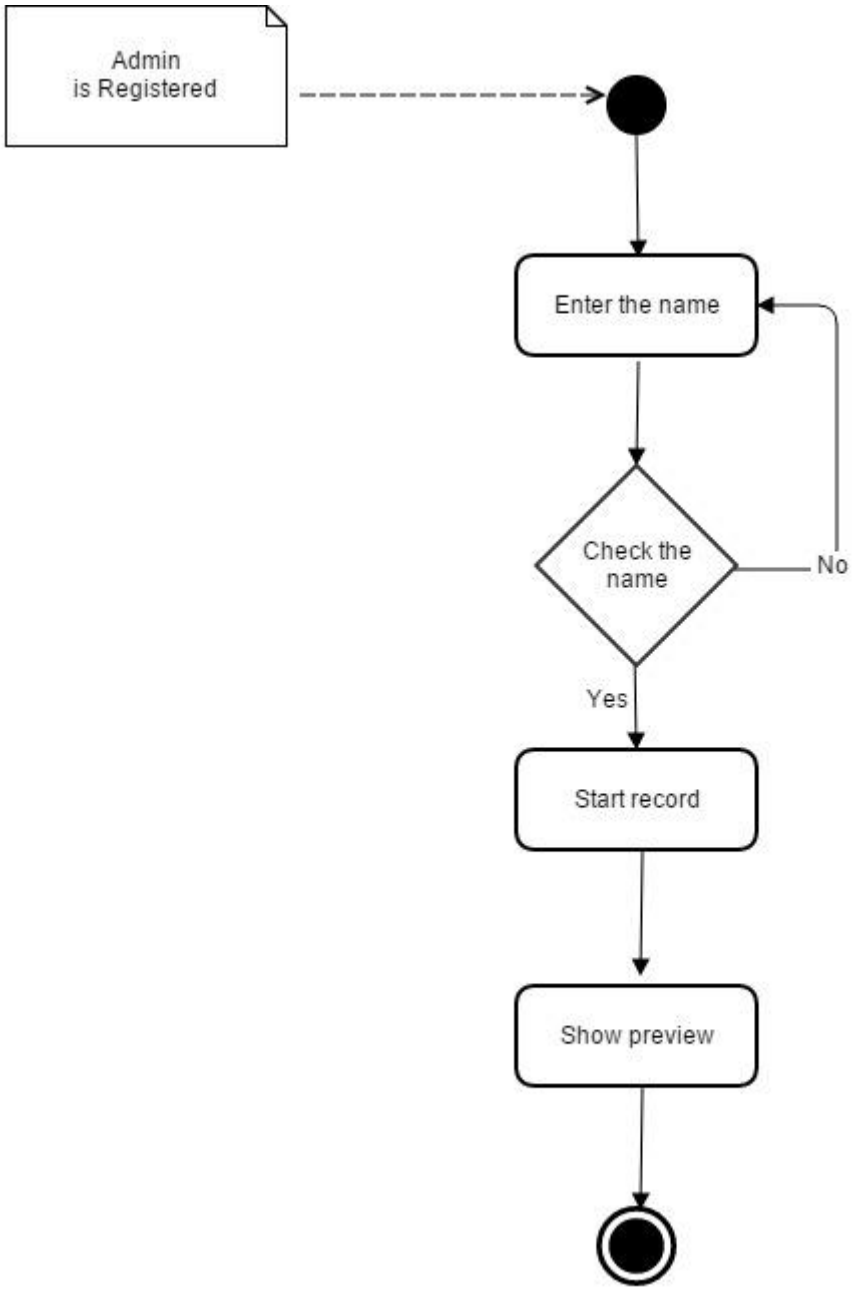
2.4.1.13 UC13:


Admin is Registered
Choose User
Edit permission for this user
Save operation

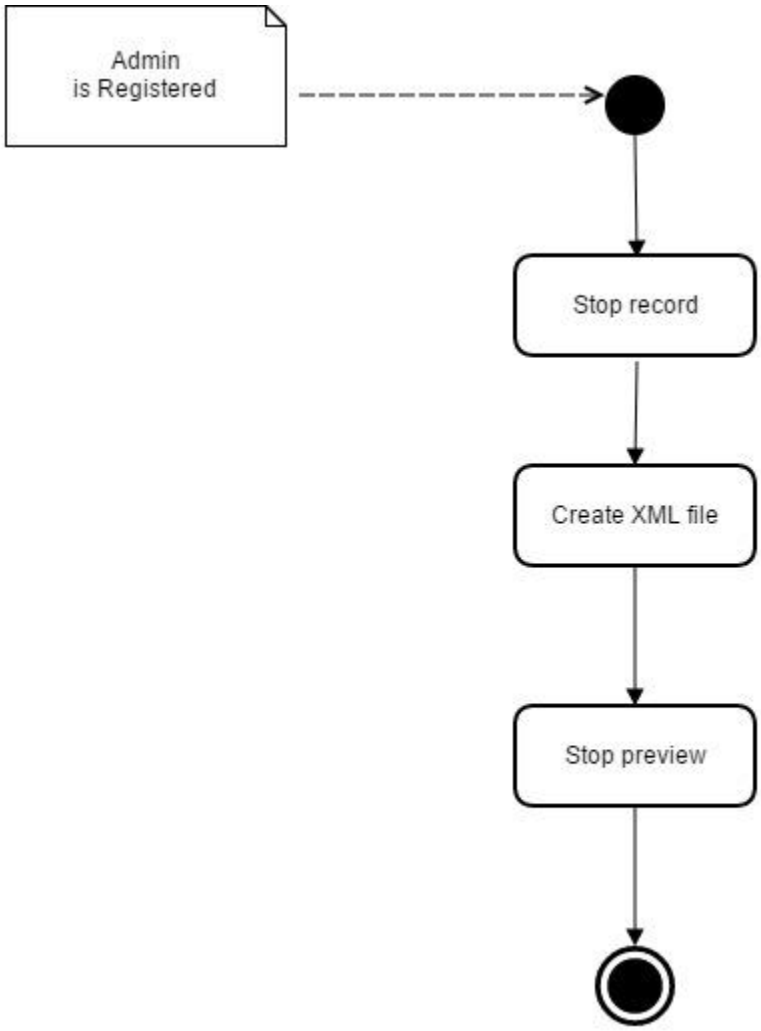
2.4.1.14 UC14:


Admin/User
is Registered
Select 3D/2D
mode
Select Traking
mode
Select exercise
Play exercise

2.4.1.15 UC15:

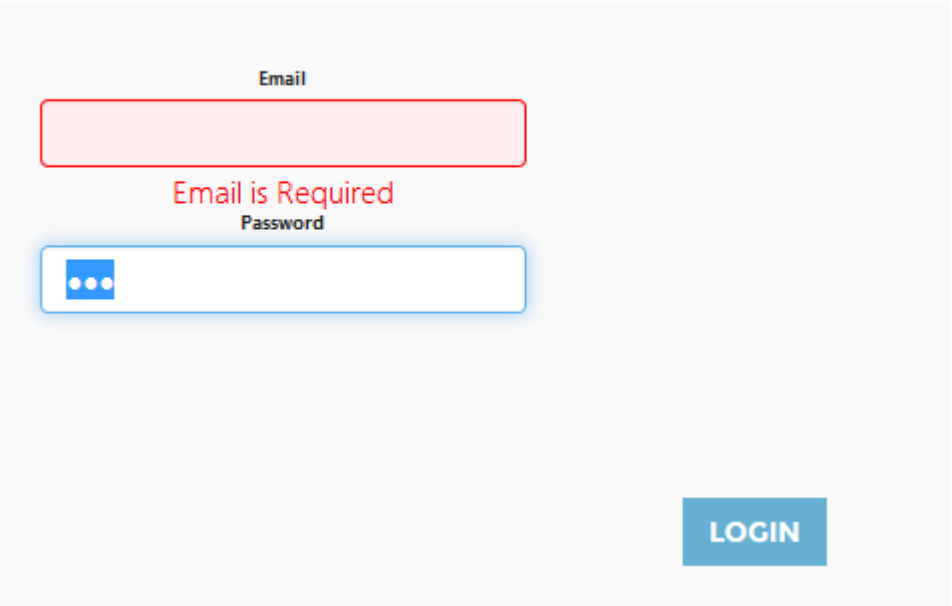

Admin
is Registered
Enter the name
Check the
name
No
Yes
Start record
Show preview

2.4.1.16 UC16:

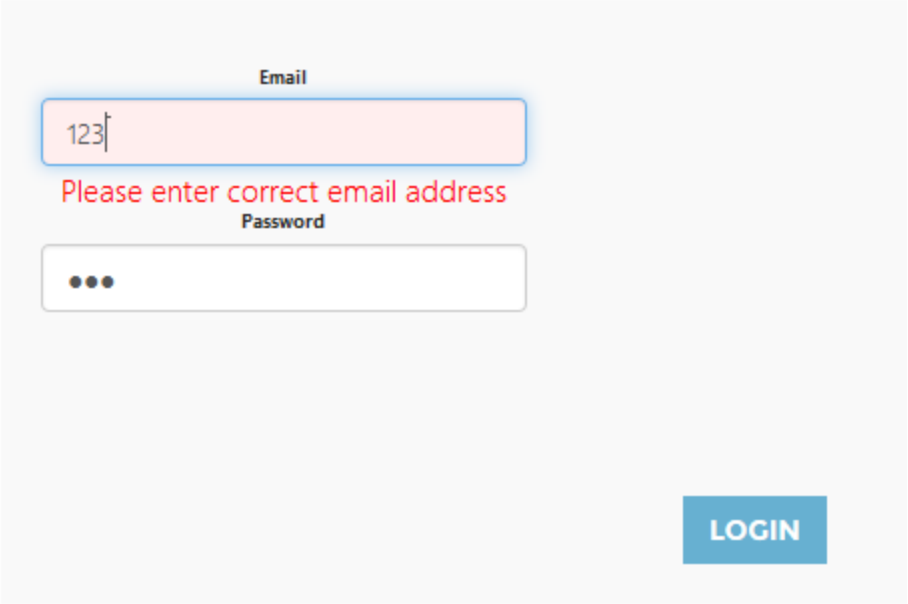

Admin
is Registered
Stop record
Create XML file
Stop preview

2.4.2 Testing:

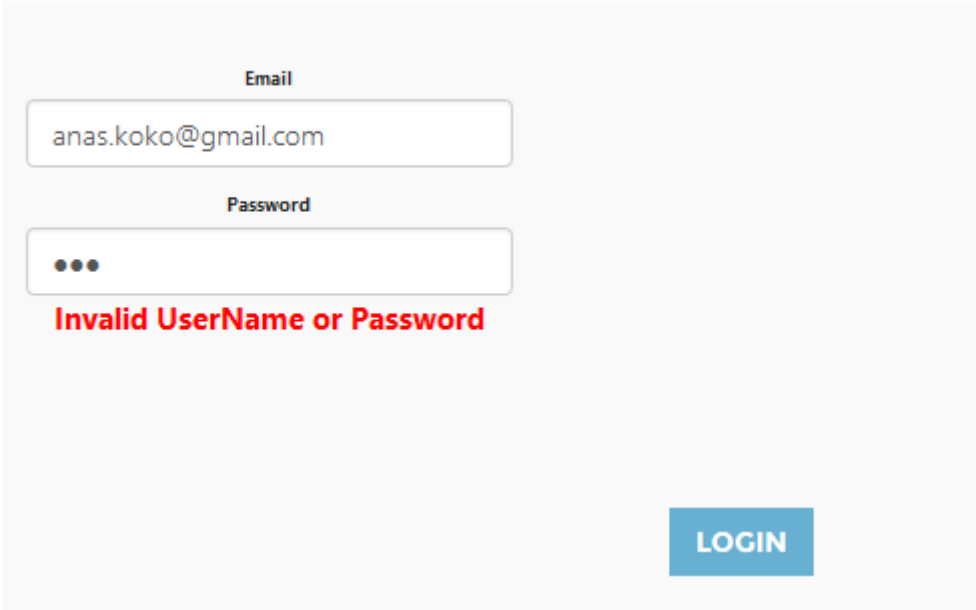
2.4.2.1 Check Email entry in log in form:



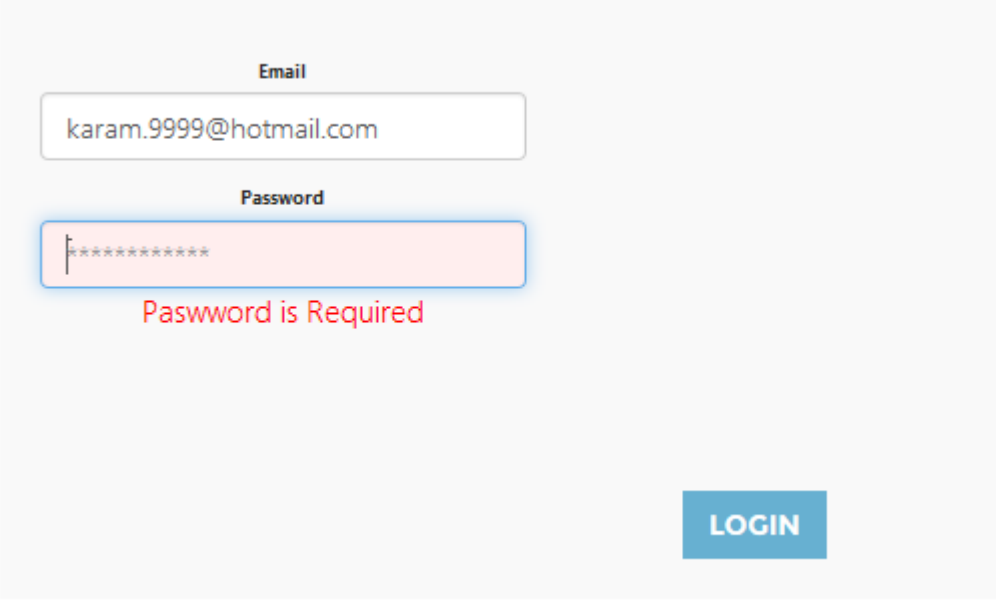
2.4.2.2 Check Parser of email in log in form:



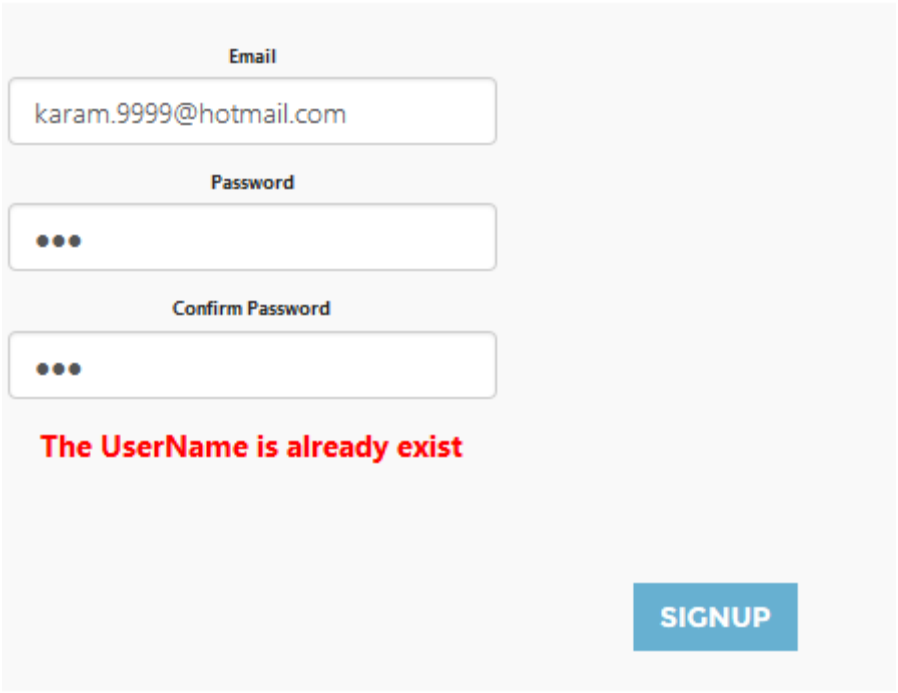
2.4.2.3 Check Email and Password in log in form:



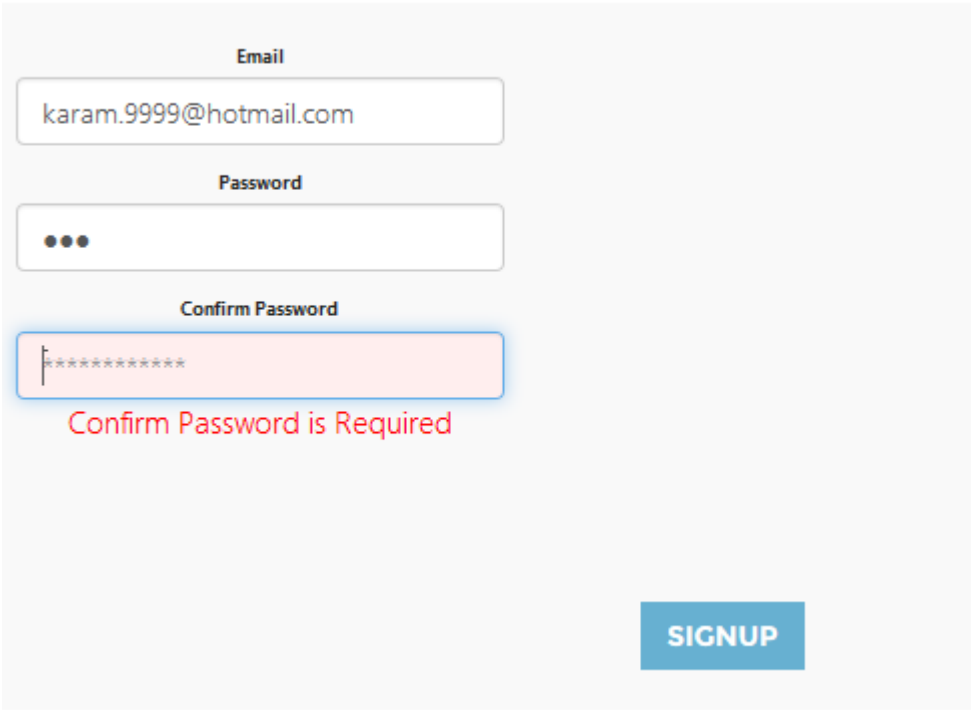
2.4.2.4 Check Password entry in log in form:



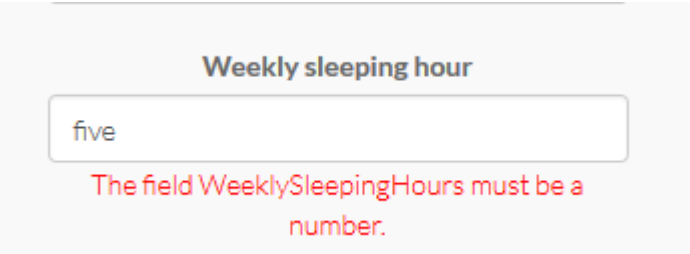
2.4.2.5 Check Username if exist in Sign up form:

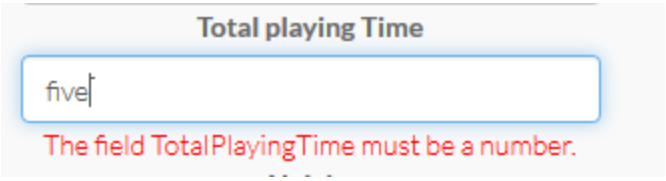


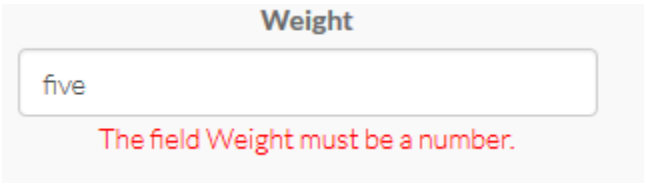
2.4.2.6 Check Confirm Password entry in Sign up form:



2.4.2.7 Check Fields in player information form :

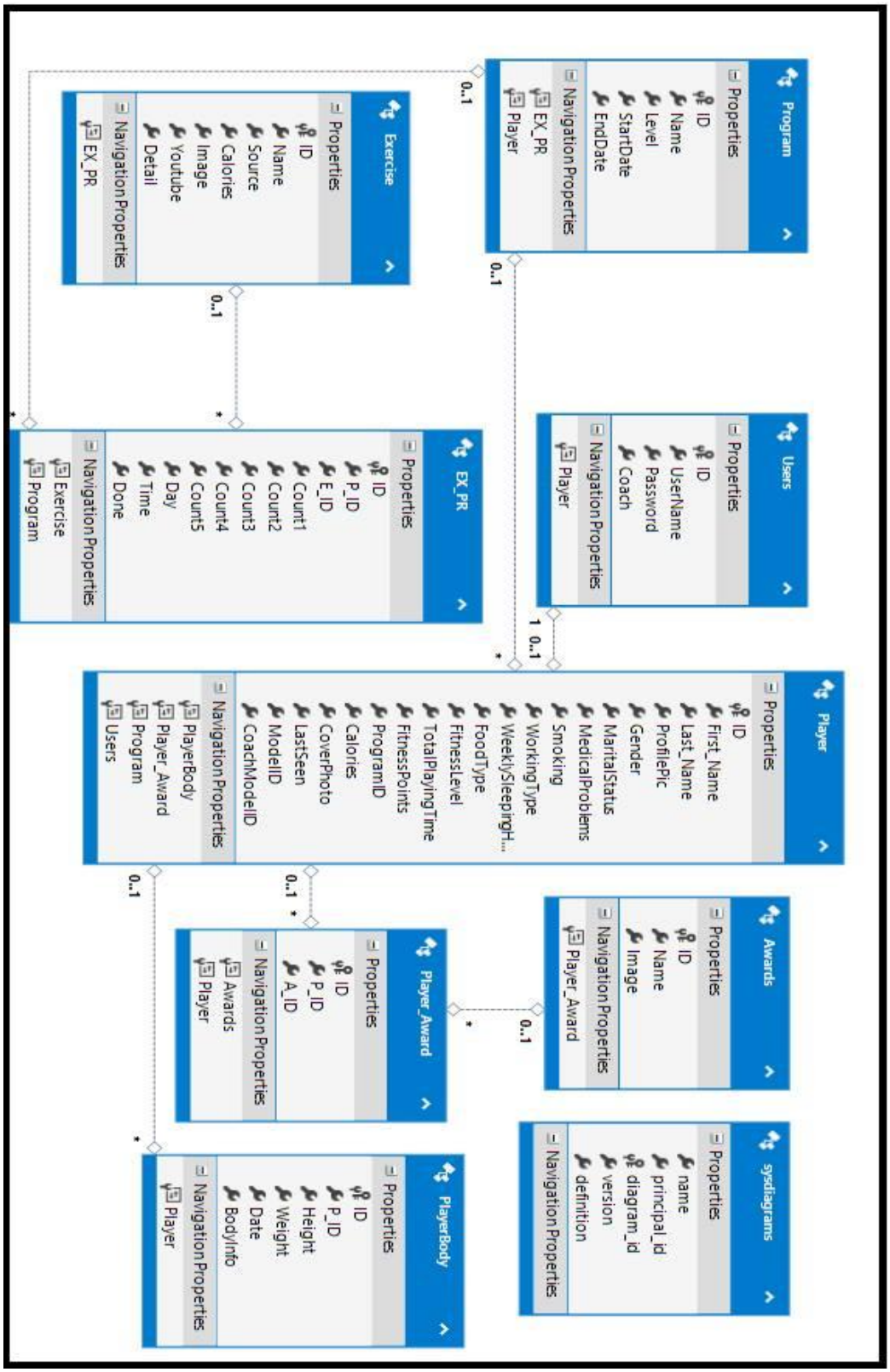






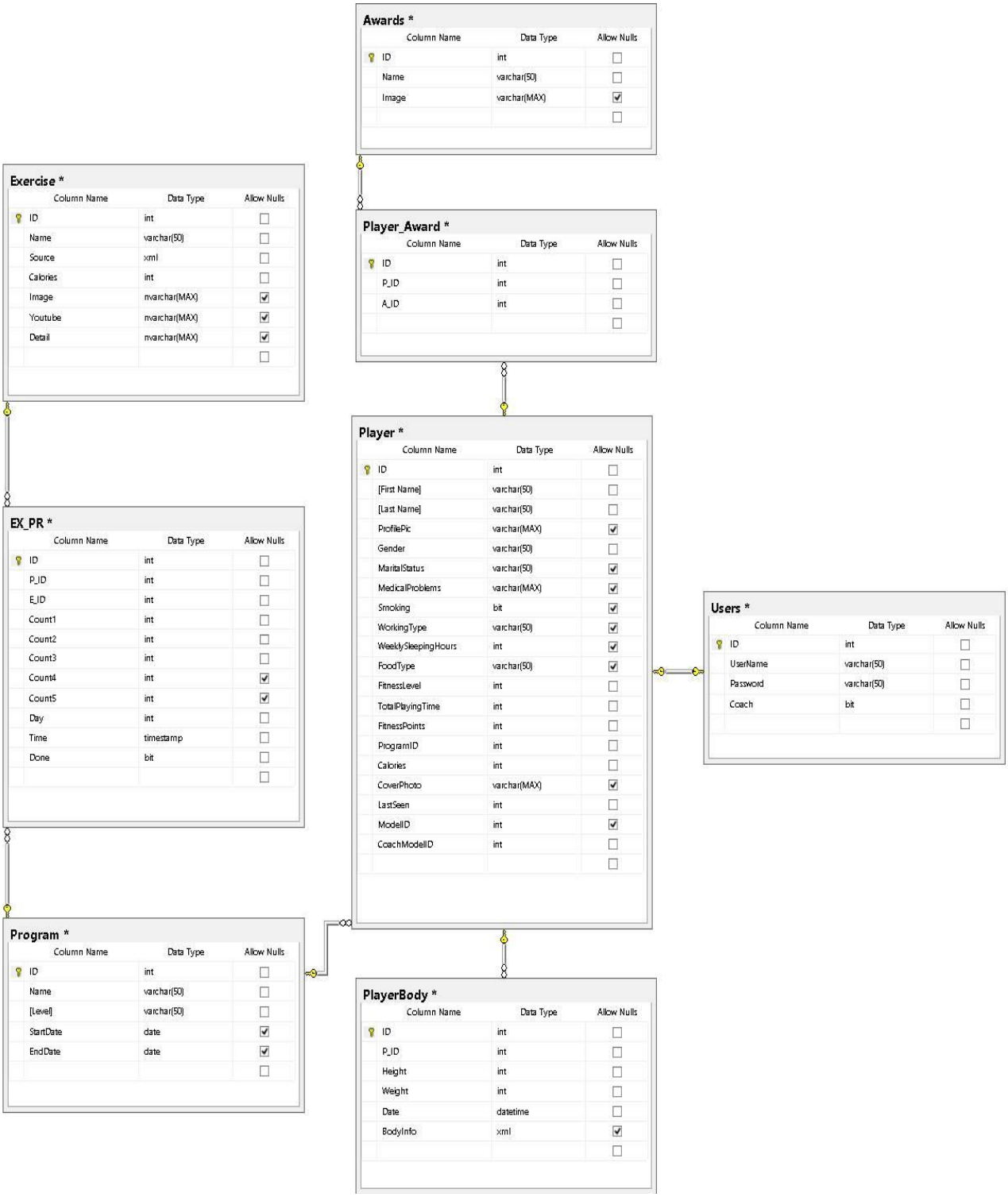
2.4.3 Class Diagram:

Figure 11 Class Diagram



2.4.4 ERD:

Figure 12 ERD



3 Website

3.1 Objectives:

The main objective of this website is to provide information about fitness and its services online in order to guide the user with playing exercises of fitness correctly and schedule his program of exercises daily.

Because we are living in a very modern and fast lifestyle, this website will also help the players saving their time by providing the ability to do their daily exercises anytime anywhere besides saving a lot of money he was paying gyms.

3.2 Project Category

This website as title “VirtualCoach” is comes under the Internet Category. It is an Internet based website, which helps the user of this system to perform any gym tasks at anywhere with the use of Internet.

3.3 Tools/Platform

This project is developed using the Internet Tools, which are most suited for development of the site. These tools are as follows:

1. HTML
2. HTML5
3. JavaScript
4. Razor web pages
5. ASP.net MVC
6. Cascading Style Sheet
7. MSSQL - Server 7.0 (For Database Storage)

3.4 ASP.NET:

ASP.NET is an open source server-side Web application framework designed for Web development to produce dynamic Web pages. It was developed by Microsoft to allow programmers to build dynamic web sites, web applications and web services.

It was first released in January 2002 with version 1.0 of the .NET Framework, and is the successor to Microsoft's Active Server Pages (ASP) technology. ASP.NET is built on the Common Language Runtime (CLR), allowing programmers to write ASP.NET code using any supported .NET language. The ASP.NET SOAP extension framework allows ASP.NET components to process SOAP messages.

ASP.NET is in the process of being re-implemented as a modern and modular web framework, together with other frameworks like Entity. The new framework will make use of the new open-source .NET Compiler Platform (code-name "Roslyn") and be cross. ASP.NET MVC, ASP.NET Web API, and ASP.NET Web Pages (a platform using only Razor pages) will merge into a unified MVC the project is called "ASP.NET".

ASP.NET is a unified Web development model that includes the services necessary for you to build enterprise-class Web applications with a minimum of coding. ASP.NET is part of the .NET Framework, and when coding ASP.NET applications you have access to classes in the .NET Framework. You can code your applications in any language compatible with the common language runtime (CLR), including Microsoft Visual Basic and C#

Figure 13 ASP.net Logo

3.5 The Three Flavors of ASP.NET: Web Forms, MVC, and Web Pages

ASP.NET offers three frameworks for creating web applications: ASP.NET Web Forms, ASP.NET MVC, and ASP.NET Web Pages. All three frameworks are stable and mature, and you can create great web applications with any of them.

Each framework targets a different audience or type of application. Which one you choose depends on a combination of your web development experience, what framework you are most comfortable with, and which is the best fit for the type of application you are creating. All three frameworks will be supported, updated, and improved in future releases of ASP.NET.

3.6 ASP MVC:

3.6.1 Definition:

Model-View-Controller (MVC) is probably one of the most quoted patterns in the web-programming world in recent years. Anyone currently working in anything related to web application development will have heard or read the acronym hundreds of times. Today, we will clarify what MVC means, and why it has become so popular.

MVC is a framework for building web applications using a MVC (Model View Controller) design:

The Model represents the application core (for instance a list of database records).

The View displays the data (the database records).

The Controller handles the input (to the database records).

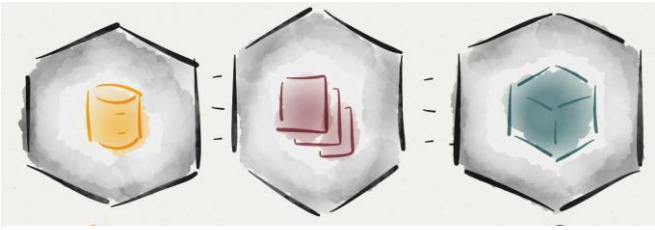


Figure 14 MVC logo

3.6.2 MVC Component:

3.6.2.1 Controller

The Controller manages the user requests (received as HTTP GET or POST requests when the user clicks on GUI elements to perform actions). Its main function is to call and coordinate the necessary resources/objects needed to perform the user action. Usually the controller will call the appropriate model for the task and then selects the proper view.

3.6.2.2 Model

The Model is the data and the rules applying to that data, which represent concepts that the application manages. In any software system, everything is modeled as data that we handle in a certain way. What is a user, a message or a book for an application? Only data that must be handled according to specific rules (date cannot be in the future, e-mail must have a specific format, name cannot be more than x characters long, etc.).

The model gives the controller a data representation of whatever the user requested (a message, a list of books, a photo album, etc.). This data model will be the same no matter how we may want to present it to the user that is why we can choose any available view to render it.

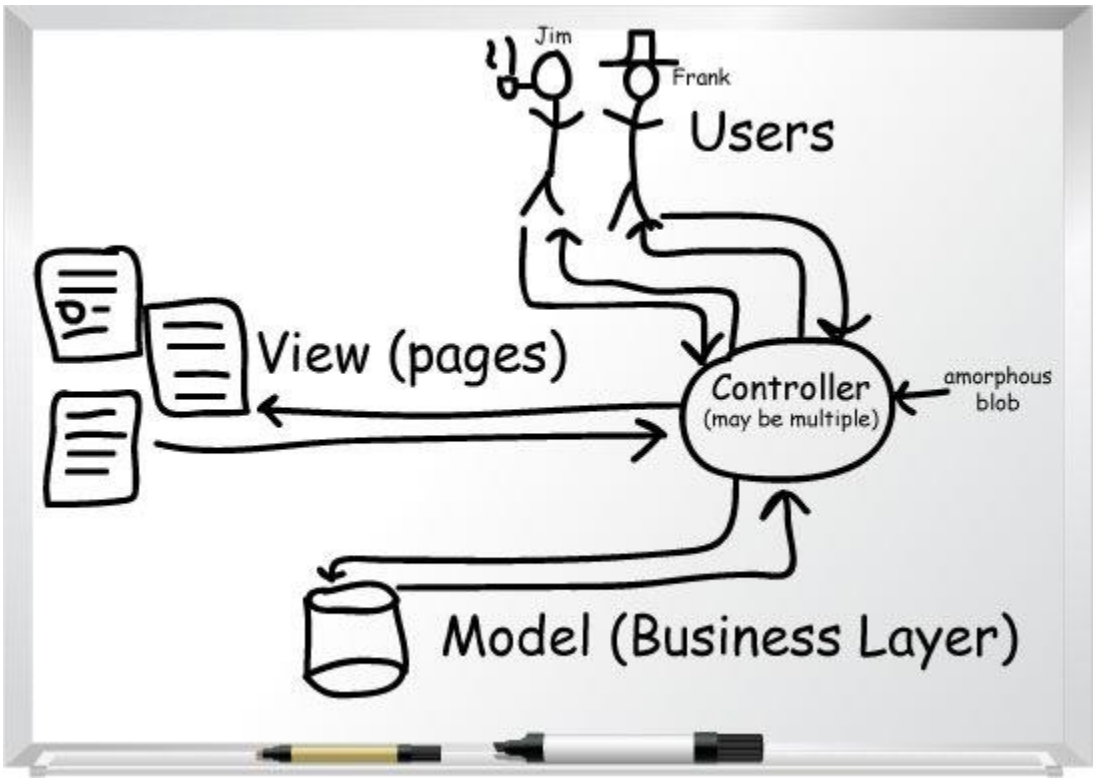
The model contains the most important part of our application logic, the logic that applies to the problem we are dealing with (a forum, a shop, a bank, etc.). The controller contains a more internal-organizational logic for the application itself (more like housekeeping).

3.6.2.3 View

The View provides different ways to present the data received from the model. They may be templates where that data is filled. There may be several different views and the controller has to decide which one to use.

A web application is usually composed of a set of controllers, models and views. The controller may be structured as a main controller that receives all requests and calls specific controllers that handle actions for each case.

Figure 15 MVC View-Model Rlation



3.7 LINQ:

Figure 16 Linq Logo

Language-Integrated Query (LINQ) is an innovation introduced in Visual Studio 2008 and .NET Framework version 3.5 that bridges the gap between the world of objects and the world of data.

Traditionally, queries against data are expressed as simple strings without type checking at compile time or IntelliSense support. Furthermore, you have to learn a different query language for each type of data source: SQL databases, XML documents, various Web services, and so on…

LINQ makes a query a first-class language construct in C# and Visual Basic. You write queries against strongly typed collections of objects by using language keywords and familiar operators. The following illustration shows a partially completed LINQ query against a SQL Server database in C# with full type checking and IntelliSense support.

3.8 IIS Server:

Figure 17 IIS logo

IIS (Internet Information Server) is a group of Internet servers (including a Web or Hypertext Transfer Protocol server and a File Transfer Protocol server) with additional capabilities for Microsoft's Windows NT and Windows Server operating systems. IIS is Microsoft's entry to compete in the Internet server market that is also addressed by Apache, Sun Microsystems, O'Reilly, and others

With IIS, Microsoft includes a set of programs for building and administering Web sites, a search engine, and support for writing Web-based applications that access databases. Microsoft points out that IIS is tightly integrated with the Windows NT and 2000 Servers in a number of ways, resulting in faster Web page serving.

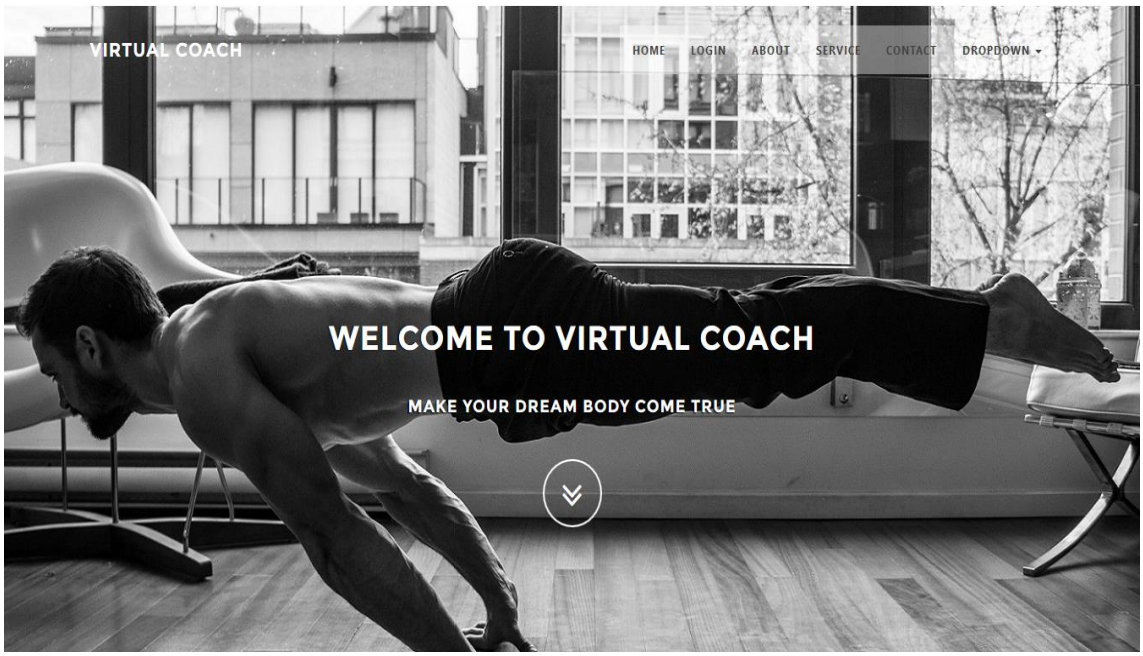
3.9 Design:

3.9.1 Welcome screen Form:

This form show home screen for the user and in the top there is a bar item contains many buttons and these buttons are:

- Home
- Login
- About
- Contact us

Figure 18 Home Page



Home: this button return the user to the home screen.

Login: this button send the user to login form.

About us: this button send user to the section that have information about designers

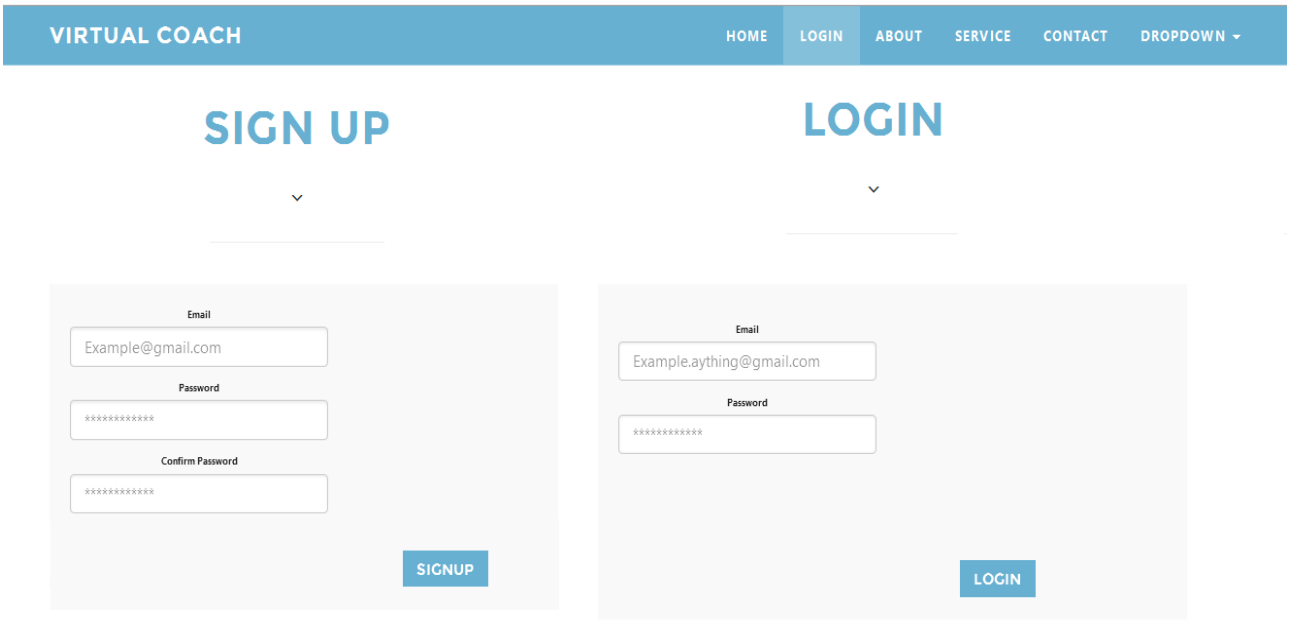
Contact us: this button send user to the section that have form to send email if any user have a problem with the system or if he wants to suggest any thing

3.9.2 Log in & Sign Up Form:

Log in form: this form allow to user to make login operation to the system by enter his email and his password and return to him message error if his email or password are wrong and it consist of two fields:

- Email
- Password

Figure 19 Signup/Login Section



Sign up form: this form allow to user to create new account in the system by enter his email and his password and confirm password and return to him message error if his email parser is wrong or his email is already exist or password's confirm are wrong and it consist of three fields:

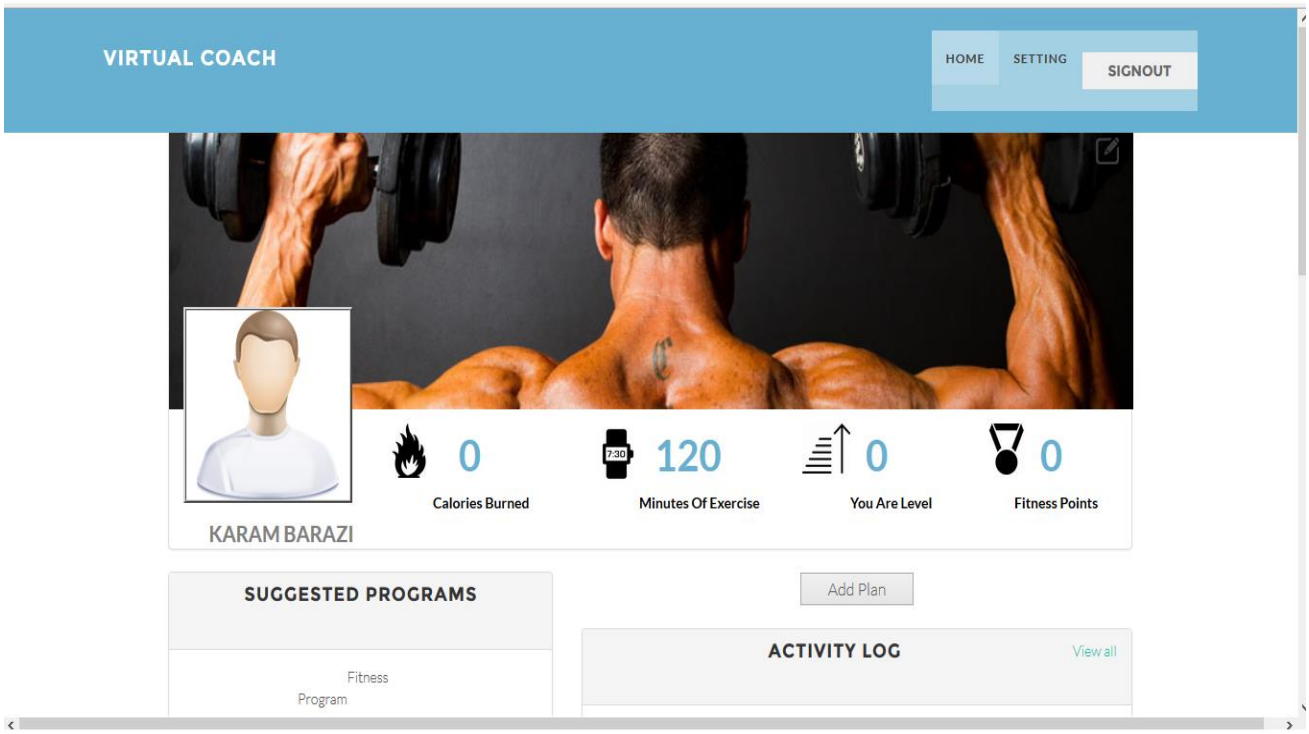
- Email
- Password
- Confirm Password

3.9.3 Profile Form:

This form show to the user important sport information and show profile picture and cover photo and show what the exercises that user have to play today and has a bar item that consist of many button:

- Home
- Settings
- Sign out

Figure 20 User Profile



Home: this button return the user to his home page (profile page).

Settings: this button show to the user settings page that allow to user to change his information.

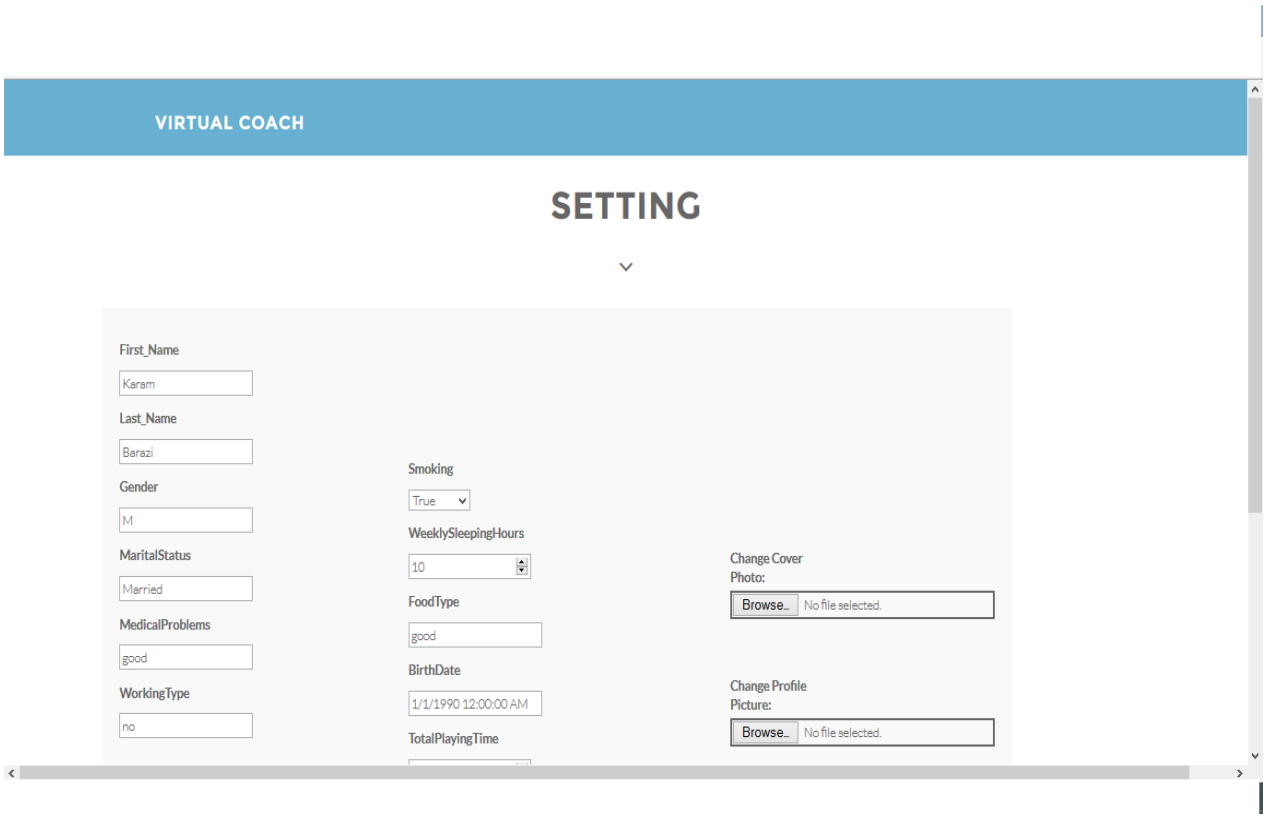
Sign out: allow user to make sign out operation from the system.

3.9.4 Settings Form:

This form allow to user to edit his own information and has a many of functions for his edit:

Like change his name, gender, smoking status, profile picture, cover photo.

Figure 21 Profile setting



After user finishing his edit then his new information will be appear in his profile page and the system return to this page directly when user is finish.

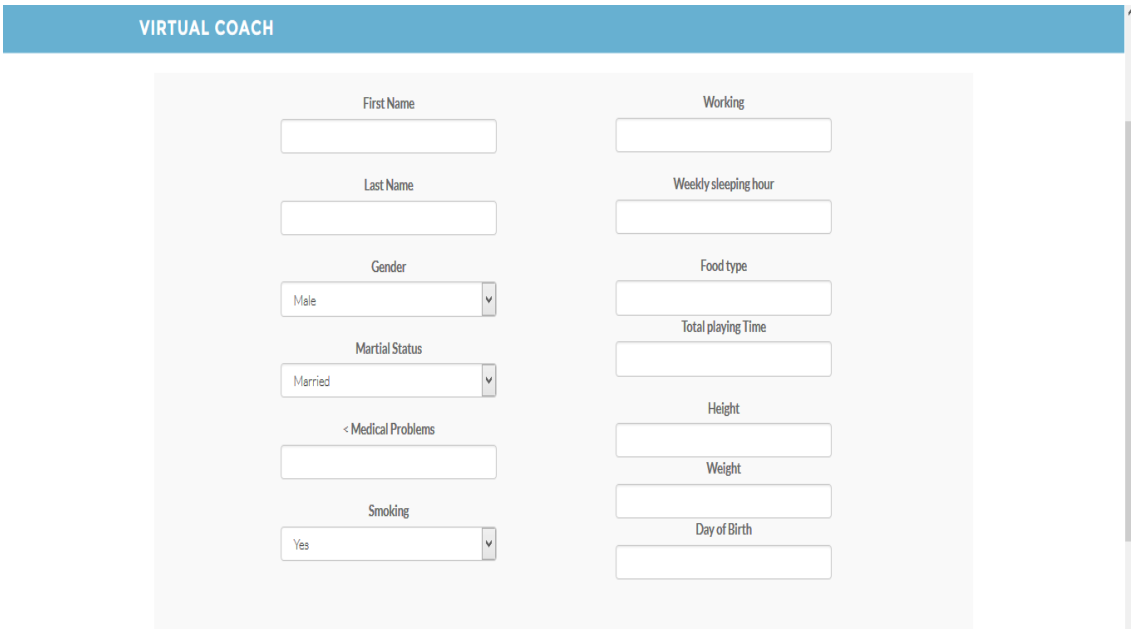
In change profile or cover photo system allow to user to upload his own picture by open a dialog box to him and then the user choose the photo he wants.

3.9.5 Player Information Form:

This form will appear to the user after sign up operation and allow to the user enter his own information and has many of fields that it's very important to choose a good program for user :

- First Name
- Last Name
- Gender
- Marital status
- Medical problems
- Weekly sleeping hour
- Food type
- Total playing time
- Height
- Width
- Date of birth

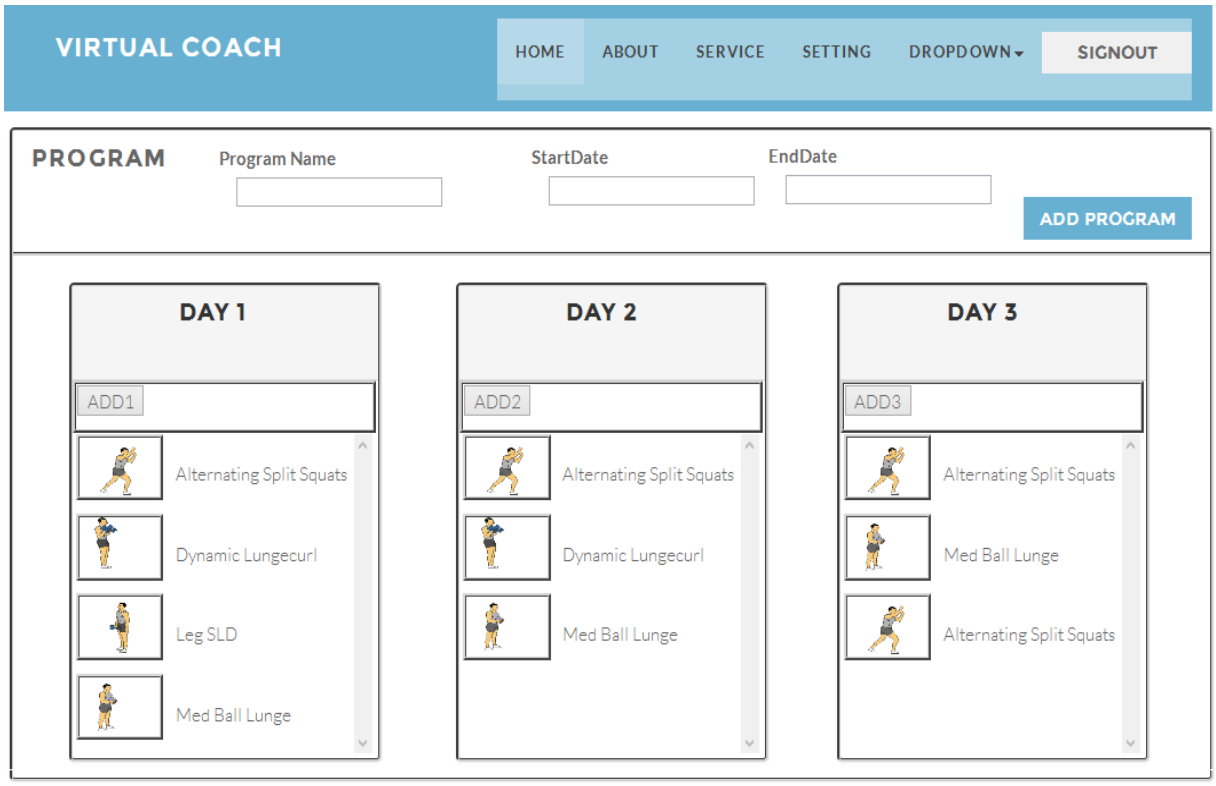
Figure 22 SignUP Section



3.9.6 Add program Form:

This form allow to user to add his own program in this form user will add his exercise in day1,day2,day3 and after finishing his exercise he will name this program and enter the start date and end date then after this operation the exercises will appear to the user in his home page

Figure 23 Add Fitness Prpgram

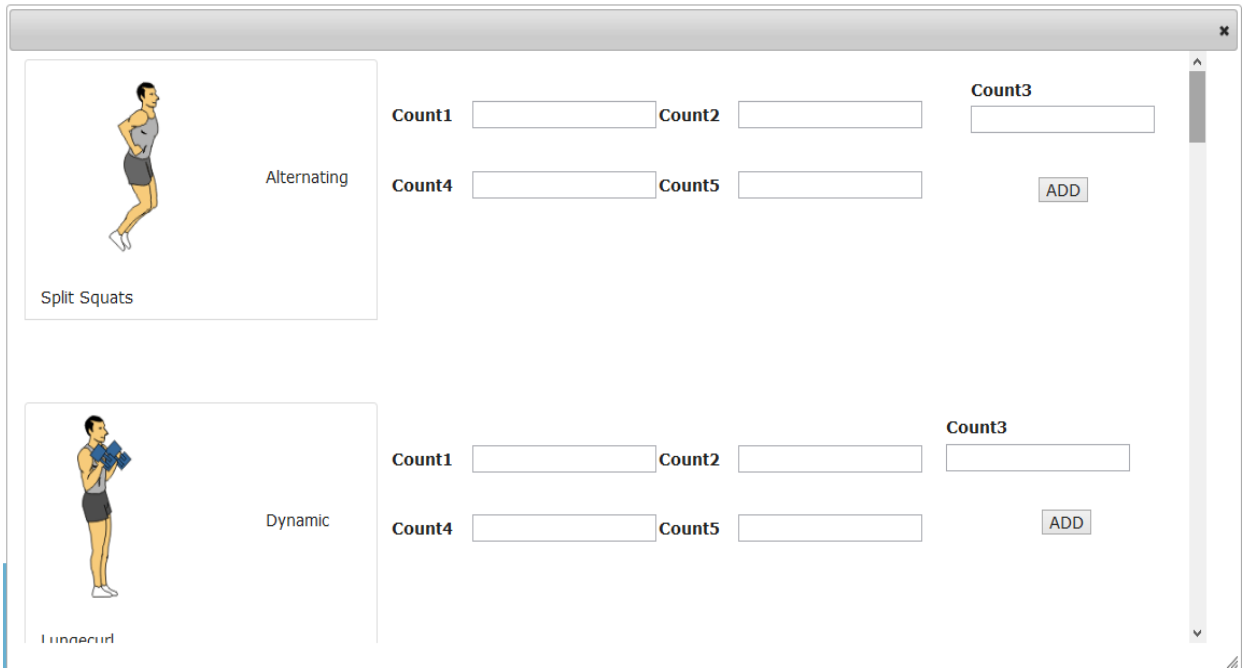


In his home page, his exercises will be appear to him and start in his first day and after finishing it the exercises of the second day will be appear to him and so.

3.9.7 Add exercise Form:

This form allow to user to choose his own exercises and add it to his new program and with each exercise there is an image to show how to do this exercise

Figure 24 Add Exercise



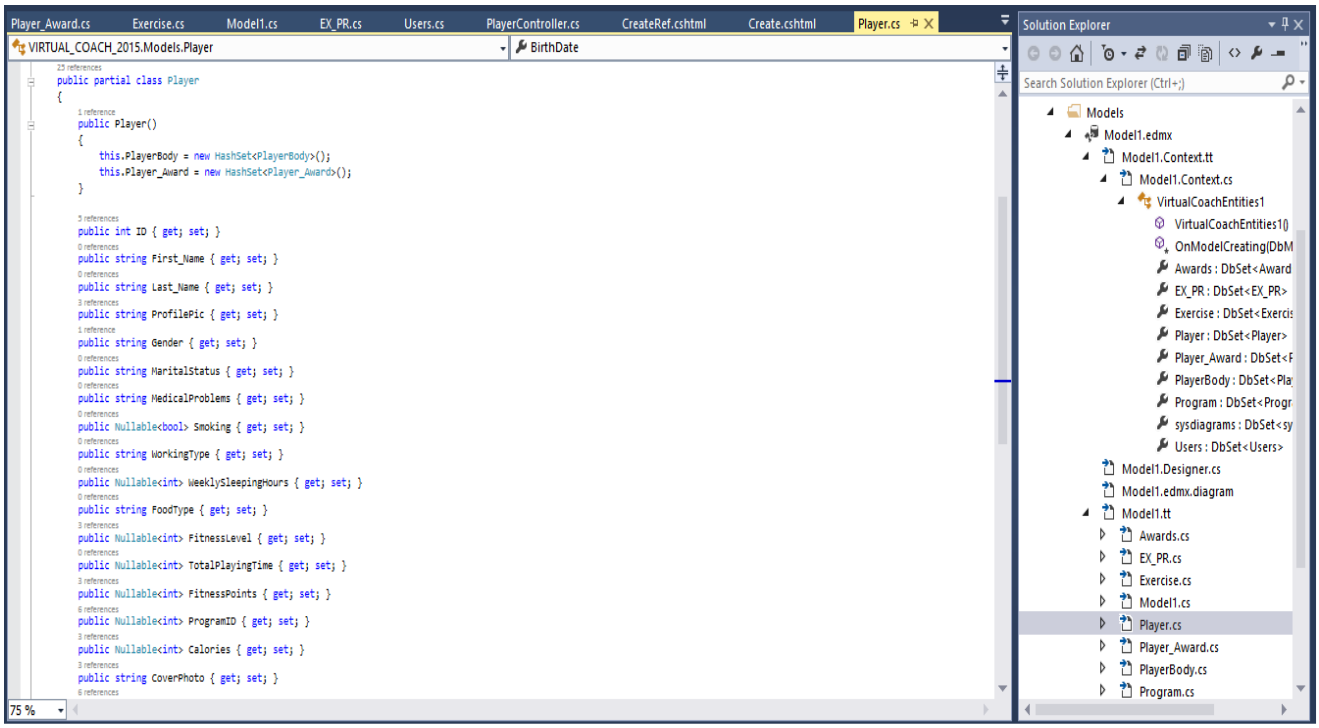
And for each exercise user must enter the first three count but count4, 5 are an optional choice When the user choose an exercise this exercise will be add automatically to his new program and will be appear in his home page

3.10 Implementation:

3.10.1Model:

The model manages the behavior and data of the application domain, responds to requests for information about its state (usually from the view), and responds to instructions to change state (usually from the controller).Model has many classes and its constructer and data member and methods needed to run system successfully mean it is a class that maps to the data relation (table) and potentially bridge tables (e.g. for many to many relations).The same class can have methods for the manipulations on the corresponding data, there could be additional classes that are not defining models by themselves, but the methods of accessing and filtering the data. However, term model in MVC does apply to describing data structures and the methods to access them in The same class can have methods for the manipulations on the corresponding data; there could be additional classes that are not defining models by themselves, but the methods of accessing and filtering the data, but term model in MVC does apply to describing data structures and the methods to access them in general.

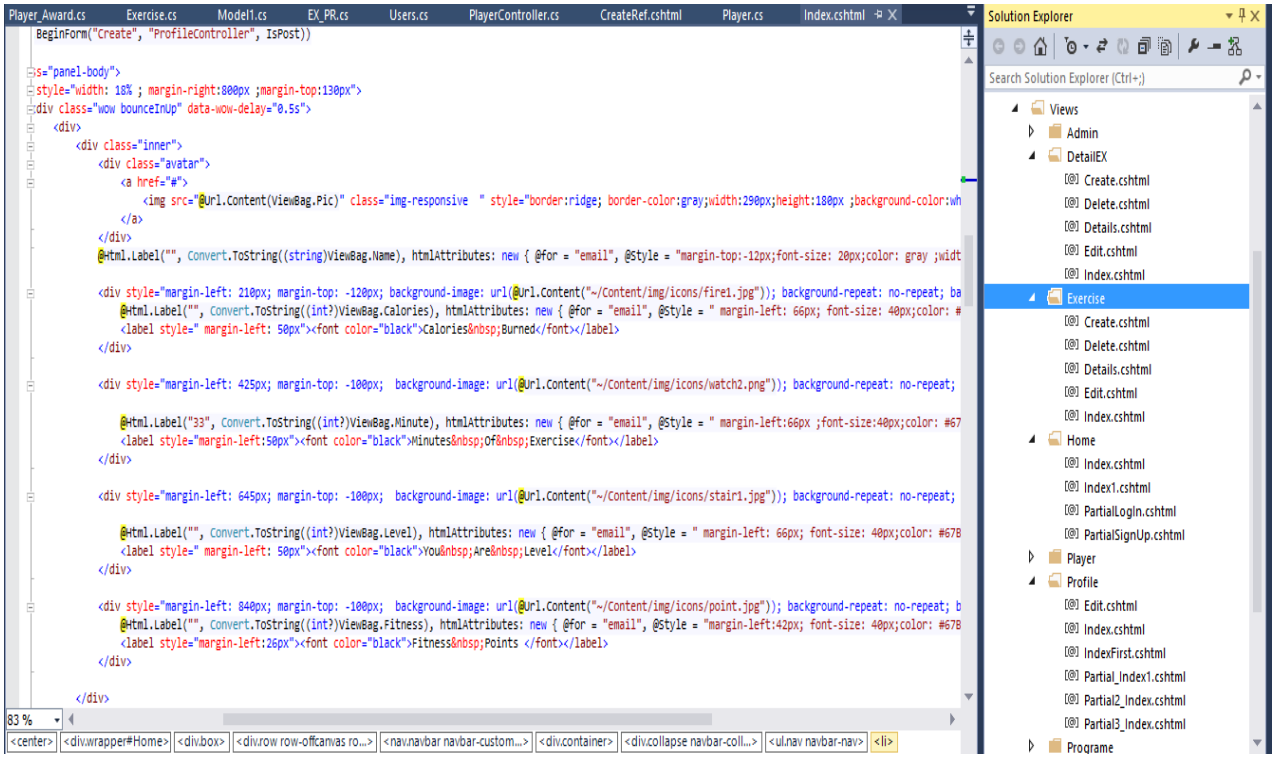
Figure 25 MVC models



3.10.2View:

The View role can be implemented in several ways. The popular view engines used in the ASP.NET MVC 3 Framework are the Razor View Engine and the Web Forms (ASPX) view engine. Other than that, a View can be implemented by accessing Controller methods using jQuery. This article illustrates all these ways of implementation of Views in an MVC application. In our view we user razor language to display UI and easy to use and can use c# inside it and make an easier communication between user and data should be represent to him. The Views folder stores the files (HTML files) related to the display of the application (the user interfaces). These files may have the extensions html, asp, aspx, cshtml, and vbhtml, depending on the language content. The Views folder contains one folder for each controller.

Figure 26 MVC Views

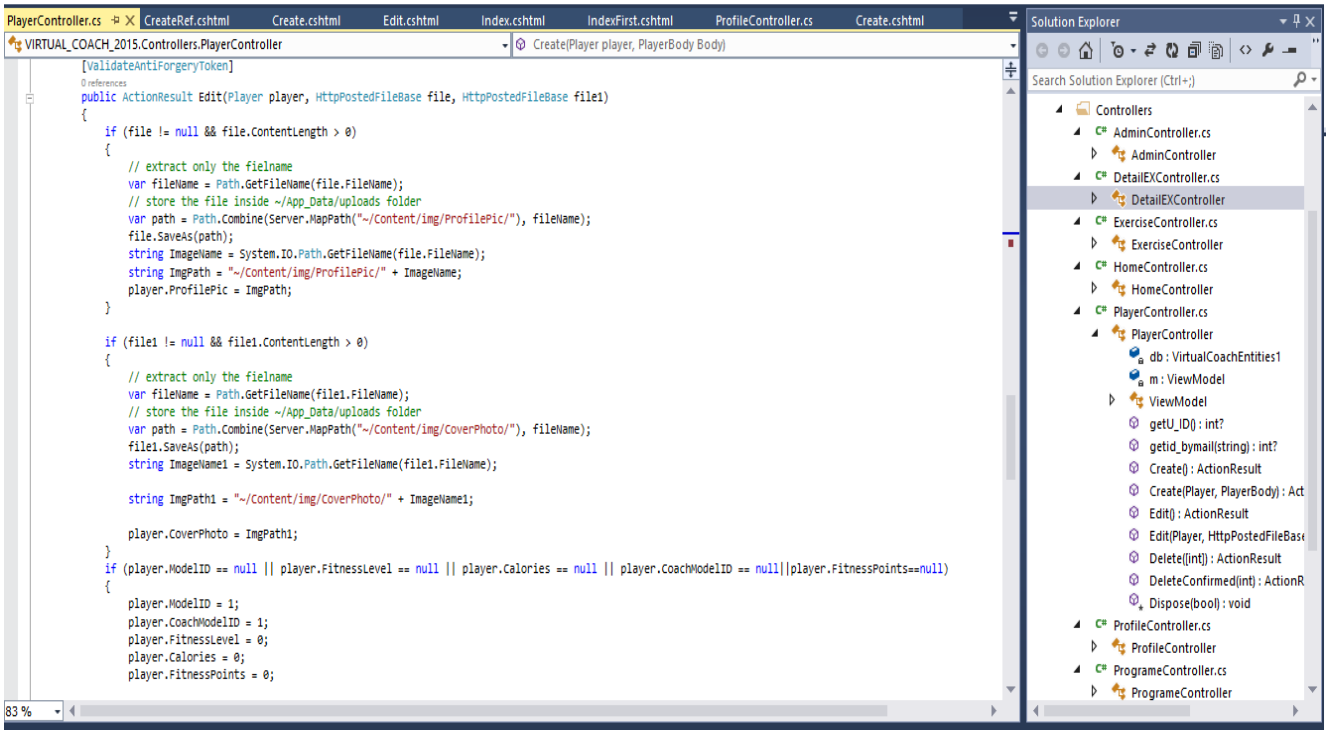


3.10.3Controller:

This refactoring uses the code-behind feature to adapt the model code to the data controls that exist on the page and to map the events that the controller forwards to the specific action methods. Because the model here returns a Dataset object, its job is straightforward. This code, like the view code, does not depend on how data is retrieved from the database.

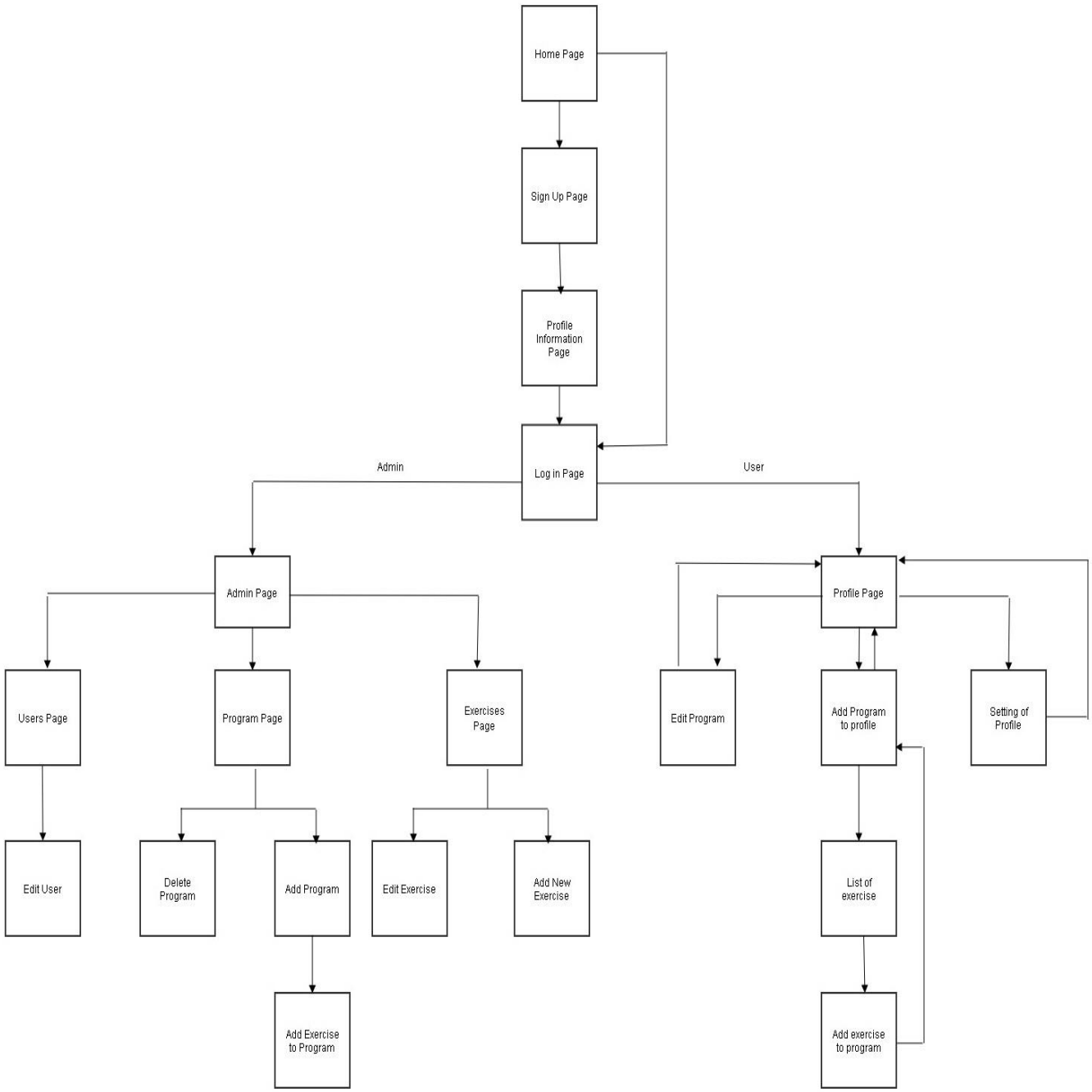
The Controllers Folder contains the controller classes responsible for handling user input and responses.MVC requires the name of all controllers to end with "Controller".

Figure 27 MVC Controllers



3.11 Navigation Diagram:

Figure 28 Navigation Diagram



4 Windows Store App

4.1 Overview

Officially released in October 2012, Windows 8 revolutionizes the user interface over other versions of Windows. Some of the new compelling features introduced include:

- The Start Screen
- The ability to run across many different types of hardware devices, from tablets based on Intel Atom processors to Ultrabook™ systems with Intel Core processors.
- Optimized for Touch while providing full keyboard and mouse support
- Apps from the Windows Store

On Windows 8, Windows Store apps can incorporate the following new UI/UX features, which we will describe in more detail later in this document:

- Live tiles
- Charms
- "Lock Screen" update
- App bar
- Snapped and fill view
- Semantic zoom

For application developers, these features can be implemented with the Visual Studio* 2012 and Windows Runtime APIs using familiar programming languages such as Visual C#*, JavaScript*, or Visual C++*.

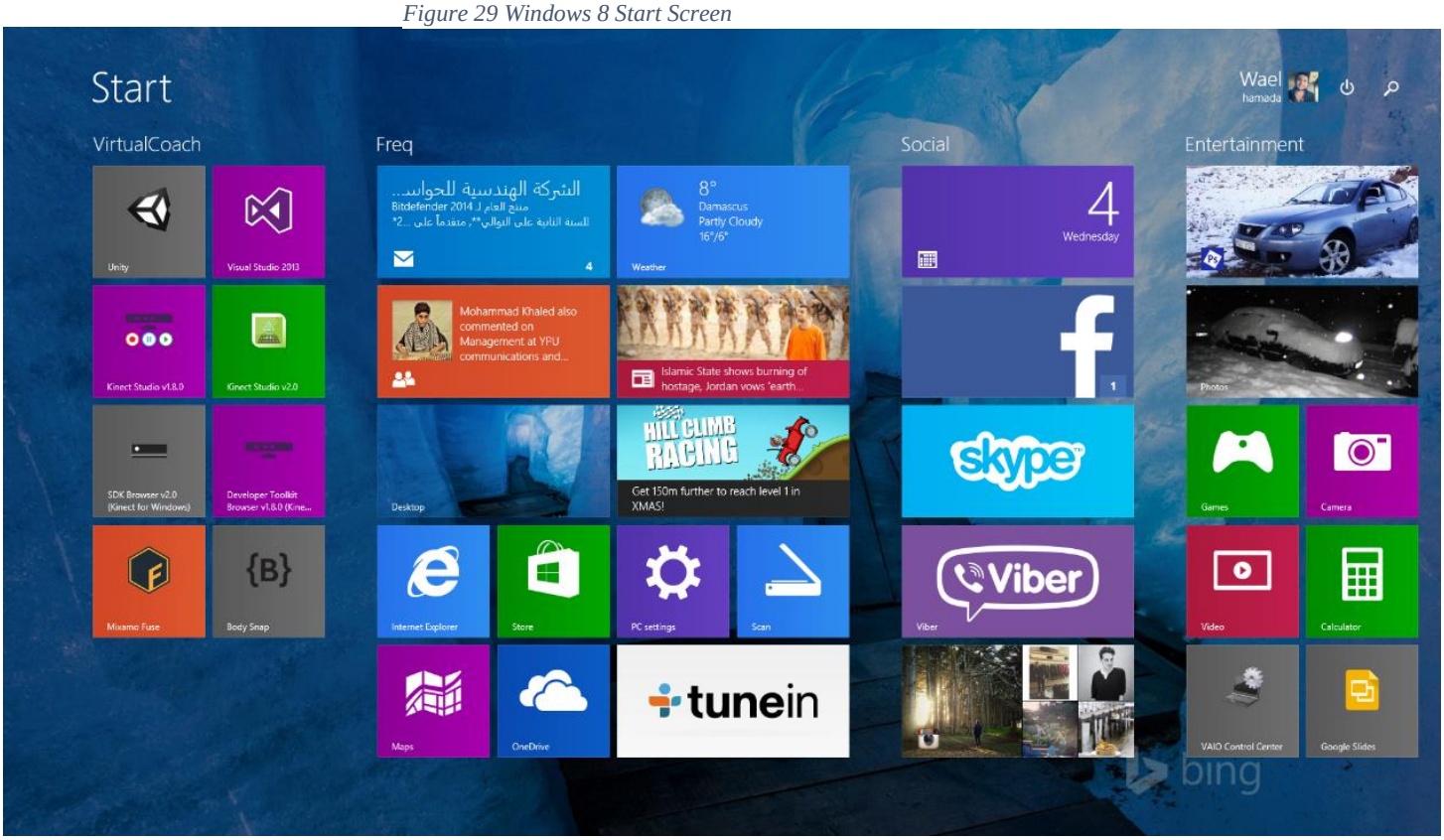
4.1.1 Live Tiles

On the Windows 8 Start screen, installed applications are depicted using icons or “Tiles.” Touching a tile or clicking on it with the mouse, launches the application.

On the Start screen, tiles can be presented in two sizes, a square (small) and a rectangle (large/wide). For applications that support both tile sizes, users can choose to display either. Tiles can either display a static image/icon or a dynamic “live” image that is updated via notifications from the application represented by the tile.

Tiles present rich content from the apps even when they are not the active application. When a user is on the Start screen, all applications that present a Live Tile, update the content while running in the background. For examples, a weather app can display your current local temperatures, and a finance app can show the current market snapshot.

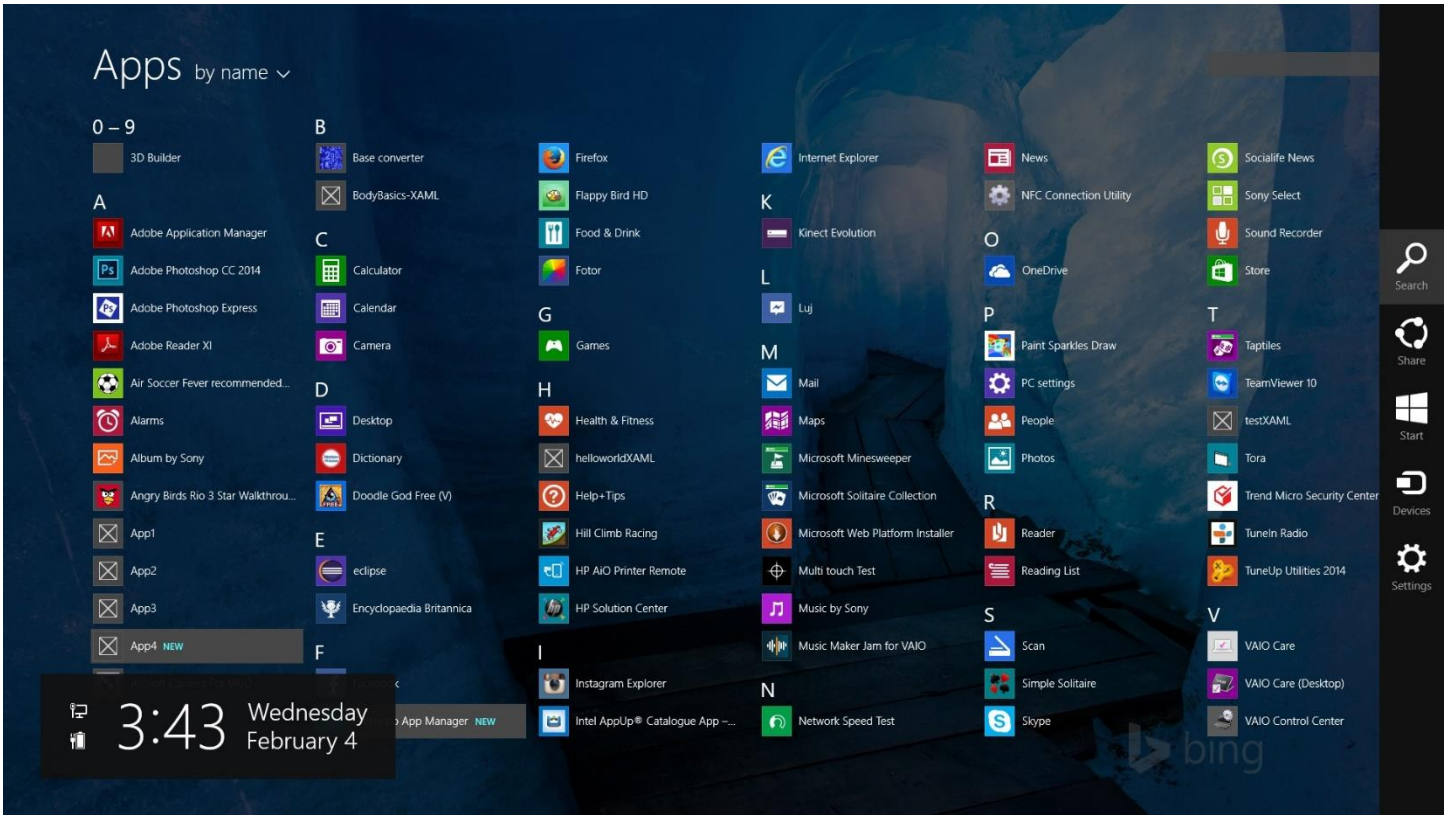
Figure 29 Windows 8 Start Screen



4.1.2 Charms

On Windows 8, at anytime and anywhere, if you swipe from the right edge into the screen or mouse point on the upper right corner or lower right corner, a vertical bar will appear on the right side of the screen showing additional functions, i.e., charms. They include Search, Share, Start, Devices, Settings, and more. Charms can be customized to include extra behaviors inside a specific app beyond what is available from the Start screen.

Figure 30 Windwos 8 Charms



4.1.3 Why Windows Store app (WinRT)

WinRT has emerged at the same time as Microsoft's "reimagining" of Windows into two new operating systems Windows 8 and Windows RT, although the timing that brings the launch of the new OSes and a new API model together is more luck than judgment. WinRT is about fixing the fundamental limitations of writing software natively for Windows. Native applications in Windows are written using the Win32 API, which is a very old, non−object-oriented API. Alongside Win32 we also have COM, an object-oriented subsystem that

allows for components to be plugged in and out of indows. If you're a relative newcomer to writing software for Windows, there's a good chance you've never used either of these, or you've used .NET. If you're slightly longer in the tooth, there is a chance that you did use these technologies once, but—especially if you've selected this book—the likelihood is that over the past n years you've been using .NET to write software that targets Windows OSes.

.NET is very different from Win32 or COM. .NET is a Java-inspired framework library and execution environment designed to make it easier to write software for Windows.

We call .NET a "managed code" environment because the runtime takes over a lot of the "management" of the code execution. Conversely, Win32 apps are "unmanaged." In its initial incarnation, .NET was built to let developers build websites in ASP.NET, or native applications with Windows Forms. (We'll ignore console applications or Windows services for the time being, as I want to talk about user interface technologies.) Both of these user interface technology tracks have evolved and changed over time, but regardless the more important thing about .NET was that it allowed developers to be more expressive. The Base Class Library (BCL) within .NET provided easy, object-oriented access either into Windows operating system features (e.g., System.IO.FileStream) or classes designed to save the developer time and effort (e.g., System.Collections.Generic.List). (The former set is relevant here, as a great deal of the BCL simply trunks down into Win32, which is how it provides access to OS functions.) In addition to the BCL, the CLR provides features like garbage collection so that under regular operations, developers do not need to worry about the mechanics of working with memory management and other important bits and pieces.

4.2 WCF Layer

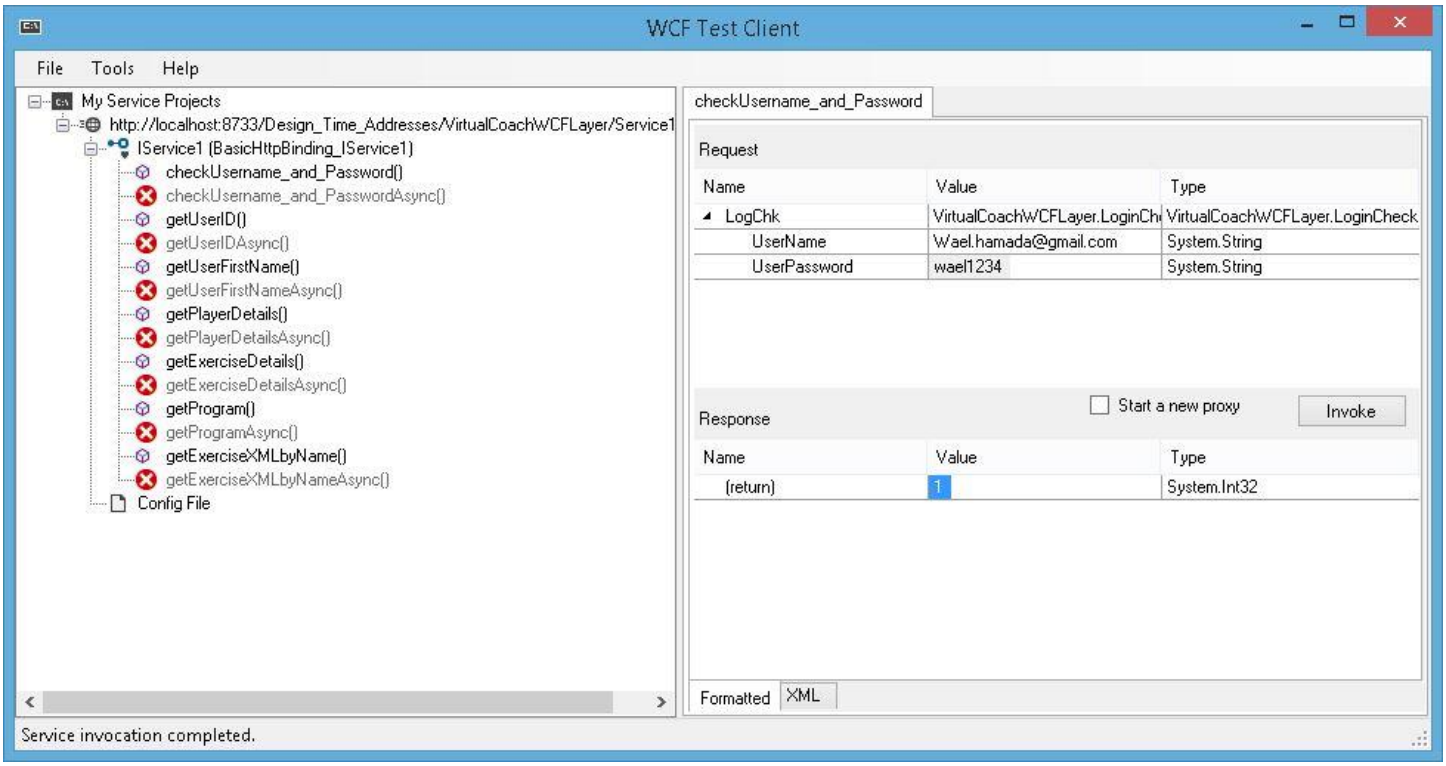
4.2.1 What is WCF?

Windows Communication Foundation (WCF) is a framework for building service-oriented applications. Using WCF, you can send data as asynchronous messages from one service endpoint to another. A service endpoint can be part of a continuously available service hosted by IIS, or it can be a service hosted in an application. An endpoint can be a client of a service that requests data from a service endpoint. The messages can be as simple as a single character or word sent as XML, or as complex as a stream of binary data. A few sample scenarios include:

- A secure service to process business transactions.
- A service that supplies current data to others, such as a traffic report or other monitoring service.
- A chat service that allows two people to communicate or exchange data in real time.
- A dashboard application that polls one or more services for data and presents it in a logical presentation.
- Exposing a workflow implemented using Windows Workflow Foundation as a WCF service.
- A Silverlight application to poll a service for the latest data feeds.

While creating such applications was possible prior to the existence of WCF, WCF makes the development of endpoints easier than ever. In summary, WCF is designed to offer a manageable approach to creating Web services and Web service clients.

Figure 31 WCF Test



4.3 Virtual Coach App

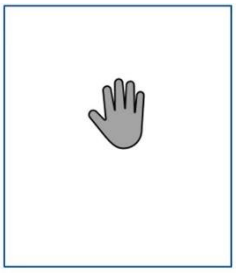
4.3.1 Overview:

This app is a windows store app that allows the user to login into his profile and check his daily sport program. If the user have the Kinect sensor it would be a plus for him as he will be able to track his body movements to archive the most accurate exercise. In addition, sensor owners would feel free to use the application without touching their mouse or keyboard because the Kinect sensor will doubles to also be used as an input device

4.3.2 Hand point gestures:

Figure 32 Engagement

Figure 33 Targeting



- Engagement: it describes that movement which tells the system that Yes I want to engage and interact
- Targeting: once you are engaged, the cursor will be shown and you can move you hand naturally in the space and the cursor will follows your hand movement
- Pressing: it focusing on trying to find the closest user interface element by matching the (X,Y) Position of the cursor with those UI elements, once it detects an element targeting would stop (freeze) the cursor in its position and trying to trigger the onClick() Event of that element as the User pushes his hand to the screen.
- User viewer: is a great form of feedback, which helps the users to understand what is happening in the scene, located in the top right corner of every page in the Virtual Coach app.

Figure 34 UserView

4.3.3 User interfaces:

4.3.3.1 Login Screen

This screen will allows the user to input his username and password to login into the system. User input can be done using keyboard and mouse or onscreen keyboard using the hand pointer.

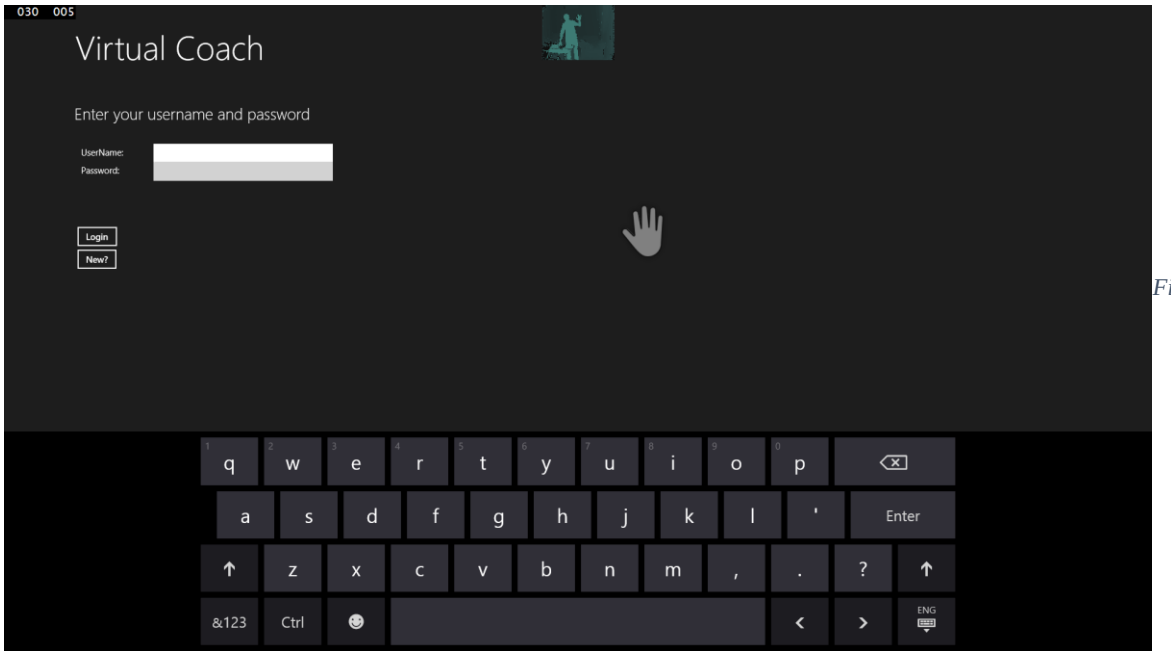


Figure 36 Dark Theme

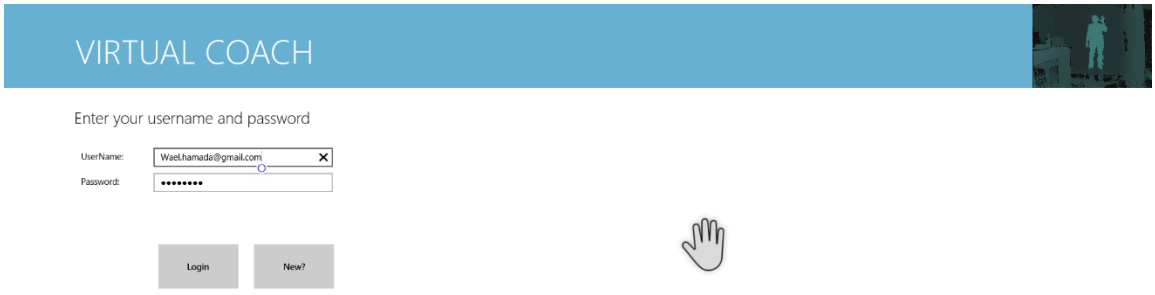


Figure 35 Light Theme

4.3.3.2 Welcome screen (Home screen)

The home screen is one of the most important interface the user will see, however. We will discuss each part separately.

Progress bar

The progress bar shows the player the following information:

- total playing time
- amount of calories burned
- fitness points
- fitness level

Figure 37 Progress bar

4.3.3.2.1 3D Mode toggle button

For fantasy player who loves to feel awesome while playing, we introduced a 3D mode, but required 3D glasses

Figure 38 3D Mode Button

4.3.3.2.2 Tracking Mode toggle button

Tracking is one of the most important feature but for those players without the Kinect sensor they still can be able to user the app to just preview the movement.

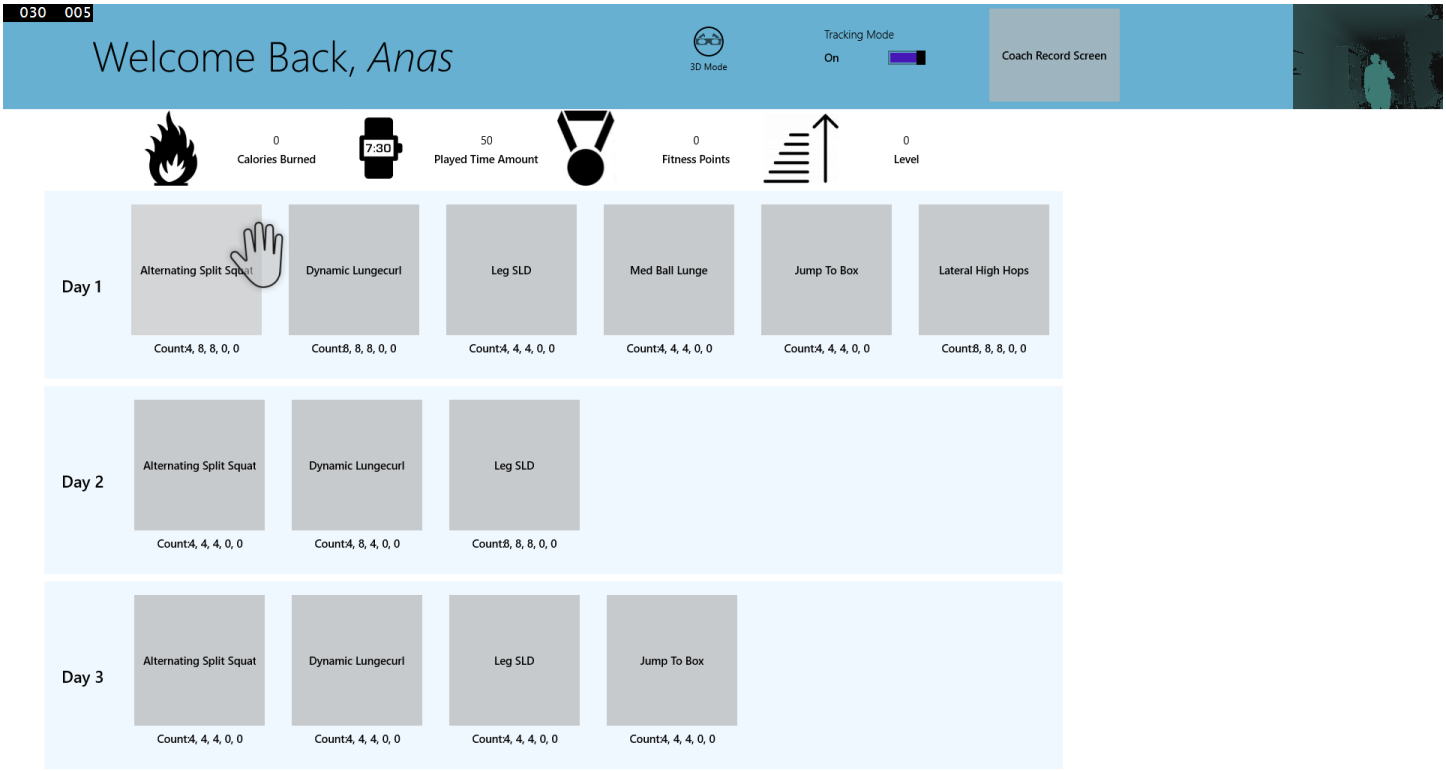
4.3.3.2.3 Record Button

Sometime, the player could also be a coach, in which he could like to submit a new exercise to the system, to do that we provided a button at the top of the screen but however it will only visible to coaches who have the permission to record a new exercise.

4.3.3.2.4 Daily exercises

In this section, VirtualCoach will generate a number of exercises according to the player program chosen previously in the website, however. When the player want to play an exercise, he can use the mouse or the hand pointer to push the button and VirtuaCoach will navigate him to the unity3D Screen.

Figure 39 Home Screen

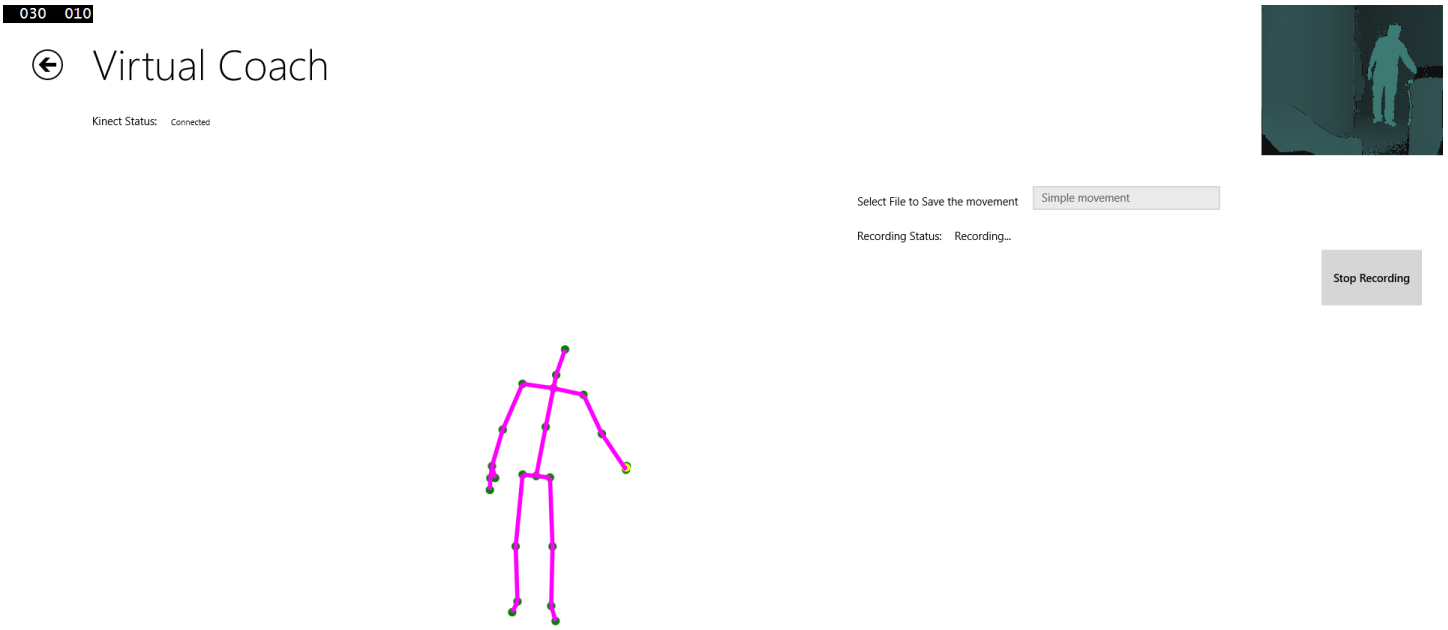


4.3.3.3 Record screen

This screen is allowed to access by coaches only; normal players will not have the ability to access it, as the recording button in his home screen will not be visible.

Recording screen is very simple to use, it shows the Kinect Status which will help the coach to determine if any connection error occurs. To record the coach must first assign a movement title and gets in his initial position and he will see an avatar generated at the bottom of the screen.

Figure 40 Record Screen



Once the coach hits the stop recording button, the virtual coach will save his previous movement to an XML file named according to the title he entered previously with a 30FPS (33ms) recording speed.

For each frame, the xml file contains a set of joints (AnkleLeft, AnkleRight, ElbowLeft, ElbowRight, FootLeft, FootRight, HandLeft, HandRight, HandTipLeft, HandTipRight, Head, HipLeft, HipRight, KneeLeft, KneeRight, Neck, ShoulderLeft, ShoulderRight, SpineBase, SpineMid, SpineShoulder, ThumbLeft, ThumbRight, WristLeft, WristRight).

Moreover, for each one of them we will record its X-Position, Y-Position, Z-Position, in addition to the Quaternion values for (X, Y, Z, W).

Figure 41 XML Movement Preview

```
<VirtualCoach_Movement TotalFrames="121">
  <Frame Number="1">
    <AnkleLeft X="-0.180738523602486" Y="-0.822494924068451" Z="2.70843338966637" RX="0.770057320594788" RY="-0.0452661290764809" RZ="-0.636322617530823" RW="0.0074939
    <AnkleRight X="0.0118336258456111" Y="-0.821476697921753" Z="2.72098994255066" RX="-0.0429265983402729" RY="-0.00362407322973013" RZ="0.99901294708252" RW="-0.010
    <ElbowLeft X="-0.295864522457123" Y="0.175682842731476" Z="2.74641060829163" RX="0.714215934276581" RY="-0.177314221858978" RZ="-0.67182731628418" RW="-0.08428142
    <ElbowRight X="0.228115037083626" Y="0.146820053458214" Z="2.75304317474365" RX="0.604280054569244" RY="0.106454849243164" RZ="0.787126004695892" RW="-0.062813527
    <FootLeft X="-0.165469512343407" Y="-0.888964056968689" Z="2.61354374885559" RX="0" RY="0" RZ="0" RW="0" />
    <FootRight X="0.0579600520431995" Y="-0.883280158042908" Z="2.72908115386963" RX="0" RY="0" RZ="0" RW="0" />
    <HandLeft X="-0.380842417478561" Y="-0.123834945261478" Z="2.66069269180298" RX="0.491026997566223" RY="-0.132787376642227" RZ="0.860175132751465" RW="-0.03686191
    <HandRight X="0.292618006467819" Y="-0.133179247379303" Z="2.67976188659668" RX="0.921687424182892" RY="-0.127440705895424" RZ="0.332564979791641" RW="-0.15379106
    <HandTipLeft X="-0.37433585524559" Y="-0.189192160964012" Z="2.65653324127197" RX="0" RY="0" RZ="0" RW="0" />
    <HandTipRight X="0.284705221652985" Y="-0.205923616886139" Z="2.67214798927307" RX="0" RY="0" RZ="0" RW="0" />
    <Head X="0.00372247816994786" Y="0.71244865655899" Z="2.70286893844604" RX="0" RY="0" RZ="0" RW="0" />
    <HipLeft X="-0.126947194337845" Y="-0.0987286195158958" Z="2.66927909851074" RX="-0.647857367992401" RY="0.721730589866638" RZ="-0.164062142372131" RW="0.18019269
    <HipRight X="0.0306278355419636" Y="-0.114268057048321" Z="2.67368865013123" RX="0.71596485376358" RY="0.662758111953735" RZ="-0.146586328744888" RW="-0.163273900
    <KneeLeft X="-0.15974648296833" Y="-0.475130796432495" Z="2.68431186676025" RX="0.716402649879456" RY="-0.0450095795094967" RZ="-0.696050286293303" RW="-0.01598404
    <KneeRight X="0.00486907502636313" Y="-0.504466474056244" Z="2.72299098968506" RX="0.695766627788544" RY="0.0220796894282103" RZ="0.714775562286377" RW="0.0672087
    <Neck X="-0.0188220981508493" Y="0.561047971248627" Z="2.71186947822571" RX="0.021056491881609" RY="0.999255537986755" RZ="-0.026520024985075" RW="0.0184838846325
    <ShoulderLeft X="-0.189280733466148" Y="0.44406059384346" Z="2.71213793754578" RX="0.777630805969238" RY="-0.628540754318237" RZ="0.0147468606010079" RW="-0.00305
    <ShoulderRight X="0.16047228872776" Y="0.435012966394424" Z="2.72579097747803" RX="0.786687076091766" RY="0.616816163063049" RZ="0.0146899418905377" RW="0.0211055
    <SpineBase X="-0.0486131757497787" Y="-0.108272232115269" Z="2.71133255958557" RX="0.0233982261270285" RY="0.999563038349152" RZ="0.0104504786431789" RW="0.014731
    <SpineMid X="-0.0328166596591473" Y="0.231224432587624" Z="2.71866893768311" RX="0.0234219543635845" RY="0.999526917934418" RZ="0.0103971976786852" RW="0.01700612
    <SpineShoulder X="-0.0223242994397879" Y="0.479905247688293" Z="2.71611666679382" RX="0.0209802594035864" RY="0.999585330486298" RZ="-0.0055262646637857" RW="0.01
    <ThumbLeft X="-0.356011092662811" Y="-0.193674549460411" Z="2.65559554100037" RX="0" RY="0" RZ="0" RW="0" />
    <ThumbRight X="0.313758432865143" Y="-0.134633421897888" Z="2.64163160324097" RX="0" RY="0" RZ="0" RW="0" />
    <WristLeft X="-0.375373571211113" Y="-0.0452977679669857" Z="2.68229746818542" RX="0.485983699560165" RY="-0.200149834156036" RZ="0.84716260433197" RW="0.077944204
    <WristRight X="0.300400078296661" Y="-0.0791855975985527" Z="2.70136952400208" RX="0.919294118881226" RY="0.103516988456249" RZ="0.346737235784531" RW="-0.1547766
  </Frame>
  <Frame Number="2">
    <AnkleLeft X="-0.179347619414333" Y="-0.823258519172668" Z="2.71010899543762" RX="0.760846614837646" RY="-0.0444393903017044" RZ="-0.647341549396515" RW="0.0093038
    <AnkleRight X="0.011864255182445" Y="-0.821869611740112" Z="2.72106266021729" RX="-0.0482393428683281" RY="-0.0039213914424181" RZ="0.998761653900146" RW="-0.0115
    <ElbowLeft X="-0.296455353498459" Y="0.174754977226257" Z="2.74804735183716" RX="0.723675072193146" RY="-0.177537113428116" RZ="-0.661982893943787" RW="-0.0809544
    <ElbowRight X="0.228015586733818" Y="0.146122246980667" Z="2.75244307518005" RX="0.602029263973236" RY="0.106215462088585" RZ="0.788834869861603" RW="-0.063392549
    <FootLeft X="-0.166290819644928" Y="-0.88848340511322" Z="2.61301970481873" RX="0" RY="0" RZ="0" RW="0" />
    <FootRight X="0.0585635118186474" Y="-0.883712947368622" Z="2.72866868972778" RX="0" RY="0" RZ="0" RW="0" />
    <HandLeft X="-0.377711057662964" Y="-0.116414651274681" Z="2.66961765289307" RX="0.0107432082295418" RY="-0.119934268295765" RZ="0.992298483848572" RW="0.02905100
    <HandRight X="0.294036448001862" Y="-0.155807480216026" Z="2.67941951751709" RX="0.952804684638977" RY="-0.0780234709382057" RZ="0.253577351570129" RW="-0.1475602
    <HandTipLeft X="-0.374864608049393" Y="-0.190492674708366" Z="2.6621880531311" RX="0" RY="0" RZ="0" RW="0" />
    <HandTipRight X="0.285115480422974" Y="-0.209582507610321" Z="2.6774799823761" RX="0" RY="0" RZ="0" RW="0" />
```

4.3.4 Unity Connection

4.3.4.1 Windows 8 sandbox

4.3.4.1.1 Sandbox overview

Since the dawn of personal computing, security and reliability have been an issue. In Microsoft Windows this has been primarily due to the decision to make security and reliability the responsibility of the software running on Windows. Windows Desktop software runs by default with Full Trust and full access to all the resources and capabilities of the computer. The upside of this is that incredibly powerful and rich applications can be created. The downside is that a poorly written or poorly tested application can crash the whole computer resulting in the dreaded "Blue Screen of Death". The openness of platforms like Desktop Windows and Google Android, while providing great benefits, has also given rise to Viruses, Malware, Trojans, Rootkits and other nasty pieces of software that have caused untold amounts of frustration, embarrassment, lost revenue, lost productivity. Here are two recent examples:

- Multiple Android apps downloaded millions of times contained code to mine crypto-currencies like Bitcoin without the user's knowledge.
- An SMS Trojan that calls premium-rate phone numbers is spreading around the world in Android apps.

In response, a multi-billion industry has arisen to combat all of this criminal activity resulting in most computers requiring Anti-Virus, Anti-Malware and other types of security software to keep our data safe. In essence, a war has been waging for decades on our computers with all of us caught in the middle. Both Apple and Microsoft have taken a different approach; with the lifecycle of iOS and Windows Store apps being strictly controlled by the Operating System. The rest of this article will explore the different facets of this in the context of Windows Store apps since I do not have any experience with Apple's products.

4.3.4.1.2 Windows store sandboxing

All Windows Store apps are tightly sandboxed. This means Windows Store apps run in their own virtual space (the sandbox) and whatever happens to it does not affect any other app running or the OS itself. It should be practically impossible for a Windows Store App to crash the entire computer, it may still crash itself but it won't be able to hurt anything else. Being in the Sandbox also means the app has no direct access to any other app or service running outside of the app's sandbox. Access to other apps or services is facilitated by Windows itself with a defined set of APIs with in the runtime environment. While this does place limits on what a Windows Store app can do the tradeoff is worth it because it should never be possible for a Windows Store app to be a Virus, Trojan or Rootkit.

4.3.4.1.3 Declaring Resources and Capabilities

Unlike traditional Desktop software which by default run with Full Trust and have access to all the resources and capabilities of the OS a Windows Store App runs in a "Least Privileged" mode and must declare all the resources and capabilities needed at the time it is made available in the Windows Store. When installed, Windows will prevent the app from accessing any resources or capabilities it has not declared. In addition for certain declared resources and capabilities, like a camera or location services, Windows will actually request the permission from the user to use that resource or capability the first time a Windows Store App tries to access them. Finally certain actions, for example opening a file, are actually controlled by Windows itself. A Windows Store app, for the most part, does not have direct access to the file system; instead it requests that Windows present a File Picker dialog to the user and Windows will return the selected file to the App. All of this should make it impossible for a Windows Store app to do or access anything without the user being aware of and ultimately in control of it.

4.3.4.1.4 URI (uniform resource identifier)

A URI is a compact representation of a resource available to your application on the intranet or Internet. The Uri class defines the properties and methods for handling URIs, including parsing, comparing, and combining. The Uri class properties are read-only; to create a modifiable object, use the UriBuilder class.

Relative URIs (for example, "/new/index.htm") must be expanded with respect to a base URI so that they are absolute. The MakeRelative method is provided to convert absolute URIs to relative URIs when necessary.

The Uri constructors do not escape URI strings if the string is a well-formed URI including a scheme identifier.

The Uri properties return a canonical data representation in escaped encoding, with all characters with Unicode values greater than 127 replaced with their hexadecimal equivalents. To put the URI in canonical form

4.3.4.1.5 Overcoming the sandbox

Figure 42 BatRT Logo

In order to allow VirtualCoach to invoke the unity scene we had to overcome the sandbox, after many tries we have finished using BatRT. BatRT is an application which allows a batch file to executed from a WinRT application using the URI loop hole, This allows a store app to compose a batch file within its application folder and then have that executed outside of the WinRT architecture – bypassing the constraints of WinRT.

4.3.4.2 Reading XML file:

To move the coach avatar in unity we should first read the recorded values and pass them to the unity scene. To archive that need we implemented a Class Called “VCM1” which represents one single frame

```
public class VCM1
{
    public Dictionary<JointType, Vector3> Coach_joints;
    public Dictionary<JointType, Quaternion> Coach_jointOrientations;
    1 reference
    public VCM1()
    {
        Coach_joints = new Dictionary<JointType, Vector3>();
        Coach_jointOrientations = new Dictionary<JointType, Quaternion>();
    }
}
```

And to allow the unity scene to ready any frame in the xml at any time we implements a function which reads the xml file tags and returns a Dictionary indicates the wanted frame

```
public static Dictionary<int, VCM1> ReadBodyJointsfromXML(string file)...
```

4.3.4.3 Avatar Matching

After we have successfully read the skeleton data from the xml file to the frames Dictionary, then it became easier to retrieve those data and compare them to the live data. But here is a number of questions arise such as what data to compare? How to calculate those data? How can we achieve this in real time? We will answer these questions as we move along. Initially, we calculated joint angles of every frame in the recorded motion and save them to a list of joint angles. We can calculate vectors joining joints coordinates then from those vectors, we can calculate angles formed between them with respect to their directions.

4.3.4.3.1 Joint Angle Calculation

From 25 body joints in which Kinect recognizes, we can calculate vectors joining these points and hence angle between those vectors with respect to their direction. The method to calculate angle between joints is explained here:

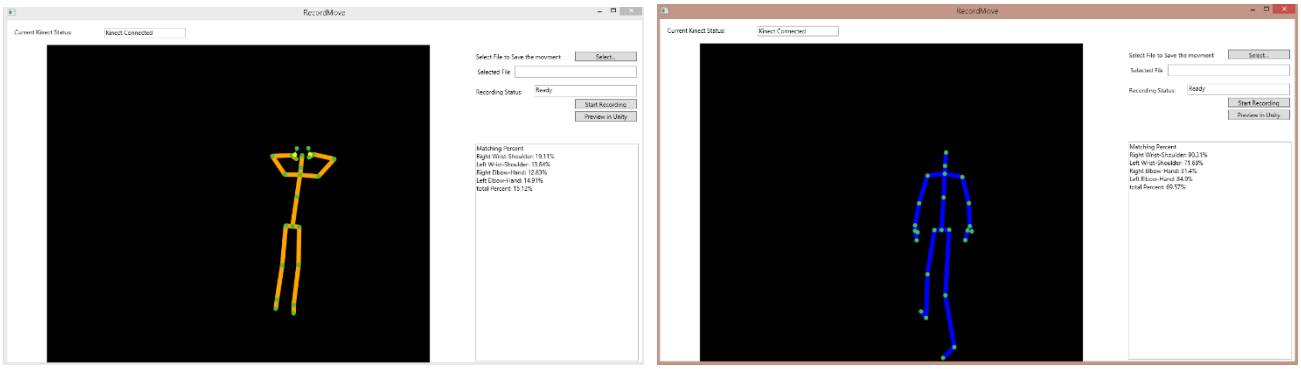
1. Calculate the vector joining every two joints points
2. Normalize the above vectors to their unit vectors
3. Find a dot product after normalizing them
4. Find the cross product between them
5. Calculate the angle formed using atan2 in radians
6. Calculate angle formed in degree
7. Round the angle into two decimal places

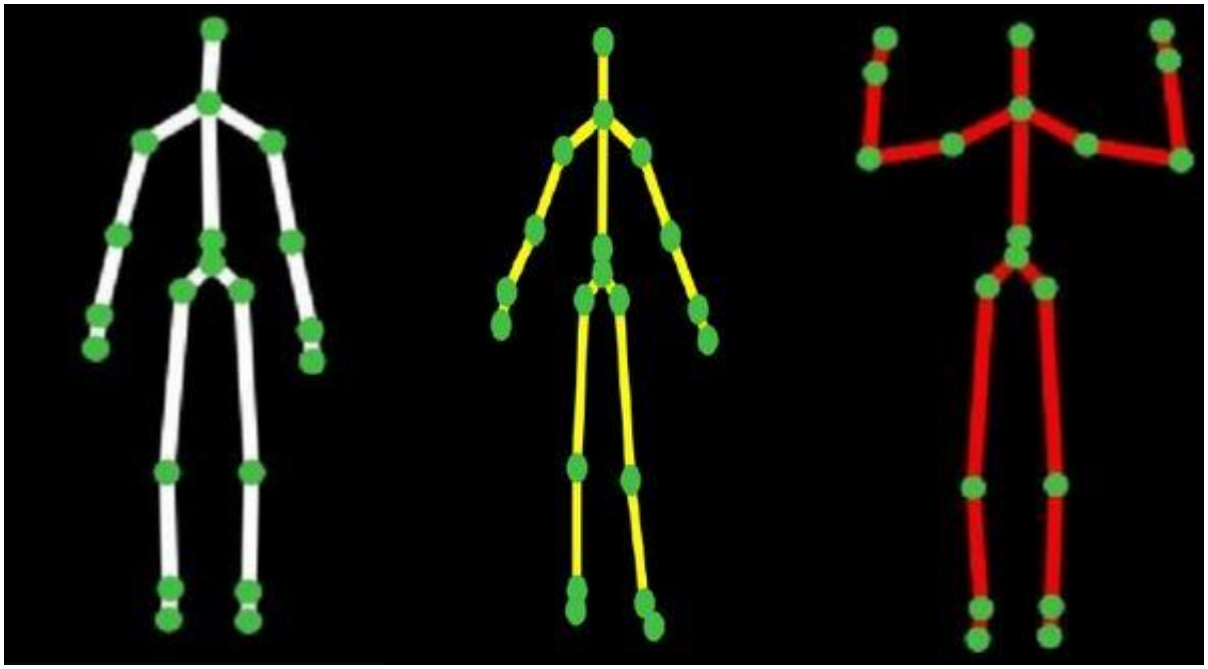
4.3.4.3.2 How to compare?

Now we have calculated angle throughout the body for each frame and saved them to the list data structure. Thus, we need to retrieve these angles and compare them with the live ones for each frame. Small deviations of angles are negligible. We have introduced a threshold of +/- 30 degree. Any angle that lies between this boundary is considered as a successful match.

We have established the mechanism that if live-skeletons match with at least 70% of total saved-skeletons, then the whole motion is considered to be imitated successfully. The user is notified at real time whether he is performing the right way.

Figure 43 WPF Matching Test





Joint	**Percent Match with the yellow skeleton**	**Percent Match with the red skeleton**
Right Wrist-Shoulder	100%	38.82%
Left Wrist-Shoulder	98.68%	46.71%
Right Elbow-Shoulder	100%	100%
Left Elbow-Shoulder	100%	99.34%
Right Shoulder-Spine	100%	100%
Left Shoulder-Spine	100%	90.79%
Right Wrist-Spine	98.03%	54.61%
Left Wrist-Spine	100%	92.76%
Right Finger-Elbow	100%	91.45%
Left Finger-Elbow	100%	96.05%
Right Ankle-Hip	100%	87.5%
Left Ankle-Hip	100%	96.05%
Right Knee-Hip	100%	100%
Left Knee-Hip	100%	100%
Right Hip-Spine	100%	100%
Left Hip-Spine	100%	100%

5 3D Modeling Section for VirtualCoach

5.1 Abstract

The project studies the 3D modeling of interactive technology in sport. The visualization has been developed in using Unity3D game engine to provide real time interactivity. A set of visualization parameters have been identified by the Microsoft Kinect version 2 and adjusted by graphical user interface to provide artistic control of the visualization. The project is part of a collaboration with Hasan Eskandarani and Ali Nawashy who developed the rest part (Kinect + Website + Database+ Windows Store).

5.2 Introduction

5.2.1 Overview

The project explored the functionality of using Unity3D game engine as a means of providing interactivity between a dancer and a creative visualization. The project utilized material learnt from the following courses at university:

- Computer Graphic (studied major contemporary graphics)
- Virtual Reality (exploring C# in Unity)
- Mathematics 2 (Vectors and its operations)

5.2.2 Background

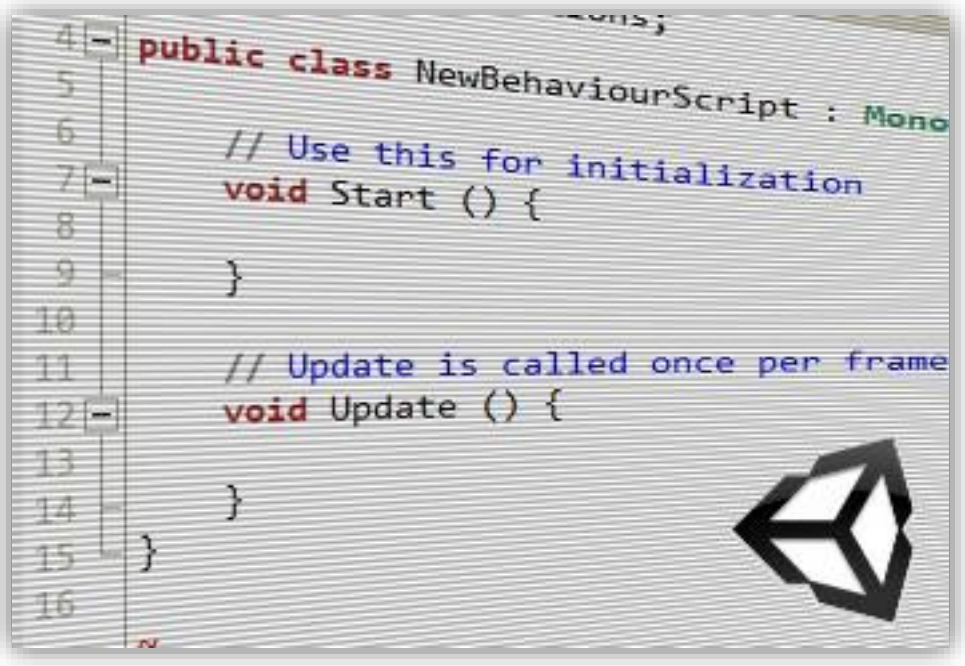
5.2.2.1 Unity (Game Engine)

Figure 44 Unity3D Logo

Unity is a cross-platform game creation system developed by Unity Technologies, including a game engine and integrated development environment (IDE). It is used to develop video games for web sites, desktop platforms, consoles, and mobile devices. First announced only for Mac OS, at Apple's Worldwide Developers Conference in 2005, it has since been extended to target more than fifteen platforms. It is now the default software development kit (SDK) for the Nintendo Wii U.

5.2.2.1.1 Features

With an emphasis on portability, the graphics engine targets the following APIs: Direct3D on Windows and Xbox 360; OpenGL on Mac, Windows, and Linux; OpenGL ES on Android and iOS; and proprietary APIs on video game consoles. Unity allows specification of texture compression and resolution settings for each platform the game supports, and provides support for bump mapping, reflection mapping, parallax mapping, screen space ambient occlusion (SSAO), dynamic shadows using shadow maps, render-to-texture and full-screen post-processing effects. Unity's graphics engine's platform diversity can provide a shader with multiple variants and a declarative fallback specification, allowing Unity to detect the best variant for the current video hardware; and if none are compatible, fall back to an alternative shader that may sacrifice features for performance.



The game engine's scripting is built on Mono, the open-source implementation of the .NET Framework. Programmers can use UnityScript (a custom language with ECMAScript-inspired syntax, referred to as JavaScript by the software), C#, or Boo (which has a Python-inspired syntax.

The engine has built-in map editor and code editor, which allows for a standard experience on users using it on both Windows and Linux.

5.2.2.2 Kinect Unity Plug-in:

This plug-in is a **unitypackage** that provides the interconnection between the Kinect for Windows and the Unity3D game engine. It makes the skeleton information (joints type, joint position, and joint orientation) available for controlling the avatar. Also the plug-in supplies the speech recognition for more interaction. In addition, it is faster, cost-efficient, and high quality support for cross-platform development, enabling developers to build their apps for the Windows Store (as in Virtual Coach requirements) using tools they already know.

5.2.2.3 MakeHuman:

Makehuman is an open source 3D computer graphics software middleware designed for the prototyping of photo realistic humanoids. It is developed by a community of programmers, artists, and academics interested in 3D modeling of characters. MakeHuman is developed using 3D morphing technology. Starting from a standard (unique) androgynous human base mesh, it can be transformed into a great variety of characters (male and female), mixing them with linear interpolation. For example, given the four main morphing targets (baby, teen, young, old) it is possible to obtain all the intermediate shapes.

Figure 45 MakeHuman Logo

5.2.2.4 3D Max Studio:

Autodesk 3ds Max, formerly 3D Studio Max, is a professional 3D computer graphics program for making 3D animations, models, games and images. It is developed and produced by Autodesk Media and Entertainment. It has modeling capabilities, a flexible plugin architecture and can be used on the Microsoft Windows platform. It is frequently used by video game developers, many TV commercial studios and architectural visualization studios. It is also used for movie effects and movie pre-visualization. One of its features is Character Studio.

Character Studio was a plugin, which since version 4 of Max is now integrated in 3D Studio Max, helping users to animate virtual characters. The system works using a character rig or "Biped" skeleton, which has stock settings that can be modified and customized to the fit character meshes and animation needs. This tool also includes robust editing tools for IK/FK switching, Pose manipulation, Layers and Keyframing workflows, and sharing of animation data across different Biped skeletons. These "Biped" objects have other useful features that help accelerate the production of walk cycles and movement paths, as well as secondary motion.

Figure 46 3D Studio Max Logo

5.2.2.5 Programming Language

Unity provides three programming languages –C#, JavaScript and Boo. Since the Virtual Coach software is a Windows store application developed in C#, I decided to use C# because it’s a programming language which I have experience in rather than Boo, and I wanted to exploit C# which provides a larger API which Kinect Unity Plug-in supports.

5.2.2.6 Coordinate Systems:

There are many coordinate systems, but there are two common ones. One is the right-handed coordinate system, which has the x-axis (from left to right), y-axis (from bottom to up) and the z-axis (from forward to backward). Positive rotation is **counterclockwise** about the axis of rotation. Another is the left-handed coordinate system that has the x-axis (from left to right), y-axis (from bottom to up) and the z-axis (from backward to forward). Positive rotation is **clockwise** about the axis of rotation. In fact, all the modeling soft wares use right-handed system (Kinect one of them) and Unity uses the left-handed system. Therefore, I faced a problem when applying the skeletal information given from Kinect in Unity. More information discussed in chapter 4.

5.2.2.7 Mathematics

5.2.2.7.1 Quaternion:

Kinect uses the quaternion to represent the joints orientations, so what is a quaternion?

A quaternion is a set of 4 numbers, [x y z w], which represents rotations the following way:

x= RotationAxis.x* sin(RotationAngle/2)

y= RotationAxis.y* sin(RotationAngle/2)

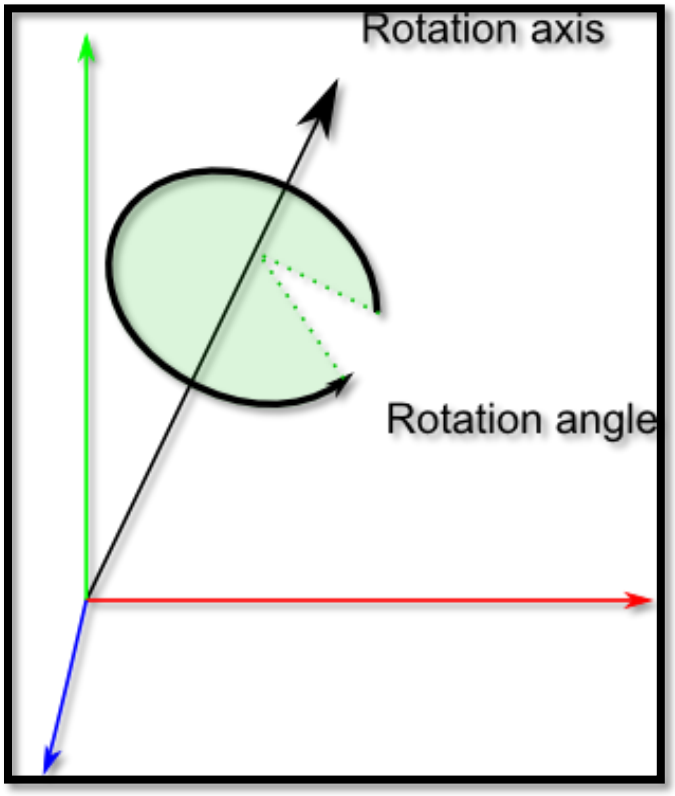
z= RotationAxis.z* sin(RotationAngle/2)

w=cos(RotationAngle/2)

RotationAxis is, as its name implies, the axis around which you want to make your rotation.

RotationAngle is the angle of rotation around this axis.

Figure 47 Rotation axis - Rotation angle



So essentially quaternions store a *rotation axis* and a *rotation angle*, in a way that makes combining rotations easy.

In mathematics, the **quaternions** are a number system that extends the complex numbers. They were first described by Irish mathematician William Rowan Hamilton in 1843 and applied to mechanics in three-dimensional space. A feature of quaternions is that multiplication of two quaternions is no-commutative. Hamilton defined a quaternion as the quotient of two directed lines in a three-dimensional space or equivalently as the quotient of two vectors.

Quaternions find uses in both theoretical and applied mathematics, in particular for calculations involving three-dimensional rotations such as in three-dimensional computer graphics, computer vision and crystallographic texture analysis. In practical applications, they can be used alongside other methods, such as Euler angles and rotation matrices, or as an alternative to them, depending on the application.

In modern mathematical language, quaternions form a four-dimensional associative normed division algebra over the real numbers, and therefore also a domain. In fact, the quaternions were the first no commutative division algebra to be discovered. The algebra of quaternions is often denoted by **H** (for *Hamilton*), or in blackboard bold by $\mathbb{H}$ (Unicode U+210D, ℍ). It can also be given by the Clifford algebra classifications $C\ell_{0,2}(\mathbf{R}) \cong C\ell^{0}_{3,0}(\mathbf{R})$. The algebra **H** holds a special place in analysis since, according to the Frobenius theorem; it is one of only two finite-dimensional division rings containing the real numbers as a proper subring, the other being the complex numbers. These rings are also Euclidean Hurwitz algebras, of which quaternions are the largest associative algebra.

The fundamental formula of quaternion algebra,

$$i^2 = j^2 = k^2 = i\,j\,k = -1,$$

By analogy with the complex numbers being representable as a sum of real and imaginary parts, $a \cdot 1 + b\,i$, a quaternion can also be written as a linear combination

$$H = a \cdot 1 + b\,i + c\,j + d\,k. \qquad (2)$$

The quaternion $a + b\,i + c\,j + d\,k$ is implemented as Quaternion [*a*, *b*, *c*, *d*] in the *Mathematica* package Quaternions` where however a,b, c, and $d$must be *explicit real numbers.*

5.2.2.7.2 Direction

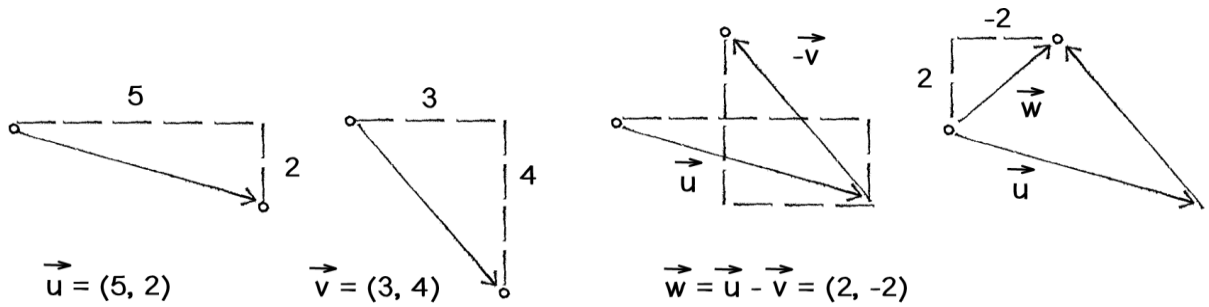
Calculate the direction vector is necessary to determine a bone direction of avatar's skeleton, which begins and ends with joint. So calculating a bone direction is by subtracting the parent joint position from the current joint position as the equation below:

```
int jParent = (int)sensorData.sensorInterface.GetParentJoint(bodyFrame.bodyData[bodyID].joint[j].jointType);
bodyFrame.bodyData[bodyID].joint[j].direction = bodyFrame.bodyData[bodyID].joint[j].position - bodyFrame.bodyData[bodyID].joint[jParent].position;
```

In math:

P(x0, y0, z0), Q(x1, y1, z1)

PQ=Q-P= (x1-x0, y1-y0, z1-z0)



5.3 Software Requirements

There are three modes for training or workout. The first is without Kinect, the other one is with Kinect connected and the last one is the stereoscopy view.

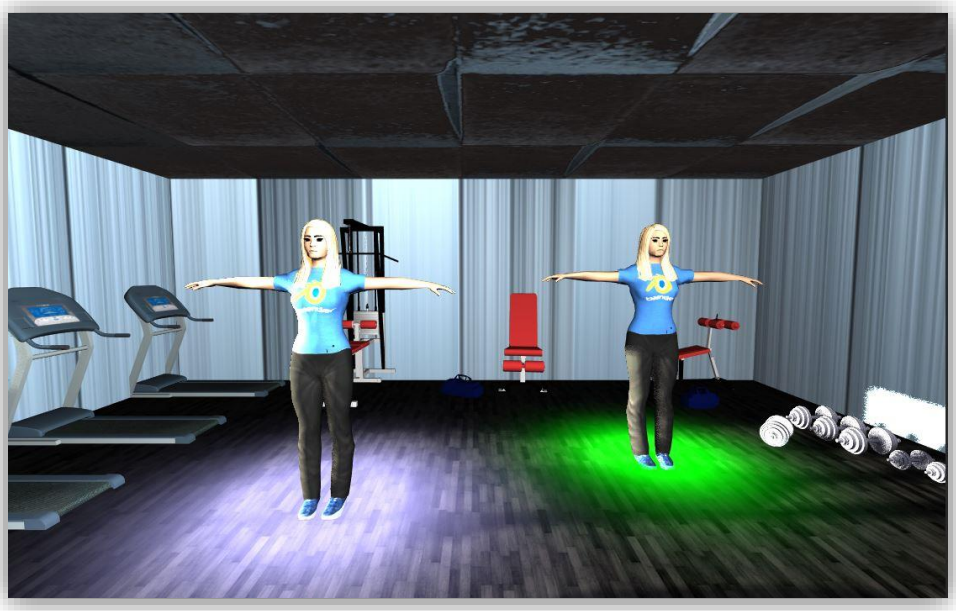
5.3.1 Provide playing without Kinect

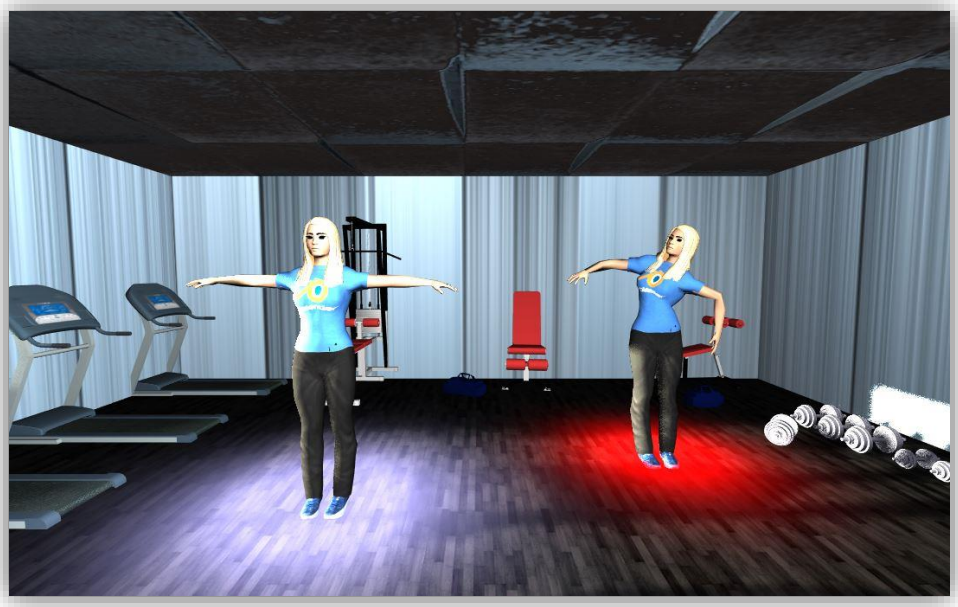
Provide playing with the coach without a Kinect is a good feature in Virtual Coach for people who do not have Kinect. It lets the player view the coach movements and workout like on video or TV!

Figure 48 Mode 1 PrintScreen

5.3.2 Provide playing with Kinect

As mentioned above, this is the second mode the player can choose and the default mode. People who have Kinect can enjoy workout by following the coach moves. In this mode, the movements are tracked to the coach ones, so the player could do the exercises correctly. According to correctness percentage and the threshold (>70% is a correct move), a green spot under the player avatar while be appear if the move is correct. Otherwise, a red spot if the move is incorrect.





5.3.3 3D Stereoscopy Mode:

What is an anaglyph?

Anaglyphic stereograms (anaglyphs) are stereo pairs of images in which each image is shown using a different color. The two images are overlapped and then viewed using red/green, red/blue, or red/cyan glasses (depending on the colors used). This means that the color channel is used for the stereo separation and therefore the perception of anaglyphs is monochrome (black and white), although color anaglyphs can be made.

The word anaglyph comes from the Greek anagluphos, meaning, "wrought in low relief"; this comes from the word anagluphein, which means, "to carve in relief" (ana = up + gluphein = to carve).

Who invented the anaglyph?

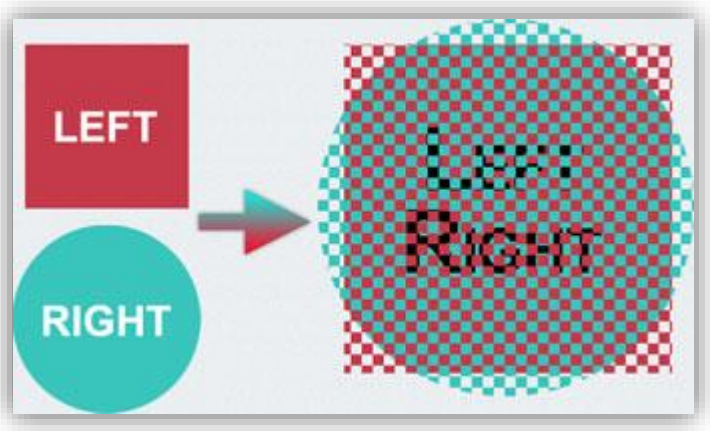
The discovery of anaglyphic 3D came in the 1850s as the result of experiments by the Frenchmen Joseph D'Almeida and Louis Du Hauron. In 1858, D'Almeida projected 3D magic lantern slide shows in which color separation took place using red and blue filters, and the audience wore red and blue goggles. Louis Du Hauron created the first printed anaglyphs using early color printing and photography techniques. William Friese-Green created the first 3D anaglyphic motion pictures in 1889, using a camera with two lenses, which were first shown to the public in 1893. Anaglyphic films called "plasticons" or "plastigrams" enjoyed great popularity in the 1920s. These used a single film with the green image emulsion on one side of the film and the red image emulsion on the other. In 1922, an interactive plasticon, entitled "Movies of the Future," opened at the Rivoli Theater in New York. The film provided the viewer with an optional ending. The happy conclusion was viewed using the green filter, while the tragic ending could be seen using the red filter. Many science fiction films of the 1950s were in anaglyphic 3D and have been shown on television (although today they are often shown in theaters using polarized 3D).

Anaglyphic images have been used in comic books, newspapers, and magazine ads. In 1953, 3D comic books were invented and distributed with red/green “space goggles.” Three- and four-color anaglyphic images are in use today in comic books and by some advertisers, but most often images are two-color.

How do the red/green glasses work?

When you look through the red lens, only red light is allowed through. The eye that is covered with the red lens will see the green image. Similarly, the green lens only allows green light through, so the eye that is covered with the green lens will see the red image. In an anaglyph, when a given color filter stops the other colors, it is called subtractive filtration. Because the red and green images are slightly offset, each eye sees a slightly different view of the picture. This disparity simulates the distance between our two eyes, which provides two views of the same scene, therefore providing us with the perception of depth, or binocular stereopsis.

Figure 49 3D Lens



How can Anaglyph Photos be made?

There are MANY ways to make anaglyph photos. In simplest terms: The basic idea is to use the principle of complementary colors to encode your stereo information. A stereo photograph requires a left eye view and a right eye view; to view a photo in stereo the left eye must see ONLY the left eye view and the right eye must see only the RIGHT eye view. Any combination of two complementary colors can achieve this.

In practice... You can produce an anaglyph stereo photo in camera one of the following methods:

By double exposing the film through complementary colored filters: click off one exposure through one filter for the first eye view, shift the camera over for the other point of view without advancing the film, change filters and expose for the other eye view. You must, of course, use color film.

You can take left and right stereo negatives and print them one at a time through complementary filters onto color positive paper to get an anaglyph photo in the darkroom. This works best with black and white negatives, but color negatives can be used, in addition, it is just a little trickier getting a pleasing color balance in the middle of the spectrum.

You could project left and right stereo positive slides (black and white is easiest, but color is not impossible) with complementary colored filters over the projection lens. The results tend to be muddy, but it does work with simple subjects.

Use a Polaroid camera with double exposure capability; use color film. Take the left eye exposure through one filter, re-cock the shutter, change filters, and shift the camera and exposure the other eye view. Presto! An instant stereo anaglyph print.

Using a mirror box and beam splitter rig, you can make anaglyph movies or stills by taping complementary colored gels over the left and right mirror openings. A company in England made such a device for amateurs, who don't want to set up dual projection or who don't care to split the already tiny 8mm frame for left and right eye views.

Well, this could go on and on. Just apply the basic principle to whatever situation you are in and to what you have to work with.

5.4 Implementation and Evolution

5.4.1 Move the coach

In every mode, there is the coach avatar, which takes the animation from the XML files. To move the avatar:

5.4.1.1 Reading from XML:

In VCWR class, I used the function to read from XML file. The function takes the file path as a string and returns a dictionary, the key is the frame number and the value is VCM1, which contains the joints and its position and orientations.

```
public static Dictionary<int, VCM1> ReadBodyJointsfromXML(string file)
{
    Dictionary<int, VCM1> toBeReturned = new Dictionary<int, VCM1>();
    XmlDocument doc = new XmlDocument();
    doc.Load(file);
    foreach (XmlNode node in doc.DocumentElement.ChildNodes)
    {
        VCM1 Tbody = new VCM1();
        int frameindex = Convert.ToInt16(node.Attributes["Number"].Value);
        foreach (XmlNode innernode in node.ChildNodes)
        {
            string JointTypeString = innernode.Name;
            Quaternion joint_Or;

            joint_Or.w = float.Parse(innernode.Attributes["RW"].Value);
            joint_Or.x = float.Parse(innernode.Attributes["RX"].Value);
            joint_Or.y = float.Parse(innernode.Attributes["RY"].Value);
            joint_Or.z = float.Parse(innernode.Attributes["RZ"].Value);
            Vector3 joint_O;
            joint_O.x = float.Parse(innernode.Attributes["X"].Value);
            joint_O.y = float.Parse(innernode.Attributes["Y"].Value);
            joint_O.z = float.Parse(innernode.Attributes["Z"].Value);
            determine the jointtype

        }
        toBeReturned.Add(frameindex, Tbody);

    }
    return toBeReturned;
}
```

5.4.1.2 Assigning Values:

Then In KinectManagerXML class assign the values from XML to the Kinect, for processing them later.

```
bodyFrame.bodyData[bodyID].position = fjle[frameCounter].Coach_joints[JointType.SpineBase];
bodyFrame.bodyData[bodyID].joint[14].position = fjle[frameCounter].Coach_joints[JointType.AnkleLeft];
bodyFrame.bodyData[bodyID].joint[18].position = fjle[frameCounter].Coach_joints[JointType.AnkleRight];
bodyFrame.bodyData[bodyID].joint[5].position = fjle[frameCounter].Coach_joints[JointType.ElbowLeft];
bodyFrame.bodyData[bodyID].joint[9].position = fjle[frameCounter].Coach_joints[JointType.ElbowRight];
bodyFrame.bodyData[bodyID].joint[15].position = fjle[frameCounter].Coach_joints[JointType.FootLeft];
bodyFrame.bodyData[bodyID].joint[19].position = fjle[frameCounter].Coach_joints[JointType.FootRight];
bodyFrame.bodyData[bodyID].joint[7].position = fjle[frameCounter].Coach_joints[JointType.HandLeft];
bodyFrame.bodyData[bodyID].joint[11].position = fjle[frameCounter].Coach_joints[JointType.HandRight];
bodyFrame.bodyData[bodyID].joint[21].position = fjle[frameCounter].Coach_joints[JointType.HandTipLeft];
bodyFrame.bodyData[bodyID].joint[23].position = fjle[frameCounter].Coach_joints[JointType.HandTipRight];
bodyFrame.bodyData[bodyID].joint[3].position = fjle[frameCounter].Coach_joints[JointType.Head];
bodyFrame.bodyData[bodyID].joint[12].position = fjle[frameCounter].Coach_joints[JointType.HipLeft];
bodyFrame.bodyData[bodyID].joint[16].position = fjle[frameCounter].Coach_joints[JointType.HipRight];
bodyFrame.bodyData[bodyID].joint[13].position = fjle[frameCounter].Coach_joints[JointType.KneeLeft];
bodyFrame.bodyData[bodyID].joint[17].position = fjle[frameCounter].Coach_joints[JointType.KneeRight];
bodyFrame.bodyData[bodyID].joint[2].position = fjle[frameCounter].Coach_joints[JointType.Neck];
bodyFrame.bodyData[bodyID].joint[4].position = fjle[frameCounter].Coach_joints[JointType.ShoulderLeft];
bodyFrame.bodyData[bodyID].joint[8].position = fjle[frameCounter].Coach_joints[JointType.ShoulderRight];
bodyFrame.bodyData[bodyID].joint[0].position = fjle[frameCounter].Coach_joints[JointType.SpineBase];
bodyFrame.bodyData[bodyID].joint[1].position = fjle[frameCounter].Coach_joints[JointType.SpineMid];
bodyFrame.bodyData[bodyID].joint[20].position = fjle[frameCounter].Coach_joints[JointType.SpineShoulder];
bodyFrame.bodyData[bodyID].joint[22].position = fjle[frameCounter].Coach_joints[JointType.ThumbLeft];
bodyFrame.bodyData[bodyID].joint[24].position = fjle[frameCounter].Coach_joints[JointType.ThumbRight];
bodyFrame.bodyData[bodyID].joint[6].position = fjle[frameCounter].Coach_joints[JointType.WristLeft];
bodyFrame.bodyData[bodyID].joint[10].position = fjle[frameCounter].Coach_joints[JointType.WristRight];
```

```
bodyFrame.bodyData[bodyID].joint[14].orientation = fjle[frameCounter].Coach_jointOrientations[JointType.AnkleLeft];
bodyFrame.bodyData[bodyID].joint[18].orientation = fjle[frameCounter].Coach_jointOrientations[JointType.AnkleRight];
bodyFrame.bodyData[bodyID].joint[5].orientation = fjle[frameCounter].Coach_jointOrientations[JointType.ElbowLeft];
bodyFrame.bodyData[bodyID].joint[9].orientation = fjle[frameCounter].Coach_jointOrientations[JointType.ElbowRight];
bodyFrame.bodyData[bodyID].joint[15].orientation = fjle[frameCounter].Coach_jointOrientations[JointType.FootLeft];
bodyFrame.bodyData[bodyID].joint[19].orientation = fjle[frameCounter].Coach_jointOrientations[JointType.FootRight];
bodyFrame.bodyData[bodyID].joint[7].orientation = fjle[frameCounter].Coach_jointOrientations[JointType.HandLeft];
bodyFrame.bodyData[bodyID].joint[11].orientation = fjle[frameCounter].Coach_jointOrientations[JointType.HandRight];
bodyFrame.bodyData[bodyID].joint[21].orientation = fjle[frameCounter].Coach_jointOrientations[JointType.HandTipLeft];
bodyFrame.bodyData[bodyID].joint[23].orientation = fjle[frameCounter].Coach_jointOrientations[JointType.HandTipRight];
bodyFrame.bodyData[bodyID].joint[3].orientation = fjle[frameCounter].Coach_jointOrientations[JointType.Head];
bodyFrame.bodyData[bodyID].joint[12].orientation = fjle[frameCounter].Coach_jointOrientations[JointType.HipLeft];
bodyFrame.bodyData[bodyID].joint[16].orientation = fjle[frameCounter].Coach_jointOrientations[JointType.HipRight];
bodyFrame.bodyData[bodyID].joint[13].orientation = fjle[frameCounter].Coach_jointOrientations[JointType.KneeLeft];
bodyFrame.bodyData[bodyID].joint[17].orientation = fjle[frameCounter].Coach_jointOrientations[JointType.KneeRight];
bodyFrame.bodyData[bodyID].joint[2].orientation = fjle[frameCounter].Coach_jointOrientations[JointType.Neck];
bodyFrame.bodyData[bodyID].joint[4].orientation = fjle[frameCounter].Coach_jointOrientations[JointType.ShoulderLeft];
bodyFrame.bodyData[bodyID].joint[8].orientation = fjle[frameCounter].Coach_jointOrientations[JointType.ShoulderRight];
bodyFrame.bodyData[bodyID].joint[0].orientation = fjle[frameCounter].Coach_jointOrientations[JointType.SpineBase];
bodyFrame.bodyData[bodyID].joint[1].orientation = fjle[frameCounter].Coach_jointOrientations[JointType.SpineMid];
bodyFrame.bodyData[bodyID].joint[20].orientation = fjle[frameCounter].Coach_jointOrientations[JointType.SpineShoulder];
bodyFrame.bodyData[bodyID].joint[22].orientation = fjle[frameCounter].Coach_jointOrientations[JointType.ThumbLeft];
bodyFrame.bodyData[bodyID].joint[24].orientation = fjle[frameCounter].Coach_jointOrientations[JointType.ThumbRight];
bodyFrame.bodyData[bodyID].joint[6].orientation = fjle[frameCounter].Coach_jointOrientations[JointType.WristLeft];
bodyFrame.bodyData[bodyID].joint[10].orientation = fjle[frameCounter].Coach_jointOrientations[JointType.WristRight];
```

The processed values is ready to update the avatar movement. This is done by calling the “Update Avatar” function, which is define in AvatarControllerXML class:

```
0 references
public void UpdateAvatar(Int64 UserID)
{
    if (!transform.gameObject.activeInHierarchy)
        return;

    // Get the KinectManager instance
    if (kinectManager == null)
    {
        kinectManager = KinectManagerXML.Instance;
    }

    // move the avatar to its Kinect position
    MoveAvatar(UserID);

    for (var boneIndex = 0; boneIndex < bones.Length; boneIndex++)
    {
        if (!bones[boneIndex])
            continue;

        if (boneIndex2JointMap.ContainsKey(boneIndex))
        {
            KinectInterop.JointType joint = !mirroredMovement ? boneIndex2JointMap[boneIndex] : boneIndex2MirrorJointMap[boneIndex];
            TransformBone(UserID, joint, boneIndex, !mirroredMovement);
        }
        else if (specIndex2JointMap.ContainsKey(boneIndex))
        {
            // special bones (clavicles)
            List<KinectInterop.JointType> alJoints = !mirroredMovement ? specIndex2JointMap[boneIndex] : specIndex2MirrorJointMap[boneIndex];

            if (alJoints.Count >= 2)
            {
                //Debug.Log(alJoints[0].ToString());
                Vector3 baseDir = alJoints[0].ToString().EndsWith("Left") ? Vector3.left : Vector3.right;
                TransformSpecialBone(UserID, alJoints[0], alJoints[1], boneIndex, baseDir, !mirroredMovement);
            }
        }
    }
}
```

The “TransformBone” function contains:

```
// Get Kinect joint orientation
Quaternion jointRotation = kinectManager.GetJointOrientation(userId, iJoint, flip);
if (jointRotation == Quaternion.identity)
    return;

// Smoothly transition to the new rotation
Quaternion newRotation = Kinect2AvatarRot(jointRotation, boneIndex);

if (smoothFactor != 0f)
    boneTransform.rotation = Quaternion.Slerp(boneTransform.rotation, newRotation, smoothFactor * Time.deltaTime);
else
    boneTransform.rotation = newRotation;
```

5.4.1.3 Move the Player

In the default mode playing with Kinect, the avatar player moves according to the values of joints that came from Kinect. In “KinectManager”, class which takes the values from Kinect and processes them. Then it calls the “UpdateAvatar” function that is defined in “AvatarController” class; it is the same method in “AvtarControllerXML”.

5.4.1.4 Tracking the movement:

Since the default mode allows user to play with the coach in real time, it tracks the movements too. To do that I used “percentBetweenJoints” function which takes every three hierarchical joints from the coach and the same from the player, then compares them and gives a percentage of matching.

The hierarchy of joints:

Figure 50 The Joints Hierarchy

The compare code:

```
double percent = 0;
percent += VCWR.percentBetweenJoints(bodyFrame.bodyData[bodyID].joint[3].position, bodyFrame.bodyData[bodyID].joint[2].position, bodyFrame.bodyData[bodyID].joint[20].position, fjle
  [frameCounter].Coach_joints[JointType.Head], fjle[frameCounter].Coach_joints[JointType.Neck], fjle[frameCounter].Coach_joints[JointType.SpineShoulder]);

percent += VCWR.percentBetweenJoints(bodyFrame.bodyData[bodyID].joint[2].position, bodyFrame.bodyData[bodyID].joint[20].position, bodyFrame.bodyData[bodyID].joint[1].position, fjle
  [frameCounter].Coach_joints[JointType.Neck], fjle[frameCounter].Coach_joints[JointType.SpineShoulder], fjle[frameCounter].Coach_joints[JointType.SpineMid]);

percent += VCWR.percentBetweenJoints(bodyFrame.bodyData[bodyID].joint[20].position, bodyFrame.bodyData[bodyID].joint[1].position, bodyFrame.bodyData[bodyID].joint[0].position, fjle
  [frameCounter].Coach_joints[JointType.SpineShoulder], fjle[frameCounter].Coach_joints[JointType.SpineMid], fjle[frameCounter].Coach_joints[JointType.SpineBase]);

percent += VCWR.percentBetweenJoints(bodyFrame.bodyData[bodyID].joint[20].position, bodyFrame.bodyData[bodyID].joint[4].position, bodyFrame.bodyData[bodyID].joint[5].position, fjle
  [frameCounter].Coach_joints[JointType.SpineShoulder], fjle[frameCounter].Coach_joints[JointType.ShoulderLeft], fjle[frameCounter].Coach_joints[JointType.ElbowLeft]);

percent += VCWR.percentBetweenJoints(bodyFrame.bodyData[bodyID].joint[4].position, bodyFrame.bodyData[bodyID].joint[5].position, bodyFrame.bodyData[bodyID].joint[6].position, fjle
  [frameCounter].Coach_joints[JointType.ShoulderLeft], fjle[frameCounter].Coach_joints[JointType.ElbowLeft], fjle[frameCounter].Coach_joints[JointType.WristLeft]);

percent += VCWR.percentBetweenJoints(bodyFrame.bodyData[bodyID].joint[5].position, bodyFrame.bodyData[bodyID].joint[6].position, bodyFrame.bodyData[bodyID].joint[7].position, fjle
  [frameCounter].Coach_joints[JointType.ElbowLeft], fjle[frameCounter].Coach_joints[JointType.WristLeft], fjle[frameCounter].Coach_joints[JointType.HandLeft]);

percent += VCWR.percentBetweenJoints(bodyFrame.bodyData[bodyID].joint[20].position, bodyFrame.bodyData[bodyID].joint[8].position, bodyFrame.bodyData[bodyID].joint[9].position, fjle
  [frameCounter].Coach_joints[JointType.SpineShoulder], fjle[frameCounter].Coach_joints[JointType.ShoulderRight], fjle[frameCounter].Coach_joints[JointType.ElbowRight]);

percent += VCWR.percentBetweenJoints(bodyFrame.bodyData[bodyID].joint[8].position, bodyFrame.bodyData[bodyID].joint[9].position, bodyFrame.bodyData[bodyID].joint[10].position, fjle
  [frameCounter].Coach_joints[JointType.ShoulderRight], fjle[frameCounter].Coach_joints[JointType.ElbowRight], fjle[frameCounter].Coach_joints[JointType.WristRight]);
```

And so on…

Calculating the matching percentage for every frame is not efficient (the player cannot do the same move with the coach), but calculating it for every 33 frames (equals every second) is good enough. In addition, what if the player did not play the whole exercise correctly?

Another percentage is calculated for the whole exercise, if the percentage is under 60% then the player must replay the exercise, else he/she passes.

5.4.2 Convert from Right-Handed to Left-Handed Coordinate System:

As we discussed in chapter 2, the Kinect is right-handed coordinate system and the Unity is left-handed coordinate system. It's easy to convert the position which is a Vector3 (x,y,z) by navigate the z, but it's harder to deal with Quaternion (x,y,z,w). For converting Quaternion from Right-Handed to Left-Handed Coordinate System, it must convert the Quaternion to a rotation matrix 4x4:

$1 - 2*Y^2 - 2*Z^2$	$2*X*Y - 2*Z*W$	$2*X*Z + 2*Y*W$	0
$2*X*Y + 2*Z*W$	$1 - 2*X^2 - 2*Z^2$	$2*Y*Z - 2*X*W$	0
$2*X*Z - 2*Y*W$	$2*Y*Z + 2*X*W$	$1 - 2*X^2 - 2*Y^2$	0
0	0	0	1

Note: In Unity, the matrix is row-major and in Kinect is column-major:

This array

$$\begin{bmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \end{bmatrix}$$

Would be stored as follows in the two orders:

Column-Major Order
e.g. Fortran

Address	Value
0	11
1	21
2	12
3	22
4	13
5	23

Row-Major Order
e.g. C

Address	Value
0	11
1	12
2	13
3	21
4	22
5	23

Then converting from right-handed matrix to left-handed matrix:

-(1 - 2*Y² - 2*Z²)	2*X*Y - 2*Z*W	2*X*Z + 2*Y*W	0
-(2*X*Y + 2*Z*W)	1 - 2*X² - 2*Z²	2*Y*Z – 2*X*W	0
2*X*Z - 2*Y*W	-(2*Y*Z + 2*X*W)	-(1 - 2*X² - 2*Y²)	0
0	0	0	1

Finally, convert from rotation matrix to quaternion:

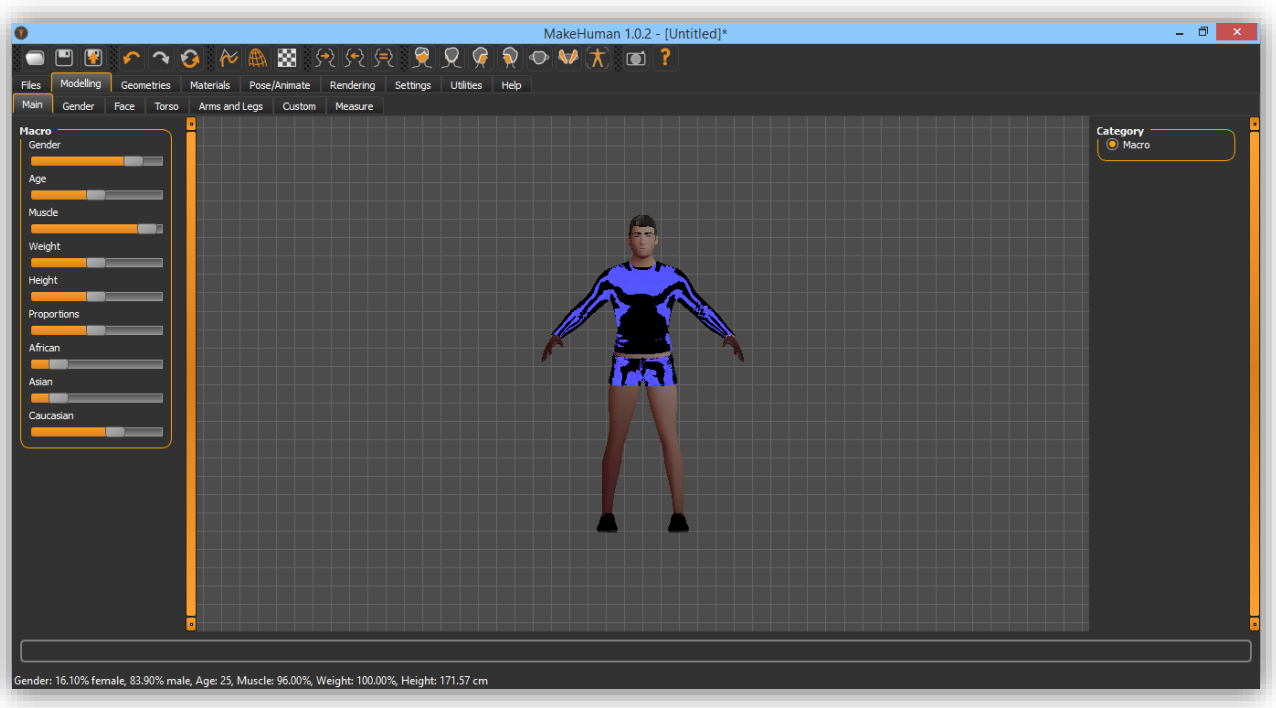
$\frac{1}{2} \times \sqrt{(1 + m[0,0] + m[1,1] + m[2,2]}$	$\frac{1}{2} \times \sqrt{(1 + m[0,0] - m[1,1] - m[2,2]}$	$\frac{1}{2} \times \sqrt{(1 + m[1,1] - m[0,0] - m[2,2]}$	$\frac{1}{2} \times \sqrt{(1 + m[2,2] - m[0,0] - m[1,1]}$

5.5 Design

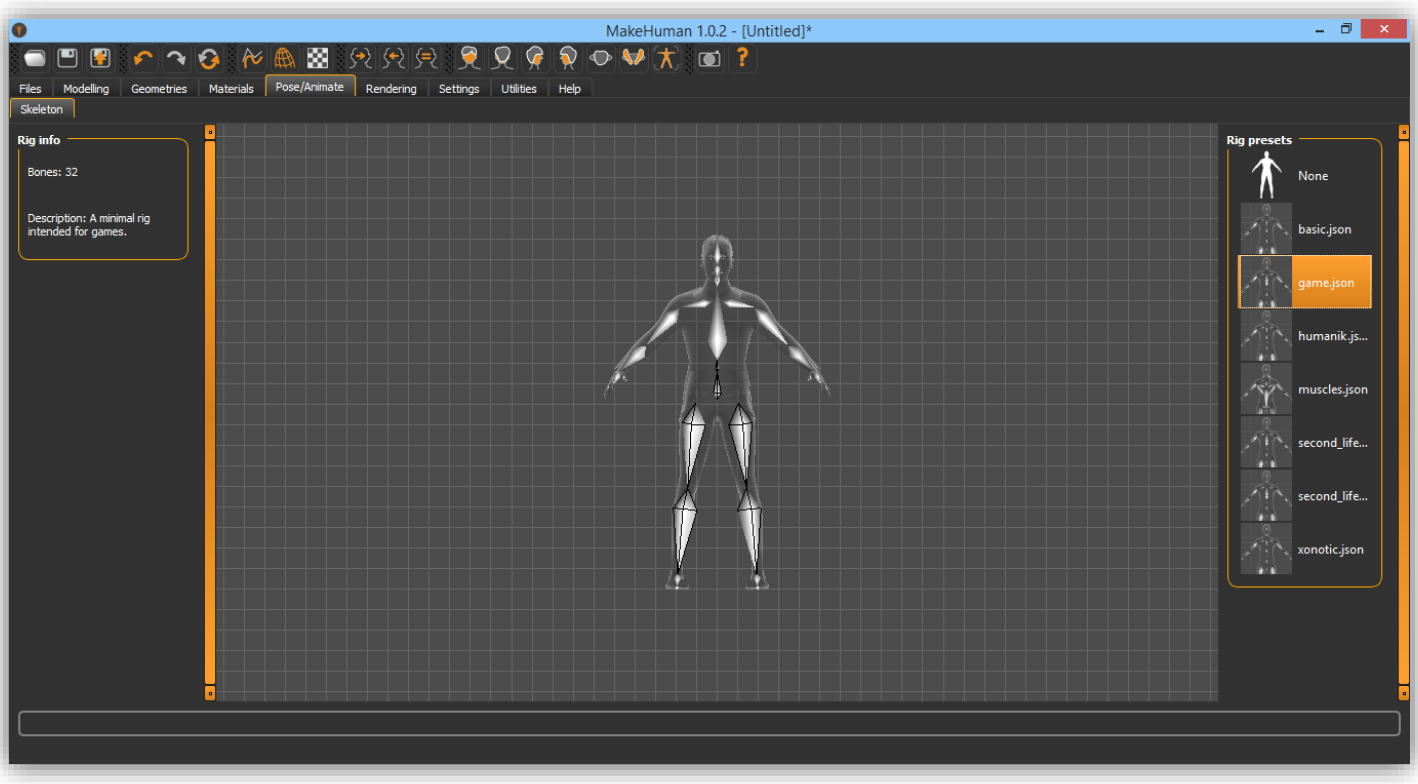
5.5.1 Modeling Avatar:

To Model custom avatars, I used MakeHuman software

Figure 51 MakeHuman Software Interface



MakeHuman allows users to choose the best rig, so I choose the game rig as it is the closest one to the Kinect one:



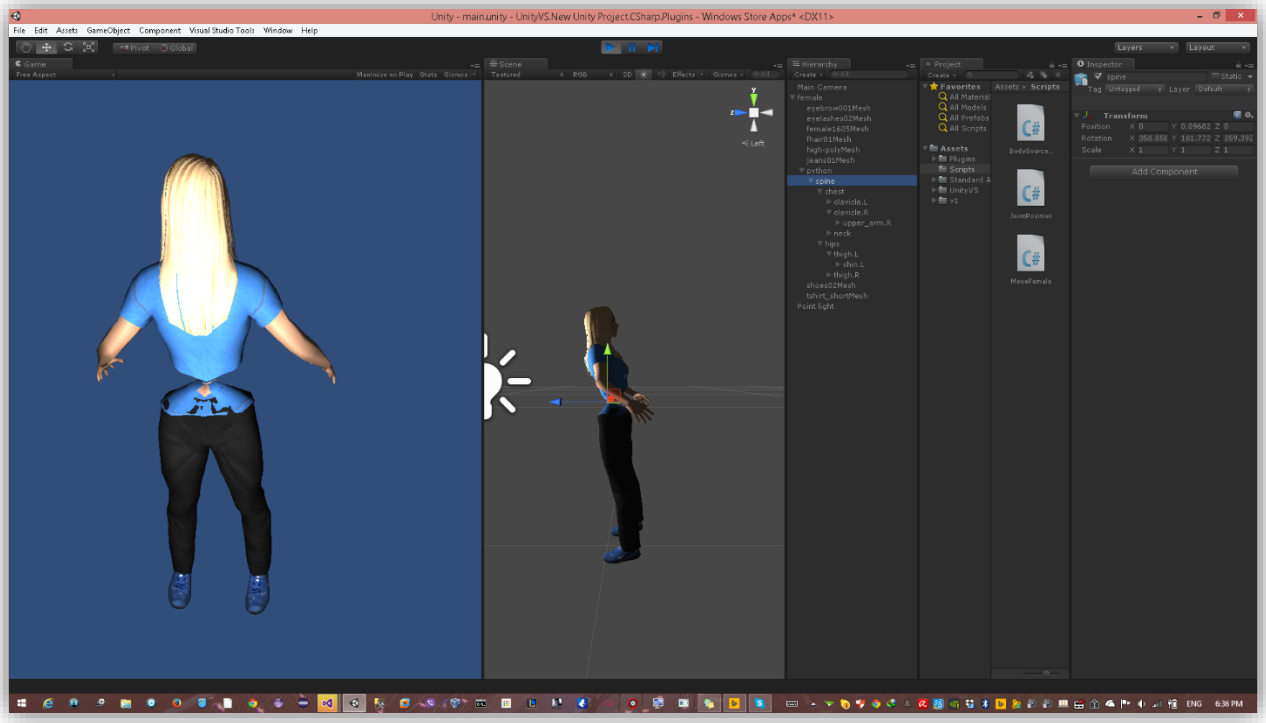
5.5.2 The Problem:

When I was trying to move an avatar using Unity3D according to the data coming from Kinect V2, I faced a problem when applying multi joints orientations rotation.

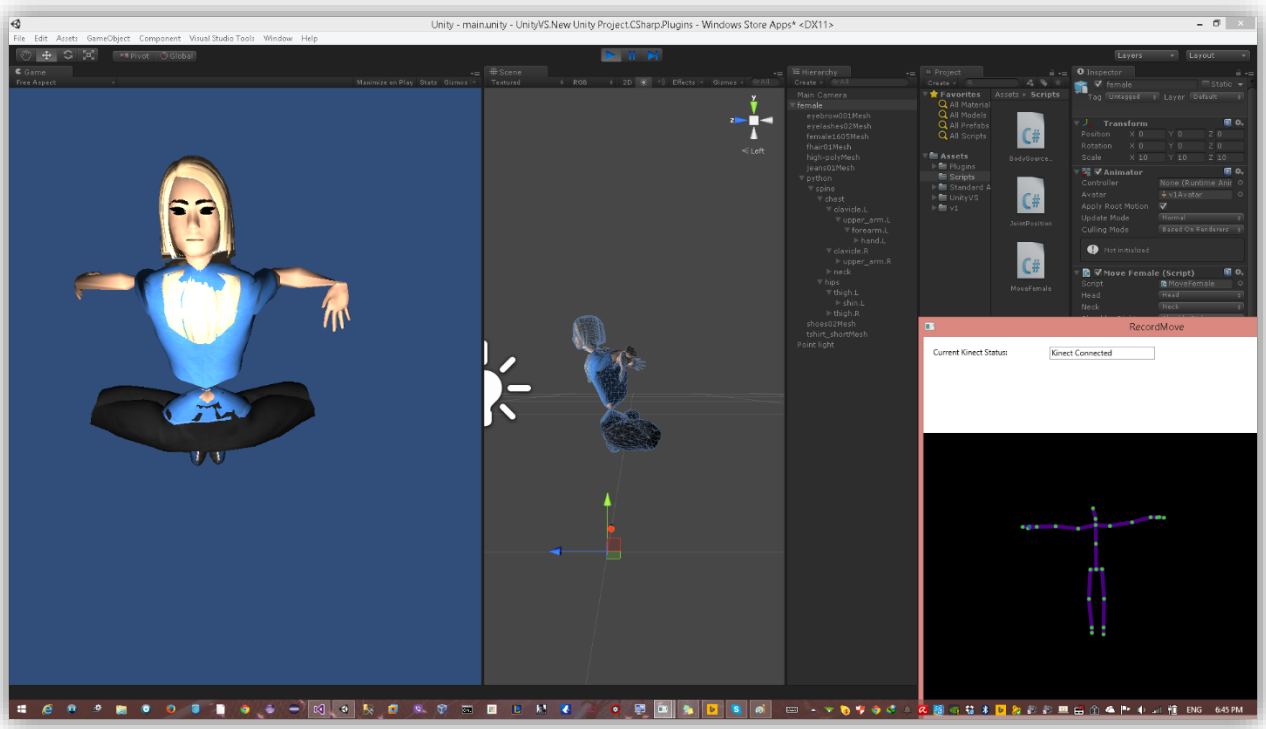
```
float ox, oy, oz, ow;

ox = body.JointOrientations[_SpineMid].Orientation.X;
oy = body.JointOrientations[_SpineMid].Orientation.Y;
oz = body.JointOrientations[_SpineMid].Orientation.Z;
ow = body.JointOrientations[_SpineMid].Orientation.W;
//SpineMid.transform.localRotation = new Quaternion(ox, oy, oz, ow);
SpineMid.transform.localRotation = Quaternion.Slerp(SpineMid.transform.localRotation, new Quaternion(ox, oy, oz,
  ow), Time.time * 0.1f);

ox = body.JointOrientations[_SpineBase].Orientation.X;
oy = body.JointOrientations[_SpineBase].Orientation.Y;
oz = body.JointOrientations[_SpineBase].Orientation.Z;
ow = body.JointOrientations[_SpineBase].Orientation.W;
//SpineBase.transform.localRotation = new Quaternion(ox, oy, oz, ow);
SpineBase.transform.localRotation = Quaternion.Slerp(SpineBase.transform.localRotation, new Quaternion(ox, oy,
  oz, ow), Time.time * 0.1f);
```



The funniest part is when i tried to apply the previous Slerp transformation to every joint in in the avatar.



Then I released that I have to fix the local orientations for every joints to match those are coming from Kinect sensor using 3D Studio Max:

Figure 52 Kinect Orientations

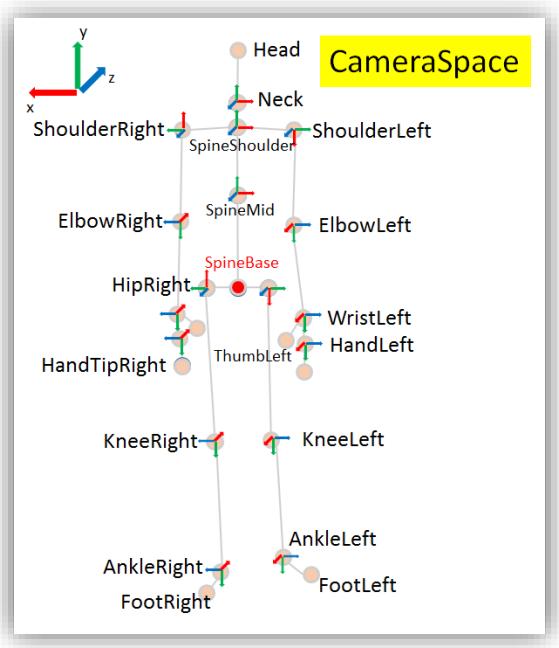
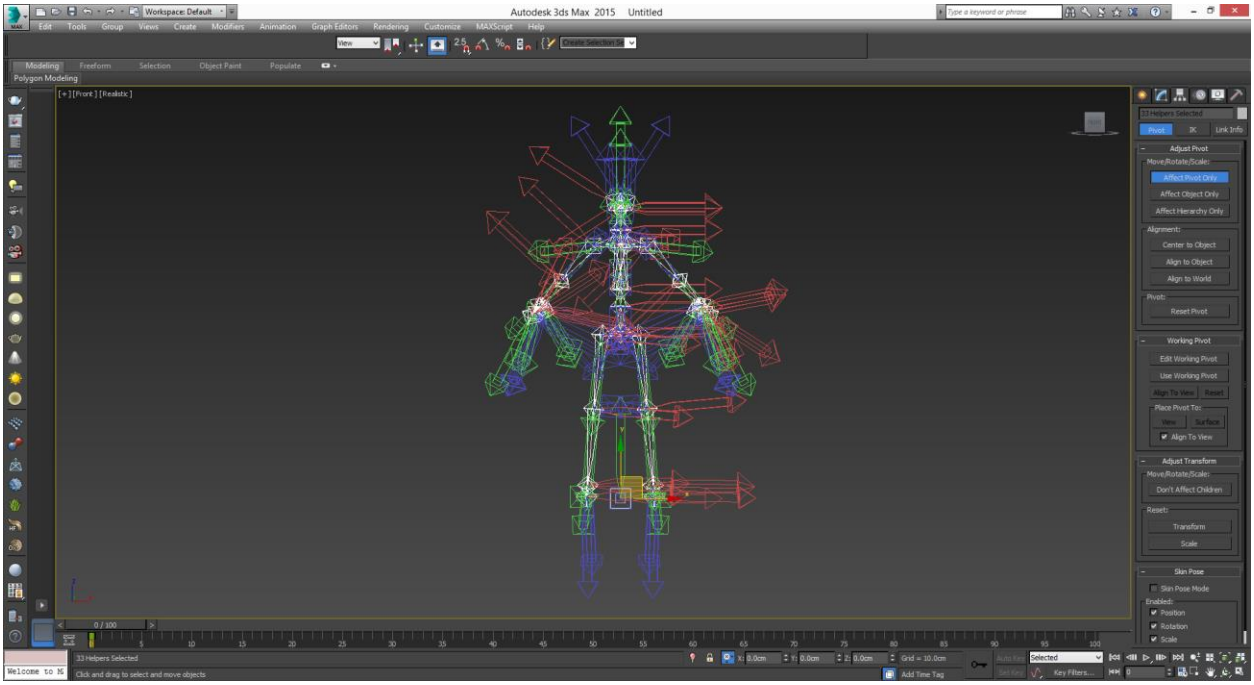
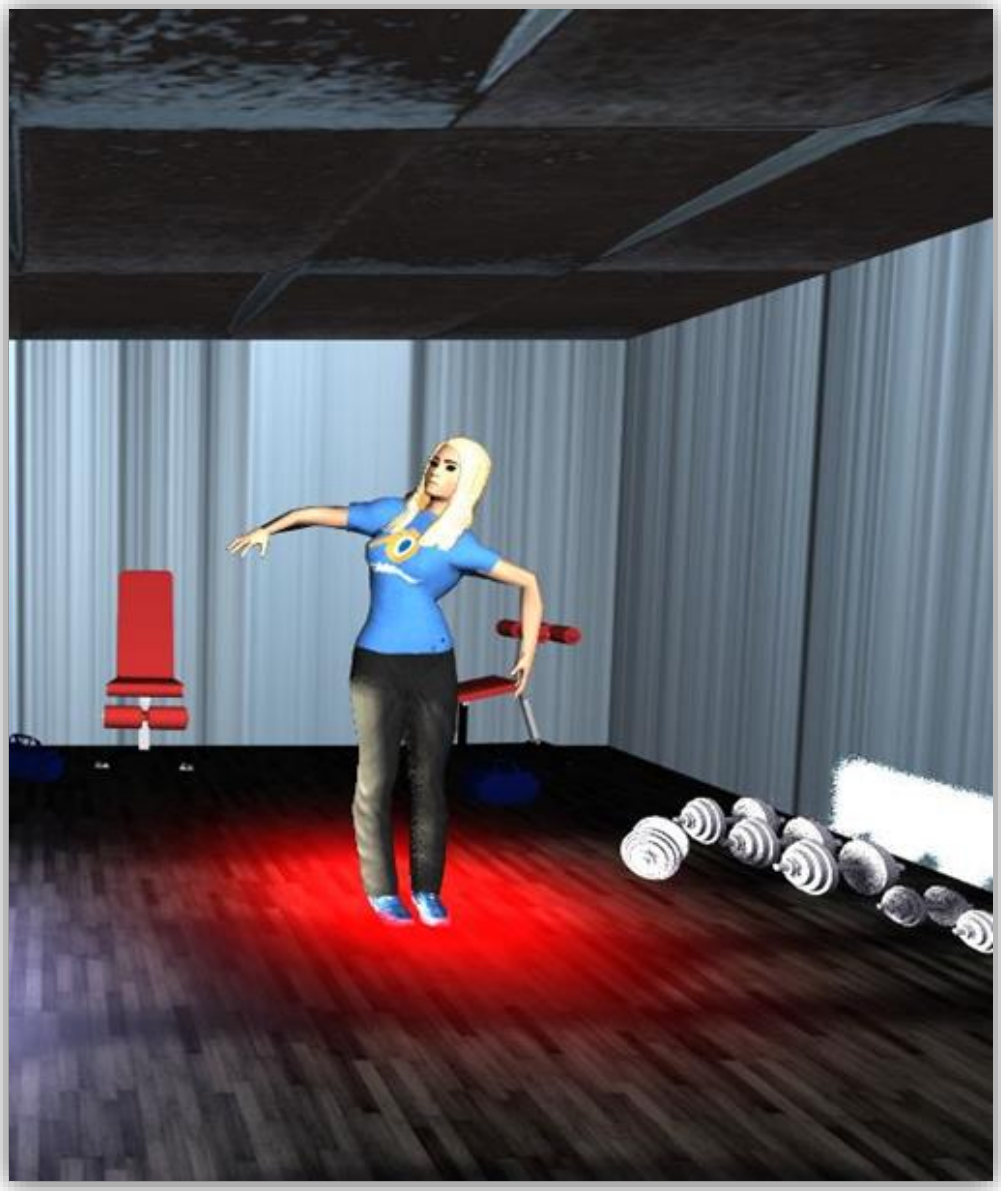


Figure 53 Fixing wrong Joints wth 3D Studio Max



The final result:



5.6 Speech

One of our problems was the ability to interact with the system while playing in front of the sensor as its impossible use mouse or keyboard while being faraway (1.5 meters or more). So in order to override the problem, we introduced a new way to interact, which is voice input.

To get started I implemented a controller called SpeechManager, which tries to recognize the sounds around the sensor in real time (for each update scene trigger). When the SpeechManager detects a sound contains a human voice it returns that voice segment to the grammar to be processed.

The main commands I have added to the system are:

- "EXIT": which terminates the unity3D scene and returns the player to his home screen in the VirtualCoach App.
- "PAUSE": which helps the player to suddenly pause the coach movement in order to be continued later.
- "CONTINUE": this command will only trigger when the system is in the Paused mode, which moves the coach again to continue the exercise.
- "MODE ONE": sometimes players want to jump from the tracking enabled mode to the static coach only mode.
- "MODE TWO": it navigates the players from the static coach only mode to tracking enabled mode in order to check the accuracy of their movement.

5.6.1 The Grammar

When testing speech recognition on a number of players I realized that it is nearly impossible to get all system users to say the particular words.

E.g., players playing in the mode two and want to disable the tracking mode they tries to say "Disable coach", "Coach off".

To fix that I have created a grammar file, which contains the most, used words by players.

Conclusion

This report introduced environmentally-friendly designs for fitness purpose. We presented information about the fitness programs, exercises, and accurate movements and provided a modern website for more accessibility. These factors would help the players who uses VirtualCoach to be more motivated than normal players.

Future Work:

- Provide muscles force
- Provide muscles volume
- Provide heartbeat rate

Table of Figures