



# **Heart Health**

## **Home-based rehabilitation system using motion capture with Serious Games**

**Andrew Daly**

**C11710699**

**DT211**

**BSc in Computing**

*School of Computing*

*Dublin Institute of Technology*

*Kevin St. Dublin 8, Ireland.*

**Supervisors:**  
**Bryan Duggan**



## **Declaration**

This thesis is a presentation of my original research work. Wherever contributions of others are involved, every effort is made to indicate this clearly, with due reference to the literature, and acknowledgements of collaborative research and discussions.

The work was completed in accordance with the Dublin Institute of Technology regulations and under the guidance of course lecturer Bryan Duggan. This thesis has not been submitted in whole or in part for an award in any other institution.

The Institute has permission to keep, to lend or to copy this thesis in whole or in part, on condition that any such use of the material of the thesis is duly acknowledged.

Signed

---

Andrew Daly

Date

---

## **Abstract**

The research presented in this abstract concern the utilization of Serious Games in terms of home-based patient rehabilitation. The main goal here is to attempt to provide motivation and enjoyment during the performance of exercises as well as monitor their condition and heartrate while using the Microsoft Kinect 2 integrated into a customized open sourced game engine.

Serious games can be thought of as any game based interfaces that have been designed for any purpose other than entertainment. Serious games have been researched in the areas of: military, health, government and education. The design of these serious games can offer valuable contributions to develop effective games in the area of rehabilitation. As any other computer game, they are fundamentally intended to capture and keep a person's attention.

Patients that require rehabilitation for balance re-training, rheumatoid arthritis rehabilitation, and rehabilitation following stroke, etc. must perform consistent exercises as a crucial element in their overall physical and mental rehabilitation. However patients at home tend to either only follow their programmes for a short period of time or do not follow them at all. The problem we are aiming to solve is that patients follow their exercise programmes regularly and to view it as a fun activity. Doctors/Physiotherapists overseeing the patient's progress may also be able to keep up to date with their progress without consistent visits.

## **Acknowledgements**

I would like to give an especially massive thanks to my supervisor Bryan Duggan for all the help and guidance he has given me throughout the course of the project and over the years I have had him as a lecturer. If it wasn't for having him as a lecturer I do not think I would've been as enthusiastic and dedicated to Programming as I am now. Bryan will always be considered a friend of mine.

I would also like to thank my family especially my mother for support throughout the project as well as her funding for me to live up in Dublin.

Many thanks to my room-mate and good friend Jack Creagh for tolerating my rants when something went wrong within the project.

A final thanks to all the lads in DT211 for any assistance I needed during the project and just being friends throughout the entire time I was in DIT.

## Table of Contents

<b>1</b>	<b>INTRODUCTION.....</b>	<b>1</b>
1.1	BACKGROUND.....	2
1.1.1	<i>Cardiovascular Diseases</i> .....	2
1.1.2	<i>Doctor-Patient Interaction</i> .....	2
1.1.3	<i>Serious Games</i> .....	3
1.2	PROJECT AIMS .....	3
1.3	PROJECT OBJECTIVES.....	4
1.4	WALKTHROUGH.....	4
1.5	TIMELINE .....	5
1.6	CONCLUSION.....	5
<b>2</b>	<b>RESEARCH AND ANALYSIS.....</b>	<b>6</b>
2.1	SIMILAR SYSTEMS.....	6
2.1.1	<i>Xbox Fitness</i> .....	7
2.1.2	<i>Lloyds Online Doctor</i> .....	8
2.2	HEURISTICS.....	8
2.2.1	<i>Usability Principals</i> .....	9
2.2.2	<i>Usability Scoring</i> .....	10
2.3	RESEARCH TOPICS .....	11
2.3.1	<i>Heart Rates</i> .....	11
2.4	ADDITIONAL RESEARCH .....	12
2.5	PROPOSED SYSTEM REQUIREMENTS .....	14
2.6	CONCLUSION.....	15
<b>3</b>	<b>RESEARCH OF TECHNOLOGIES .....</b>	<b>15</b>
3.1	ARCHITECTURE.....	16
3.1.1	<i>Project Overview</i> .....	16
3.1.2	<i>Project Components</i> .....	17
3.2	AVAILABLE TECHNOLOGIES .....	17
3.2.1	<i>Game Engines</i> .....	18
3.2.2	<i>Motion Capture</i> .....	19
3.2.3	<i>Cloud Services</i> .....	21
3.2.4	<i>Server-Side Languages</i> .....	22
3.2.5	<i>Database Management System</i> .....	24
3.3	REVIEW.....	26
3.3.1	<i>Game Engines</i> .....	26
3.3.2	<i>Motion Capture</i> .....	27
3.3.3	<i>Cloud Services</i> .....	28
3.3.4	<i>Server-Side Languages</i> .....	29
3.3.5	<i>Database Management</i> .....	30
3.4	ADDITIONAL SOFTWARE.....	31
3.4.1	<i>GitHub</i> .....	31
3.4.2	<i>Visual Studio</i> .....	31
3.4.3	<i>RoboMongo</i> .....	31
3.5	CONCLUSION.....	32
<b>4</b>	<b>DESIGN .....</b>	<b>33</b>
4.1	METHODOLOGIES.....	33
4.1.1	<i>Extreme Programming Methodology</i> .....	34
4.1.2	<i>Scrum (Agile Methodology)</i> .....	35
4.1.3	<i>Iterative and Incremental Method</i> .....	36
4.1.4	<i>Methodology Analysis</i> .....	36
4.2	USE-CASE .....	37
4.3	ENTITY RELATION DIAGRAM.....	38
4.4	SYSTEM ARCHITECTURE DIAGRAM.....	39
4.5	FLOWCHART .....	40
4.6	USER INTERFACE PROTOTYPE.....	41

4.7	CONCLUSION.....	41
<b>5</b>	<b>IMPLEMENTATION.....</b>	<b>42</b>
5.1	SYSTEM OVERVIEW .....	42
5.1.1	<i>Heart Health Website</i> .....	42
5.1.2	<i>Heart Health Kinect</i> .....	43
5.2	SYSTEM COMPONENTS IMPLEMENTATION .....	44
5.2.1	<i>Database Connection Implementation</i> .....	44
5.2.2	<i>Website Implementation</i> .....	49
5.2.3	<i>Application Implementation</i> .....	55
5.3	PROBLEMS ENCOUNTERED.....	69
5.3.1	<i>Database Connection Implementation</i> .....	69
5.3.2	<i>Website Implementation</i> .....	69
5.3.3	<i>Application Implementation</i> .....	70
5.4	CONCLUSION.....	70
<b>6</b>	<b>TESTING AND EVALUATION .....</b>	<b>71</b>
6.1	TESTING METHODS .....	71
6.1.1	<i>Black Box Testing</i> .....	71
6.1.2	<i>White Box Testing</i> .....	71
6.1.3	<i>Unit Testing</i> .....	71
6.1.4	<i>Integration Testing</i> .....	72
6.1.5	<i>Verification and Validation Testing</i> .....	72
6.1.6	<i>Testing Methods Conclusion</i> .....	72
6.2	TESTING PHASE.....	72
6.2.1	<i>Test Cases</i> .....	73
6.2.2	<i>Test Case Results</i> .....	74
6.2.3	<i>Unit Tests</i> .....	75
6.2.4	<i>Heuristic Evaluation</i> .....	76
6.3	CONCLUSION.....	77
<b>7</b>	<b>CONCLUSION.....</b>	<b>78</b>
7.1	WALKTHROUGH.....	78
7.2	FUTURE WORK.....	79
7.2.1	<i>Unity Web Player</i> .....	79
7.2.2	<i>Heart Rate Monitoring</i> .....	79
7.3	PERSONAL STATEMENT.....	80
	<b>REFERENCES .....</b>	<b>81</b>
	<b>APPENDIX .....</b>	<b>83</b>

## Table of Figures

FIGURE 1: GANTT CHART .....	5
FIGURE 2: XBOX FITNESS .....	7
FIGURE 3: LLOYDS ONLINE DOCTOR .....	8
FIGURE 4: SYSTEM OVERVIEW .....	17
FIGURE 5: UNITY LOGO .....	18
FIGURE 6: UNREAL ENGINE LOGO .....	19
FIGURE 7: KINECT v1 .....	20
FIGURE 8: KINECT v2 .....	20
FIGURE 9: WINDOWS AZURE LOGO .....	21
FIGURE 10: AMAZON WEB SERVICES LOGO .....	22
FIGURE 11: PHP LOGO .....	23
FIGURE 12: PYTHON LOGO .....	23
FIGURE 13: ASP.NET LOGO .....	24
FIGURE 14: MONGODB LOGO .....	25
FIGURE 15: MYSQL LOGO .....	25
FIGURE 16: VISUAL STUDIO PUBLISH BUTTON .....	29
FIGURE 17: EXTREME PROGRAMMING METHODOLOGY .....	34
FIGURE 18: SCRUM METHODOLOGY .....	35
FIGURE 19: ITERATIVE AND INCREMENTAL METHOD .....	36
FIGURE 20: USE-CASE DIAGRAM .....	37
FIGURE 21: ENTITY RELATION DIAGRAM .....	38
FIGURE 22: SYSTEM ARCHITECTURE DIAGRAM .....	39
FIGURE 23: FLOWCHART .....	40
FIGURE 24: PATIENTVIEW PROTOTYPE .....	41
FIGURE 25: PATIENTDATA SCREEN .....	41
FIGURE 26: HEART HEALTH WEBSITE HOME PAGE .....	43
FIGURE 27: HEART HEALTH KINECT LOGIN PAGE .....	44
FIGURE 28: ERD .....	45
FIGURE 29: WINDOWS AZURE MONGO LAB .....	46
FIGURE 30: ASP.NET PROJECT CREATION .....	50
FIGURE 31: WINDOWS AZURE SITE CREATION .....	51
FIGURE 32: WINDOWS AZURE ALL ITEMS .....	51
FIGURE 33: PATIENTVIEW DRAWN PROTOTYPE .....	53
FIGURE 34: PATIENTVIEW SCREEN .....	53
FIGURE 35: PATIENTDATA DRAWN PROTOTYPE .....	54
FIGURE 36: PATIENTDATA SCREEN .....	54
FIGURE 37: KINECT v1 PROTOTYPE .....	55
FIGURE 38: BASIC WORKOUT SCREEN .....	56
FIGURE 39: KINECTMANAGER ATTACHED SCRIPT .....	56
FIGURE 40: RESTING HEART RATES[22] .....	59
FIGURE 41: KINETIK HEART RATE MONITOR .....	63
FIGURE 42: HEART BEAT ANIMATION .....	65
FIGURE 43: SIMON SAYS GAME SCREEN .....	66
FIGURE 44: HEART RACER GAME SCREEN .....	67
FIGURE 45: ORBS GAME SCREEN .....	67
FIGURE 46: APPLICATION LOGIN SCREEN .....	68
FIGURE 47: ACCOUNT INFORMATION SCREEN .....	68
FIGURE 48: UNITY ISSUE .....	70

## Table of Tables

TABLE 1: HEURISTIC EVALUATION .....	11
TABLE 2: FUNCTIONAL REQUIREMENTS .....	15
TABLE 3: GAME ENGINE COMPARISON .....	27
TABLE 4: MOTION CAPTURE COMPARISON .....	28
TABLE 5: CLOUD SERVICES COMPARISON.....	29
TABLE 6: SERVER-SIDE LANGUAGES COMPARISON .....	29
TABLE 7: DATABASE MANAGEMENT SYSTEM COMPARISON.....	31
TABLE 8: TEST CASES.....	73
TABLE 9: TEST CASE RESULTS .....	74
TABLE 10: SUMMARY OF UNIT TESTS (APPENDIX).....	75
TABLE 11: HEURISTIC EVALUATION (ANDREW DALY) .....	76
TABLE 12: HEURISTIC EVALUATION (STUDENT) .....	76



## Table of Code Snippets

CODE 1: PATIENTDETAILS BSON .....	45
CODE 2: PATIENTDETAILS CLASS .....	46
CODE 3: MONGOHELPER CLASS.....	47
CODE 4: MONGOHELPER VARIABLE .....	47
CODE 5: PATIENTDETAILS COLLECTION CREATION .....	47
CODE 6: PATIENTDETAILS QUERY .....	48
CODE 7: MEMBERSHIP PROVIDER STRING.....	48
CODE 8: SECURE LOGIN CODE .....	48
CODE 9: LOGIN BY QUERYING DATABASE .....	49
CODE 10: DOCTORLOGIN CLASS.....	49
CODE 11: SAMPLE CSHTML CODE .....	52
CODE 12: COLOURVIEW CODE.....	57
CODE 13: GESTURE DETECTION CODE .....	57
CODE 14: GETRESTINGHEARTRATE FUNCTION .....	60
CODE 15: GETMAXIMUMHEARTRATE FUNCTION .....	61
CODE 16: GETMAXIMUMHEARTRATEHEIL FUNCTION .....	62
CODE 17: SQUATPERFORMED FUNCTION .....	64
CODE 18: HEART BEAT ANIMATION CODE.....	65
CODE 19: GAME EXIT AND WORKOUT DATA SAVE CODE .....	66
CODE 20: USERNAME QUERY.....	69
CODE 21: CHECKUSERNAME FUNTION .....	69

## Table of Equations

EQUATION 1: FETAL HEART RATE .....	60
EQUATION 2: FUTHER ELABORATED FETAL HEART RATE .....	60
EQUATION 3: KARVONEN HEART RATE .....	61
EQUATION 4: HEIL HEART RATE .....	61

# 1 Introduction

Medical doctors and General Practitioners are highly respected by many people for their medical advice and diagnosis of illness. Becoming a doctor and helping people through illness has become a safe career choice for people with skill and nerve for the job. However, due to ever-changing and evolving technology is it possible that someday doctors may be obsolete? A lot of technology has been created in recent years that have made some professions obsolete due to the fact that a machine or some software can do it. Due to internet people now have the ability to get diagnosed by a doctor from their own home without even the need for the doctor to visit.

One type disease very common with many people is Heart disease. There are many types of Heart disease and to recover from such requires a lot of exercise and doctor appointments/check-ups.

This system plans to eliminate the need for patients to go to ongoing doctor check-ups and to allow for their exercise and rehabilitation to be performed from home.

Patients that require rehabilitation must perform consistent exercises as a crucial element in their overall physical and mental rehabilitation. However patients at home tend to either only follow their programmes for a short period of time or do not follow them at all. The inclusion of a creating a gamified environment to the system by means of Serious Games, try ensure that patients follow their exercise programme regularly and to view it as a fun activity.

This chapter will cover:

**Background:** The areas in which the project covers.

**Project Aims:** The Aims of this project and what is hoped to be achieved.

**Project Objectives:** Objectives to be completed over the course of the project.

**Walkthrough:** A walkthrough of the entire thesis.

**Timeline:** A Timeline represented as a Gantt chart to outline all timeline of the entire project.

**Conclusion:** Overview of the chapter.

## **1.1 Background**

### **1.1.1 Cardiovascular Diseases**

The main area of rehabilitation focused on for this project is rehabilitation of people with cardiovascular diseases. Cardiovascular Disease (or Heart disease) is a class of diseases that involve the heart, the blood vessels (arteries, capillaries, and veins) or both.

Patients that require rehabilitation for cardio-vascular diseases must perform consistent exercises as a crucial element in their overall physical and mental rehabilitation. These exercises help the patient in many ways: [1]

- Strengthens your heart
- May improve congestive heart failure symptoms
- Lowers your blood pressure
- Makes you stronger
- Helps you reach (and stay at) a healthy weight
- Helps manage stress
- Boosts your mood and self-esteem
- Improves sleep

However patients at home tend to either only follow their programmes for a short period of time or do not follow them at all. By using Serious Games with this project, patients may follow their exercise programmes regularly and to view it as a fun activity.

### **1.1.2 Doctor-Patient Interaction**

Communication between doctor and patient is a key success factor in medical treatment. [2] And as such the doctor–patient relationship is central to the practice of healthcare and is essential for the delivery of high-quality health care in the diagnosis and treatment of disease or rehabilitation. Doctors must maintain a professional rapport with patients, uphold patients’ dignity, and respect their privacy.

In terms of online medical systems between doctors and patients, there has been an emergence of websites set up for booking appointments, order repeat prescriptions and view your medical record with certain Doctors/General Practitioners depending on if they choose to use the website.

The planned system will be in some ways similar to these online medical systems between doctors and patients except that results taken from patient's exercises will be sent straight to the doctor without the need for direct communication and check-ups and to allow for their exercise and rehabilitation to be performed from home.

### **1.1.3 Serious Games**

Serious games can be thought of as any game based interfaces that have been designed for any purpose other than entertainment. Serious games have been researched in the areas of: military, health, government and education.[3] In this case, Serious Games can be used in the area of health. The Serious games aspect of the may be a solution to motivate patients to continue exercises in an engaging way. Patients are immersed in the game simulation and play the games regularly and as such attribute to their rehabilitation.

Lots of research has been conducted in using serious games in terms of rehabilitation. Effective rehabilitation must be early, intensive and repetitive. Serious games provide a means to maintain motivation for people undergoing therapy [4] by means of exercises. Games of virtual reality and imaging of webcam-based games are usually the solution to provide an engaging and motivating tool for physical rehabilitation.

## **1.2 Project Aims**

Research and Deliver Home-based rehabilitation system for patients in rehabilitation of cardio-vascular diseases with a user-friendly interface and also a doctor-patient interaction system for doctors to diagnose and examine patients without face-to-face interaction.

### **1.3 Project Objectives**

1. Research if project feasible.
2. Research similar technologies to determine user requirements for this project.
3. Research technologies to determine the most appropriate implementation of this project.
4. Implement exercise tracking in patient system.
5. Implement serious game in patient system.
6. Implement doctor system.
7. Implement Heart-rate tracking in patient system.

### **1.4 Walkthrough**

#### Chapter 2: Research and Analysis

Similar Solutions to Heart Health will be covered in this chapter and a heuristic evaluation of these systems, as well as Research Topics which are covered within the project and additional research in the form of an interview.

#### Chapter 3: Research of Technologies

This chapter will analyse the proposed project, state its components and display the user requirements of the system.

#### Chapter 4: Design

This chapter will outline the overall design of the system with the use of diagrams as well as discuss the methodologies which were research and then decided on.

#### Chapter 5: Implementation

This chapter will outline the overall implementation of the project, as well outline the problems which were encountered and how they were solved.

#### Chapter 6: Testing and Evaluation

This chapter will discuss the approach taken to testing the overall system. It will discuss the test-cases that are needed to ensure the end system is working correctly, and also outline the tests which were taken and results from these tests.

## Chapter 7: Conclusion

This chapter will conclude the project; give a walkthrough on what was covered within the thesis, what was learned from the course of the project, future plans for the system.

## 1.5 Timeline

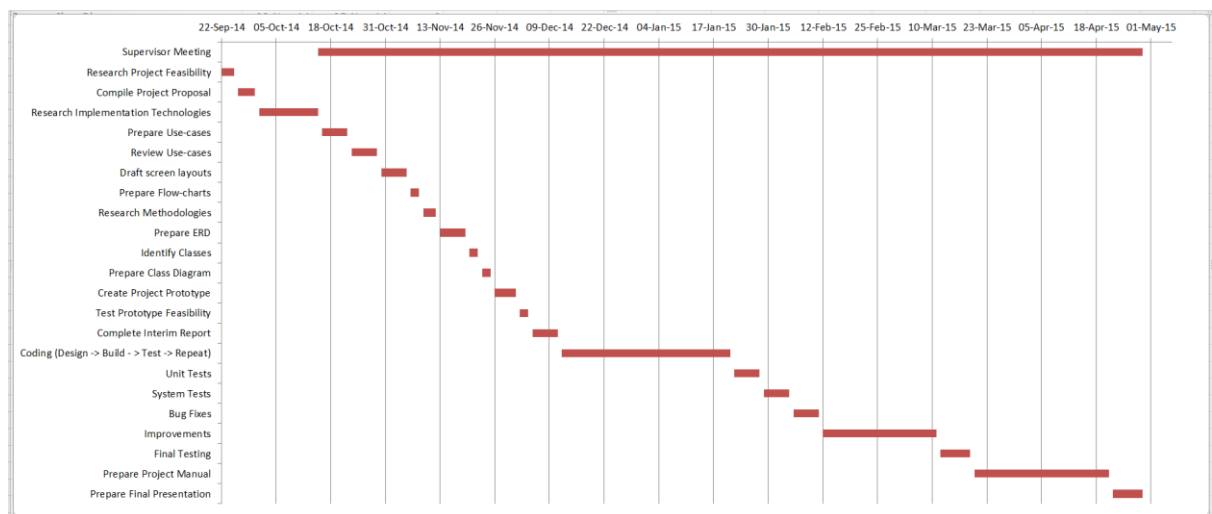


Figure 1: Gantt Chart

Project Timeline represented as a Gantt chart (Figure 1).

Meetings with Supervisor (Bryan Duggan) takes place every Wednesday which was agreed upon at the first meeting. Both College Semesters are divided with Research and Design taking place mainly in the first semester and implementation taking place more so in the second semester.

## 1.6 Conclusion

This chapter has covered the Background topics of the project in relation to Cardio-Vascular Diseases, Doctor-Patient Interaction and Serious games, which are all covered throughout the course of the project. The Aims and Objectives of the project which have been achieved are detailed and proven in the upcoming chapters.

## 2 Research and Analysis

This Chapter Covers the all the research involving important elements of what the project covers, similar alternatives to the project and heuristic evaluations of these alternatives.

Sections Covered:

**Similar Systems:** The Similar Systems section discusses systems similar to the project itself.

**Heuristics:** Discusses what heuristics is and then displays the results of a heuristic evaluation of the Similar Systems.

**Research Topics:** The Research Topics looks into important topics to be researched which relate to the system.

**Additional Research:** Additional Research Performed as part of the project is displayed.

**Proposed System Requirements:** The Use-case requirements for the system are displayed.

**Conclusion:** Summary of what was done within this chapter.

### 2.1 Similar Systems

Research of existing Doctor-Patient, Home-based Rehabilitation systems which implemented motion capture and heart-rate tracking showed little or no results. The alternative solutions found are solutions which closely relate more to the two parts of the system, the Patients system and the Doctors system. The Systems chosen are Xbox Fitness, Lloyds Online Doctor. Each of these systems complies with necessary components of the project with Xbox Fitness focusing on the Motion Tracking, Heart Rate Monitoring and Serious Games aspect, Lloyds Online Doctor focusing on the Doctor-Patient Online Interaction.



### 2.1.1 Xbox Fitness



*Figure 2: Xbox Fitness*

The biggest to similarity to a system which tracked heart rate as exercises were performed was Xbox Fitness (Figure 2). Xbox Fitness is a game /service exclusion to Xbox One which uses the Kinect V2 sensor for motion capture as well tracking the player's heart rate and estimating their calorie amount burned based on their workout. [5] The workouts are given by professional fitness trainers and given as a workout training video where the user works out along with the video. Based on reviews of Xbox Fitness, it has gained much success, with accurate tracking of the Kinect and excellent workouts and basing results of workouts by tracking heart rate.

### 2.1.2 Lloyds Online Doctor

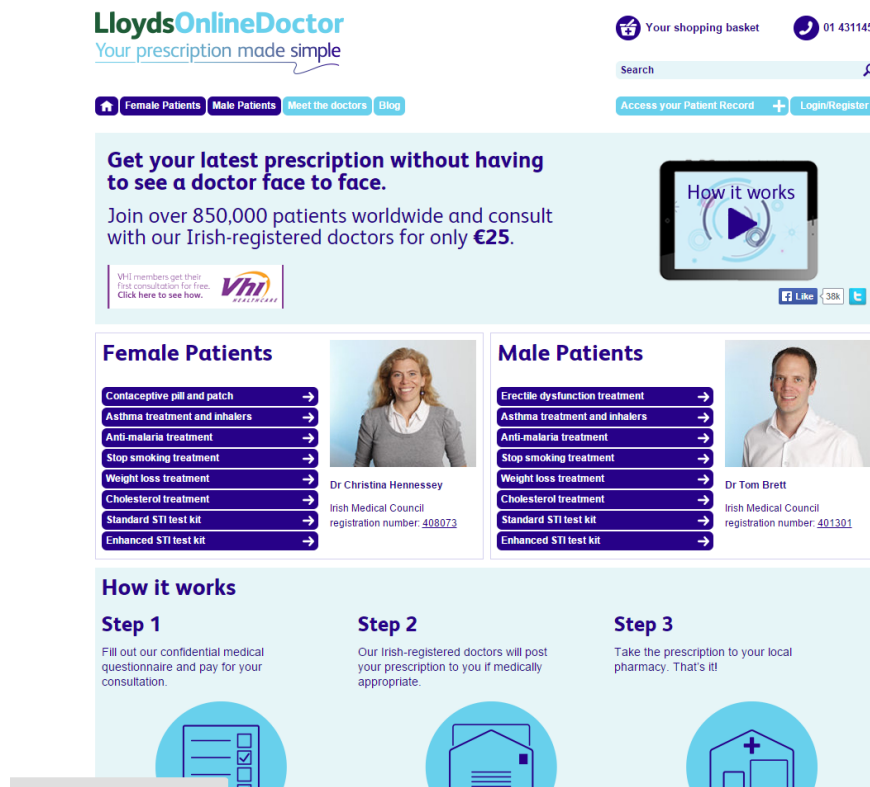


Figure 3: Lloyds Online Doctor

In relation to the Doctor's system the biggest similarity to an online Doctor-Patient system was Lloyds Online Doctor (figure 2.2). In the system patients can choose their own doctor to diagnose them based on their condition. Users also have the ability to fill out a medical questionnaire and from there, doctors on the site evaluate the questionnaire and post out prescriptions to the patients if medically appropriate. [6]

## 2.2 Heuristics

Heuristics in relation to Computer Science is the study of usability by means of a heuristic evaluation of a software user interface. To identify problems in the user interface (UI) design, it is judged against 'Nielsen's heuristics' which are a set of ten usability principles. The main goal of heuristic evaluations is to identify any problems associated with the design of user interfaces and thus determine its quality. The most commonly used heuristics were developed by Jakob Nielsen in 1994 (Nielsen 1994, 116-147). [7]

### 2.2.1 Usability Principals

- **Visibility of System Status**

The system should keep users informed about what is going on using appropriate feedback or notifications within reasonable time. [8]

- **Match between System and the Real World**

The system should follow the real-world conventions when informing or displaying information to users. Phases should be user oriented rather than system oriented, the system should use real world symbolism and conventions in order to present information to the user in a natural manner. [8]

- **User Control and Freedom**

The user should always be able to leave sections of the program they have selected by mistake. Nielsen refers to these methods as “emergency exits” for the user. The user should be able to escape any situation and not have to endure long dialogue should they choose to leave a section of the system. The escape route should be clearly marked and consistent throughout the system. The system should also support undo and redo. [8]

- **Consistency and Standards**

The system must be consistent with its terminology or icons, Users should not have to wonder whether different words, icons, actions or situations mean the same thing. [8]

- **Error Prevention**

Users should not be allowed to create errors within the system, try to account for all situations where an error may occur within the system and fix them, if it is not possible to prevent whatever error may occur, make sure the user can easily recover from the error. Clear language must be used to indicate what the problem is and why it occurred. There should be no need for error messages within the system unless they are necessary. [8]

- **Recognition over Recall**

Users should not have to remember information from different parts of the system, reduce the need for the user to memorise what action each control performs within the system. [8]

- **Flexibility and Efficiency of Use**

The system should be tailored to all levels of users (users who are computer-literate, and user who are not, or those familiar with the system itself). Expert level users should be able to use accelerators (such as key-bindings) to speed up interaction within the system. Accelerators meant for expert users should be hidden from novice users to avoid confusion with the simplest form of the system or information overload. [8]

- **Aesthetic and Minimalist Design**

Every piece of information within the system should be relevant and have a purpose or meaning. Irrelevant information may diminish the most critical information in displayed within the system and adds to a reduction in user visibility. [8]

- **Help and Documentation**

Users should not have to use a manual or documentation to use the system, however if the system is of a certain complexity, it may need to be documented. If a manual or documentation is necessary, it should be freely available, easy to locate and contain material only relevant to the system. [8]

- **Help User Recognize, Diagnose, and Recover from Errors**

Should the system display error messages, it should be written in a language the user would understand and not contain system code to confuse the user. It should also indicate to the user where the error was made so it can be avoided in future. [8]

### **2.2.2 Usability Scoring**

Using the Usability Principles stated above, each of the similar systems will be compared against the principles. The ten usability principles will be used as a rough

scoring method with a maximum score of ten based on each of the principles. If the system complies with a principle, it will be score out of ten. As such each principle will be given a rating from 1 to 10.

A heuristic evaluation will take place of the completed project during the testing phase.

	<b>Xbox Fitness</b>	<b>Lloyds Online Doctor</b>
<b>Visibility of System Status</b>	9	6
<b>Match between System and the Real World</b>	7	7
<b>User Control and Freedom</b>	5	7
<b>Consistency and Standards</b>	7	8
<b>Error Prevention</b>	9	6
<b>Recognition over Recall</b>	8	5
<b>Flexibility and Efficiency of Use</b>	7	4
<b>Aesthetic and Minimalist Design</b>	10	5
<b>Help and Documentation</b>	7	3
<b>Help User Recognize, Diagnose, and Recover from Errors</b>	8	9

*Table 1: Heuristic Evaluation*

## 2.3 Research Topics

Topics to be covered within creation of the Heart Health system are Heart Rates and how they are measured.

### 2.3.1 Heart Rates

Heart rates are never a stable value and every person has a different heart rate relative to different factors. Heart rates decrease or increase to maintain equilibrium within the human body between requirement and delivery of oxygen and nutrients. [9] The rates at which heart rate is assessed are a person's Resting Heart Rate, their Maximum Heart Rate and their Target Heart Rate. [10]

#### **Resting Heart Rate**

Resting Heart Rate is a person's heart rate while not active exercise, in a neutrally temperate environment, and not under any form of stress or surprise. The typical

resting heart rate for an adult is generally 60-100 beats per minute. However Resting Heart Rates are very much dependant on the persons state such as their age, their level of fitness (such as if they are athletic) and their gender.

### **Maximum Heart Rate**

Maximum Heart Rate is the highest heart rate a person can achieve. Maximum heart rates in a person decrease by age. [11] Weight as well as a person's level of fitness can also decrease a person's Maximum heart rate.

### **Target Heart Rate**

Target Heart Rates are generally used to find a desired range of heart rate reached during an exercise. They are taken into account to ensure an exercise is done properly and enables a person's heart and lungs to gain the most benefit from a workout. [12]

## **2.4 Additional Research**

Interview with Zachery Tan (Orthopaedic surgeon).

### **Interview**

*1. What do you do? (Occupation)*

I'm an Orthopaedic surgeon at Bon Secours hospital in Glasnevin.

*2. Do you work with many patients with cardio-vascular disease(s)?*

Yes!

*3. What are the main exercises patients with cardio-vascular disease(s) most perform?*

There are no particular set exercises for people who suffer with cardio-vascular diseases. Depending on the type of stage patients are in terms of heart disease,

patients must perform a lot of cardio exercises in a gym before surgery of a heart bypass and after surgery they must perform light exercises such as walking or swimming. Check out DCU's weight loss programme for more information on the specific type of exercises for these people.

*4. What is the average timeline of improvement for people going through this rehabilitation?*

Depending on the severity of heart problems, it can take very long, however exercise is definitely crucial in recovery, if patients do minimum exercises then there is little or no recovery, however if patients actually do exercises there is a big improvement, improved health and patient quality of life.

*5. Would you say people adhere to their exercise programmes?*

No, most patients do not and because of this their quality of life doesn't change, patients who became ill due to smoking habits or lack of exercise just get worse by not performing any exercise even though they know they must. They also continue with bad smoking habits even though they should not.

*6. What other diseases are there where exercises must be performed as part of rehabilitation?*

Everything! Diabetes, endocrinology diseases, Orthopaedic conditions, etc.

*7. What exercises must be performed in relation to said disease(s)?*

There are no particular set exercises for any of them, basically the part of the body which is damaged or diseases needs to be exercised. For example, if there is damage to a patients knee, they must perform knee physio exercises which improves muscle build on the knee and increases a lot in the joint.

*8. Do you think this system will work for people in rehabilitation?*

Link the system for Physiotherapists and not doctors as people who are in rehabilitation and must perform exercises are in more contact with physiotherapists

than doctors. The Elderly must be taken into account as they are the majority of people with cardio-vascular diseases.

*9. Your opinion of the system, any additional input for the system?*

System sounds good, however need a bit more medical experience, have check counters for maximum heart rate of different types of patients, also different levels of exercises need to be implemented for different types of people depending on their fitness, for example obese patients exercises versus patients who are in good physical condition and used to run marathons. Make sure to test the system on normal people as well as people with heart conditions.

### **Things Noted**

Doctors will not be the main users of Doctor-Patient system, Physiotherapists will most be in charge of patient check-ups, however for this project, Doctor's will be used to refer to the group of Physiotherapists, Doctors, General-Practitioners and whoever else may make use of the system.

There may be different levels of exercises depending on the patient to be implemented in both exercise and serious game of patient's system, on patient account creation doctor (Physio) will specify correct level.

## **2.5 Proposed System Requirements**

These are the Proposed System Requirements for the System based off the Research done, as well as personal opinion on what requirements should be included within the system.

Req ID	Name Of	Description	Priority	User	Use-case
--------	---------	-------------	----------	------	----------



	Requirement			Contact	Ref
1	Log In (Patient System)	Login to Patient system, Account created by Doctor	Medium	Patient	7
2	Perform Basic Exercise	Perform basic exercises in front of Kinect, results are recorded	High	Patient	1
3	Play Game	Play game-based exercises in front of Kinect, results are recorded	Low	Patient	1
4	View Records (Workouts)	Patient/Doctor Can view results of patient workout, Doctor comments also available	High	Patient, Doctor	1,8
5	Create Doctor Account	Create account on Doctor system	Medium	Doctor	
6	Log In (Doctor System)	Login to doctor system	High	Doctor	5
7	Create Patient Account	Create user account for patient	Medium	Doctor	6
8	Select Patient	Select specific patient account	High	Doctor	6,7
9	Edit Patient Details	Can edit selected patient account	Low	Doctor	8
10	Send Message	Send message to patient	Low	Doctor	8
11	Comment On Workout	Comment on specific workout by patient	Low	Doctor	4

*Table 2: Functional Requirements*

## 2.6 Conclusion

This chapter has covered Similar Alternative to the project of Xbox Fitness and Lloyds Online Doctor. A heuristic evaluation was also performed on these Alternatives. Research into heart rates has shown valuable information into functions which may be used within the system.

The functional requirements were also made.

## 3 Research of Technologies

This chapter analyses a range of technologies that have been researched for this project. An overview is to be provided for each technology as well as section with compares them and the reasons for choosing them. An overview of the system so far is also done

This chapter covers:

**Architecture:** The architecture section details what the system is composed of and the components which are to be implemented.

**Available Technologies:** The Available technologies section discusses alternative technologies which were possibilities as the chosen technologies for the project.

**Review:** The Review section compares the Available technologies and details why they were not chosen.

**Additional Software:** Additional Software section details additional software which was used in creation of the system

**Conclusion:** Summary of what was done within this chapter.

## 3.1 Architecture

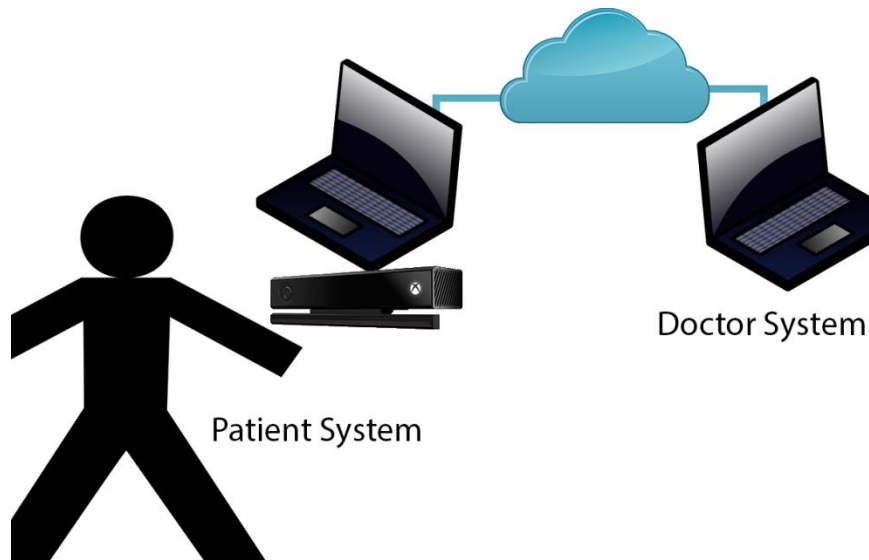
The Architecture of the System is all the components in which the system is composed of.

### 3.1.1 Project Overview

Heart Health is a home-based rehabilitation system which uses the Xbox Kinect 2. Users of the system are usually patients in treatment of any type of Cardio-Vascular Disease. User must perform exercises in front of the Kinect, Exercises are played out on the form of a game, different exercises are provided and patient's heart rate is monitored. The System is also connected online to the cloud for patient's doctor /physiotherapist to view. Doctor / physiotherapist views how well exercises are performed and also keeps up to date with the patient without the need of continuous doctor visits.

Rehabilitation is in areas of Cardio-Vascular Diseases and Monitoring Heart-rate. Exercises may include: Squats, Jumping Jacks, Various Stretches and Basic Body Movements, etc.

### 3.1.2 Project Components



*Figure 4: System Overview*

Patient's System; Computer with Game/Rehabilitation Application and Kinect 2 attached.

Admin/Doctor's System; Computer with Admin Application.

Patient's System; System will have the game and rehabilitation workout routines, Patient logs into their user-account (which is set up by Doctor) and application commences. Patient has ability to commence normal workouts, perform workouts in Gamified environment or view records from pervious workouts as well as comments from Doctor.

Admin/Doctor's System; System will be a Doctor check-up system, Doctor logs into system, can view patients, create account for patient, can view specific patient records and send comment to patient on record to give progress, there may also be an ability to view exact patient workout.

## 3.2 Available Technologies

The system can be split into sections depending on how each function was made with what technology or what technology is used for a function. Different technologies can

be chosen for what Game Engine is used to make the Application, what Motion Capture device is used to capture the data, what cloud service is used to host the website, what Database management system is used to store the data or what Server-side Language is used to make the website.

### 3.2.1 Game Engines

The Patient application which includes the Serious Game aspect of the system was decided to be built using a game engine as game engines contain all the tool available for building game-based systems without the need of coding them which would take a large amount of time.

Deciding on the proper game engines is important as different game engines may or not be able to implement some aspects of the system. Some game engines are not able to support the motion capture device of the Xbox Kinect which may be a big deciding factor on whether it will be used in implementation of the system.

#### Unity3D



*Figure 5: Unity Logo*

Unity3D (or just Unity) is a cross-platform game creation system with its own game engine and a built-in Integrated Development Environment (IDE) developed by Unity Technologies. Games built using Unity can be made for PC platforms like Windows and Mac, or consoles like the Xbox 360 and PlayStation 3, and even mobile devices like Android and iPhone.

Developers can program in scripts like C#, JavaScript or Boo (Python inspired syntax). MonoDevelop is the IDE built into Unity. Scripts do not need to be written with the built-in MonoDevelop, however MonoDevelop necessary to debug them.

Unity3D has many plugins to implement many different types of motion capture. In terms of Microsoft's Kinect, Microsoft has released their own plugins for Unity to implement Kinect Version 1 and Version 2 projects. [13]

### **Unreal Engine**



*Figure 6: Unreal Engine Logo*

The Unreal Engine is a game engine developed by Epic Games. The Unreal Engine also comes with the Unreal Development Kit, a game creation system for developing Unreal Engine games. The current release is Unreal Engine 4. Unreal Engine 4 uses C++ for scripting.

### **3.2.2 Motion Capture**

Deciding on the right motion capture device and software is very important in this project. The Xbox Kinect being the main open-source (depending on if you own a Kinect) motion capture device is primarily chosen. However depending on which version of the kinect certain aspects are required for the project such as the whichever

version of the kinect has the ability of heart rate monitoring, a necessary component of this project.

## **Kinect**



*Figure 7: Kinect v1*

The Kinect (Kinect version 1) (Figure 7) is a motion sensing input device developed by Microsoft and PrimeSense for the Xbox 360 console and Windows PCs. Based on a webcam style peripheral device, it acts as an alternative type of input and allows users to interact with their console/computer without the need of a game controller.

## **Kinect 2**



*Figure 8: Kinect v2*

The Kinect V2 (Figure 8) is a motion sensing input developed by Microsoft as the predecessor to the Kinect. It is developed for the Xbox One console and Windows PCs. It holds most of the same functions as its predecessor except many new features

as well as upgraded features. It has the ability of heart-rate tracking by using its infrared sensor.

### 3.2.3 Cloud Services

Cloud Services for the system need to be decided upon in terms of web-hosting of the Doctor's System as well as database storage of Patient-Doctor account credentials as well as patient exercise results.

Choosing the correct cloud service depends on which programming languages it supports to facilitate the application we have to deploy. In cloud computing terms this is known as Platform-as-a-Service, which means that an application is created using tool and/or libraries from the provider. The provider provides the network, servers, storage and other services that are required to host an application.

#### Windows Azure



*Figure 9: Windows Azure Logo*

Windows Azure is Microsoft's cloud-computing platform. Consumers of Windows Azure can build, deploy and manage applications through a global network of Microsoft-managed datacentres. It also provides web-hosting and the ability to create and deploy Virtual Machines. Users can develop applications in Java, PHP, Python, ASP.NET and Node.js. Windows Azure has the ability to allow any language or framework, to be deployed onto the cloud however middleware must be used for some to work, and client libraries are available on GitHub.

## Amazon Web Services



*Figure 10: Amazon Web Services Logo*

Amazon's solution to cloud computing is Amazon Elastic Compute Cloud (EC2) which is part of Amazon's Web Services.

Amazon Web Services holds many tools especially for developers such as load balancers, relational databases add-ons, caching systems, notification systems and content delivery networks. These are constantly being optimized by Amazon and more services and online tools are constantly being added. [14] It also provides support for every major programming language.

### 3.2.4 Server-Side Languages

Communication between the client and server is an essential part of the Doctor's System. The Doctor's System must perform many specific functions such as creating patient accounts, viewing patient accounts and view patient workout results and heart rate condition. Server side languages perform the programming functions of the system on the server. Once the task has been completed here, the result is returned to the client through the front end.

## PHP





*Figure 11: PHP Logo*

PHP is scripting language also used as a programming language. PHP is probably one of the most popular languages in terms of creating server side web applications. PHP's syntax is very similar to that of languages like Java and C++, making PHP familiar and easy to learn. It can be considered as an option as I am very familiar with it, however it is considered outdated by many because of the more recent web frameworks available which allow for less code and faster completion.

**Python (Django)***Figure 12: Python Logo*

Python is a high-level programming language which has a unique code syntax which expresses concepts in fewer lines of code than other high-level programming languages. Python can be written for applications and web-based applications.

Django is a free and open source web application framework, written in Python. Django's motivation lies in the "intensive deadlines of a newsroom and the stringent requirements of the experienced Web developers who wrote it". [15] Which means that it is a fast and easy to manage framework, it is also readable for people not exactly familiar with the framework, which is of benefit to myself as I am neither familiar with Python or Django. It follows the Model, View, Controller class technique where every class involved in the framework must be organised into specific roles of Models (Storing Functions and Information), Views (Output of information) and Controllers (Managing information and connecting the view and model classes).

**ASP (ASP.Net)***Figure 13: ASP.NET Logo*

ASP.NET is an open source 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. [16] Any .Net language can be used along with ASP.NET. ASP.net can be also used with HTML, CSS and JavaScript. There are different types of frameworks to be used with ASP.NET of ASP.NET MVC Framework. Since 2010 a new type of syntax has been developed to be used with ASP.net of Razor view engine. It was released along with ASP.NET MVC 3 and allows for creation of dynamic webpages using C#, it uses a new filename extension of .cshtml, which allows execution of C# code within webpages.

**3.2.5 Database Management System**

Deciding on the right database management is crucial to how the whole system operates. The system requires a Database management system to keep track of doctor accounts, patient accounts and patient records. One of the main deciding factors on picking the right Database management system is that will it be compatible with each component in the system. For example, does the Cloud service support it? Can the patient application connect to it? Can the doctor system connect to it? Is it the best Database management system to organise and store the data in?

Of the two examples compared, a comparison is done of whether to use a NoSQL Database and a Relational Database, which is also a point to be consider in choosing the correct database.

## MongoDB (JSON)



*Figure 14: MongoDB Logo*

MongoDB is the most popular open source document-oriented database. Classified as a NoSQL database, [17] MongoDB performs differently to the normal table-based relational database structure of most popular databases and stores data in JSON (JavaScript Object Notation) documents (known as BSON with MongoDB). Being a different type of database management system to the traditional relational database system MongoDB does contain some limitations such it cannot join different data types across the database, a functions which is commonly used in most relational databases, however it does allow for data querying.

## MySQL



*Figure 15: MySQL Logo*

MySQL is an open-source relational database management system. MySQL is one of the most popular database management systems in the world with being open source a big part of the reason. Large organisations such as Facebook, Google and YouTube still make use of MySQL databases. A thing to note is that SQL Developer can also be used as a front end solution to interface with the MySQL backend. [18]

### 3.3 Review

#### 3.3.1 Game Engines

On comparison both Unity and Unreal Engine (UE4) are both superb game engines, however Unity wins outright for many reasons above Unreal in terms of my own preference and reasons to do with the project based on the following factors from my own experience working with them

To note, I was already familiar with Unity, having worked on many projects before with it which aided my decision, however I gave Unreal a chance and looked into what it had to offer.

	Unity	Unreal
<b>Programming/Scripting Languages</b>	C#, JavaScript, Boo	C++
<b>Kinect V2 Compatible</b>	Microsoft Officially released Kinect V2 Plugin	User-made plugins still in development for UE4
<b>Hardware requirement</b>	Unity can run on low spec system and still make good games	Unreal requires high spec system for good games
<b>Cross Platform</b>	Windows, Mac, Wii, iPhone,	Windows, Mac, Wii,

	iPad, Android, PS3, Xbox, Xbox 360, Web Applications	iPhone, iPad, Android, PS3, Xbox, Xbox 360, Web Applications
<b>License</b>	All tools to create game are free, except for pro which only has Glass refactoring and shadows	Free, but limits to many tools unless paid
<b>Stability</b>	Rarely Crashes	Crashes Often
<b>Learning Curve</b>	Very User friendly and easy to understand	Difficult to understand User interface, however much documentation available

*Table 3: Game Engine Comparison*

Unreal had a bigger learning curve and different User Interface to Unity which would of taken time to get a hang of. Unity is also free unless you want the Unity Pro paid edition which only include small nit-picking features such as glass refactoring and shadows whereas Unreal Engine is not free. One last important which greatly influenced my decision was that there is no official plugin or support for the Unreal Engine to use the Kinect or Kinect 2. I had found that some people created plugins for the engine however Microsoft released their own plugins especially for Unity for the Kinect v2 and on testing them they worked fine. Also to note C# will be the scripting language used in Unity3D as I have more experience using it.

### 3.3.2 Motion Capture

On comparison of both the Kinect and the Kinect 2, the Kinect 2 is the outright winner for the obvious reason of its ability to measure heart rate, however it is also good to note the positive things of the original Kinect.

	<b>Kinect</b>	<b>Kinect 2</b>
<b>Color Camera [19]</b>	640 x 480 @30 FPS	1920 x 1080 @30 FPS
<b>Depth Camera [19]</b>	320 x 480	512 x 424
<b>Skelton Joints [19]</b>	20 joints	26 joints
<b>Number of Skeletons Tracked</b>	2	6
<b>Bone Orientations [20]</b>	No	Yes
<b>Measuring Heart Rate</b>	No	Yes
<b>Muscle Simulation [20]</b>	No	Yes
<b>Face Tracking</b>	Yes	Yes
<b>Recognizing Expressions</b>	No (You must Write	Yes

[20]	Algorithm to do so)	
<b>Hardware requirement</b>	Either USB 2.0 or USB 3.0	PC must have USB 3.0
<b>Cost</b>	€30+ (Retail Price)	€95+ (Retail Price)
<b>SDKs Available</b>	Kinect for Windows SDK v1.8, OpenNI v2.2	Kinect for Windows SDK v2.0
<b>Unity Plugins</b>	Kinect with MS-SDK (Unity Asset Store), Many more available	KinectForWindows_UnityPro_PublicPreview, Available from Microsoft

*Table 4: Motion Capture Comparison*

Based on the results, in comparison the Kinect V2 with its new and upgraded features as opposed to its predecessor wins outright, but it is important to note that it's 2 flaws which can influence a decision in a different scenario is its Hardware Requirement and its Cost. A lot of computers, some custom-built or old models, do not have a USB 3.0 port and as such the Kinect V2 will not work without one. Cost may be a big part of influencing decision as well and these factors must be taken into account in the implementation of this project. The Project is stated for home-based rehabilitation and if the patient does not have the money for an expensive Kinect V2 or their PC does not have a USB 3.0 then the Patient's system will be of no use.

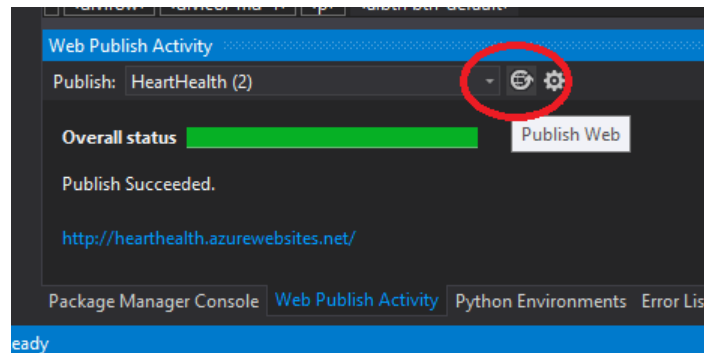
### 3.3.3 Cloud Services

The main things to be considered on picking the right cloud service are the Programming Languages it supports, and the Database Management system it already supports.

	<b>Windows Azure</b>	<b>Amazon</b>
<b>Programming Languages Supported</b>	Java, PHP, Python, ASP.NET and Node.js	Supports every main programming language except Node.js
<b>Database Management Systems Included</b>	MySQL, Oracle, SQL Azure, DocumentDB, Storage Blobs, Storage Tables, MongoDB (MongoLab)	MySQL, Oracle, SQL Server, PostgreSQL, Aurora
<b>Ease of use</b>	Beginner friendly (A lot of Documentation)	Said to be extensive and diverse and difficult to newcomers [14]

*Table 5: Cloud Services Comparison*

Windows Azure seems to win out in terms of Cloud Services not so much for the Programming Languages it supports or its Database Management Systems but it's Ease of use and friendly User interface which makes it easy to adjust for newcomers to cloud software like myself. Another factor which added to Windows Azure being chosen is its compatibility with Visual Studio, any web applications written can be published straight away with just one click.

*Figure 16: Visual Studio Publish Button*

### 3.3.4 Server-Side Languages

For the Doctor system choosing the correct language depends on preference more than anything, however some languages would be more suitable than others for some reasons.

	PHP	Python (Django)	ASP.NET
<b>Ease of Use</b>	Average Ease of Use	Known for simplicity	Easier since released of Razor view engine
<b>Web-Based</b>	Yes	Yes	Yes
<b>Windows Azure Compatible</b>	Yes	Yes	Yes

*Table 6: Server-Side Languages Comparison*

It should be noted that at the start Python with Django was chosen as Server-side language to be used within the project for the reason of gaining experience into different types of programming languages. However ASP.NET was chosen instead due to the fact that the project's database classes to connect to the MongoDB storage

were already written in C#, the classes had to be written in C# as they needed to be the same as in the Patient's system (Unity application) which was written in C#. It was too difficult to convert the classes to Python or even to create DLLs to be used within Python, Iron Python was a solution to this, but it was decided instead to use ASP.NET which perfectly supported C# and any DLLs which contained my database classes.

So it was decided that ASP.NET MVC 4 (recently released) was to be used as the Server-side scripting language.

### 3.3.5 Database Management

The two Database Management Systems to be compared are totally different types of management systems, one being the new NoSQL database and the being the well-known relational tabular database management system. This comparison is really to compare which is the best way to store data, by using the well-known SQL or the new NoSQL method of JSON.

	MongoDB	MySQL																																																																																																		
Data Storage	Collection (Document-Oriented Storage with BSON)	Tabular-Storage (SQL)																																																																																																		
Data Querying	Yes	Yes																																																																																																		
Windows Azure Compatible	May require some work to set up by means of Virtual Machines or can install MongoLab onto Azure	Already set up, just allocate that it is needed.																																																																																																		
Unity3D Compatible	Can download .dlls to connect to MongoLab	Can already connect to MySQL server																																																																																																		
Storage Format	BSON <pre>{   person: {     first_name: "Peter",     last_name: "Peterson",     addresses: [       {street: "123 Peter St"},       {street: "504 Not Peter St"}     ],   }, }</pre>	SQL <table><tr><th></th><th>FirstName</th><th>LastName</th><th>Email</th><th>Phone</th><th>Position</th><th>Branch</th></tr><tr><td></td><td>Andrew</td><td>Fuller</td><td>afuller@contoso.com</td><td>(205) 555 - 9898</td><td>CEO</td><td>TopManagement</td></tr><tr><td></td><td>Jeremy</td><td>Boather</td><td>jboather@contoso.com</td><td>(205) 555 - 9888</td><td>President QA</td><td>QA</td></tr><tr><td></td><td>Anne</td><td>Dodsworth</td><td>adodsworth@contoso.com</td><td>(205) 555 - 9887</td><td>VP QA</td><td>QA</td></tr><tr><td></td><td>Alexander</td><td>Tuckings</td><td>atuckings@contoso.com</td><td>(205) 555 - 9886</td><td>Team Lead...</td><td>QA</td></tr><tr><td></td><td>Brenda</td><td>Smith</td><td>bsmith@contoso.com</td><td>(205) 555 - 9885</td><td>Senior QA</td><td>QA</td></tr><tr><td></td><td>Mary</td><td>Bird</td><td>mbird@contoso.com</td><td>(205) 555 - 9885</td><td>Team Lead...</td><td>QA</td></tr><tr><td></td><td>Steven</td><td>Buchanan</td><td>sbuchanan@contoso.com</td><td>(205) 555 - 9897</td><td>President ...</td><td>Development</td></tr><tr><td></td><td>Robert</td><td>King</td><td>rking@contoso.com</td><td>(205) 555 - 9896</td><td>VP Dev De...</td><td>Development</td></tr><tr><td></td><td>Laura</td><td>Callahan</td><td>lcallahan@contoso.com</td><td>(205) 555 - 9892</td><td>Team Lead...</td><td>Development</td></tr><tr><td>0</td><td>Jason</td><td>Roland</td><td>jroland@contoso.com</td><td>(205) 555 - 9872</td><td>Senior Dev</td><td>Development</td></tr><tr><td>1</td><td>Eric</td><td>Danstin</td><td>edanstin@contoso.com</td><td>(205) 555 - 9882</td><td>Team Lead...</td><td>Development</td></tr><tr><td>2</td><td>Elizabeth</td><td>Lincoln</td><td>elincoln@contoso.com</td><td>(205) 555 - 9862</td><td>Senior Dev</td><td>Development</td></tr><tr><td>3</td><td>Margaret</td><td>Peacock</td><td>mpeacock@contoso.com</td><td>(205) 555 - 9852</td><td>Senior Dev</td><td>Development</td></tr></table>		FirstName	LastName	Email	Phone	Position	Branch		Andrew	Fuller	afuller@contoso.com	(205) 555 - 9898	CEO	TopManagement		Jeremy	Boather	jboather@contoso.com	(205) 555 - 9888	President QA	QA		Anne	Dodsworth	adodsworth@contoso.com	(205) 555 - 9887	VP QA	QA		Alexander	Tuckings	atuckings@contoso.com	(205) 555 - 9886	Team Lead...	QA		Brenda	Smith	bsmith@contoso.com	(205) 555 - 9885	Senior QA	QA		Mary	Bird	mbird@contoso.com	(205) 555 - 9885	Team Lead...	QA		Steven	Buchanan	sbuchanan@contoso.com	(205) 555 - 9897	President ...	Development		Robert	King	rking@contoso.com	(205) 555 - 9896	VP Dev De...	Development		Laura	Callahan	lcallahan@contoso.com	(205) 555 - 9892	Team Lead...	Development	0	Jason	Roland	jroland@contoso.com	(205) 555 - 9872	Senior Dev	Development	1	Eric	Danstin	edanstin@contoso.com	(205) 555 - 9882	Team Lead...	Development	2	Elizabeth	Lincoln	elincoln@contoso.com	(205) 555 - 9862	Senior Dev	Development	3	Margaret	Peacock	mpeacock@contoso.com	(205) 555 - 9852	Senior Dev	Development
	FirstName	LastName	Email	Phone	Position	Branch																																																																																														
	Andrew	Fuller	afuller@contoso.com	(205) 555 - 9898	CEO	TopManagement																																																																																														
	Jeremy	Boather	jboather@contoso.com	(205) 555 - 9888	President QA	QA																																																																																														
	Anne	Dodsworth	adodsworth@contoso.com	(205) 555 - 9887	VP QA	QA																																																																																														
	Alexander	Tuckings	atuckings@contoso.com	(205) 555 - 9886	Team Lead...	QA																																																																																														
	Brenda	Smith	bsmith@contoso.com	(205) 555 - 9885	Senior QA	QA																																																																																														
	Mary	Bird	mbird@contoso.com	(205) 555 - 9885	Team Lead...	QA																																																																																														
	Steven	Buchanan	sbuchanan@contoso.com	(205) 555 - 9897	President ...	Development																																																																																														
	Robert	King	rking@contoso.com	(205) 555 - 9896	VP Dev De...	Development																																																																																														
	Laura	Callahan	lcallahan@contoso.com	(205) 555 - 9892	Team Lead...	Development																																																																																														
0	Jason	Roland	jroland@contoso.com	(205) 555 - 9872	Senior Dev	Development																																																																																														
1	Eric	Danstin	edanstin@contoso.com	(205) 555 - 9882	Team Lead...	Development																																																																																														
2	Elizabeth	Lincoln	elincoln@contoso.com	(205) 555 - 9862	Senior Dev	Development																																																																																														
3	Margaret	Peacock	mpeacock@contoso.com	(205) 555 - 9852	Senior Dev	Development																																																																																														



*Table 7: Database Management System Comparison*

MongoDB has been chosen as my Database Management System because of my own personal preference, I would like to gain a bit of insight into NoSQL and to find out exactly why MongoDB is so popular.

## **3.4 Additional Software**

### **3.4.1 GitHub**

GitHub is a web-based software project repository hosting service. Github uses Git which is a distributed revision control system with an emphasis on speed, data integrity, and support for distributed, non-linear workflows. [21] GitHub provides a web-based graphical interface for Git based projects. Users can make either private or public repositories if they want to share their code or not.

GitHub will be used for the version control of the whole system and a private repository will be used until project completion.

### **3.4.2 Visual Studio**

Visual Studio 2013 is an IDE (Integrated Development Environment) by Microsoft used for the creation of Windows based applications. Visual Studio will be the IDE used for writing both the Server-Side application and the Client-Side application as there are Windows Azure SDKs available to be used with these applications to help in deployment.

### **3.4.3 RoboMongo**

RoboMongo gives a nice GUI (Graphical User Interface) to the MongoDB database collections as opposed to the otherwise text shell based commands. It is a

management tool for MongoDB databases which can be achieved from any location, provided correct user credentials are entered.

### **3.5 Conclusion**

The Software's which will be involved in the implementation of the whole system:

- Unity3D
- Kinect 2 (Kinect for Windows SDK v2.0)
- Windows Azure
- MongoDB
- GitHub
- 3DS Max
- Visual Studio 2013 (ASP.NET)

## **4 Design**

This chapter gives examples of methodologies which could be used by the system and compares them. It also describes the overall design of the system based on a Use-case Diagram, Entity Relation Diagram, Flowchart and User Interface Prototype.

This chapter covers:

**Methodologies**

**Use-Case**

**Entity Relation Diagram**

**System Architecture Diagram**

**Flowchart**

**User Interface Prototype**

**Conclusion**

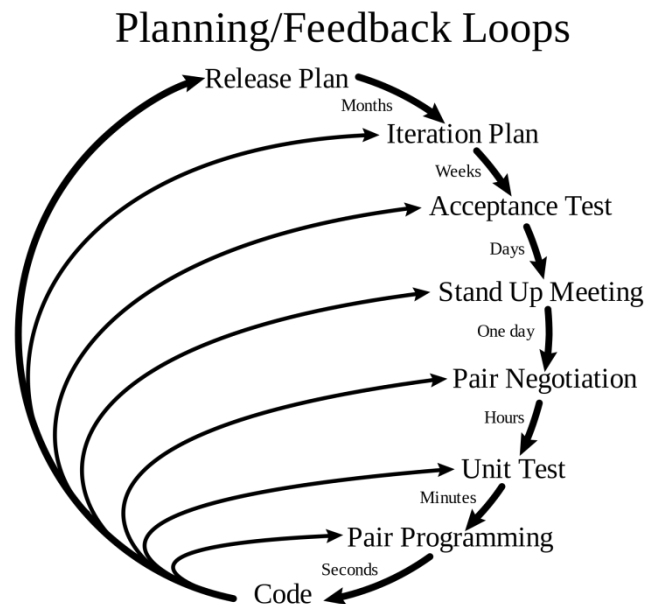
### **4.1 Methodologies**

Choosing the correct Software Development Methodology is an important factor in the design, implementation and development of the system. It makes the life of a developer working on a project a whole lot simpler by laying out a set of predefined steps to follow from the beginning to the end of the project.

For the Heart Health project, I am considering one of three software development methodologies of the Agile Software Development's Scrum method and Iterative and

Incremental method. Each methodology will be analysed and compared to one another to determine which is the most suitable for use in developing this project.

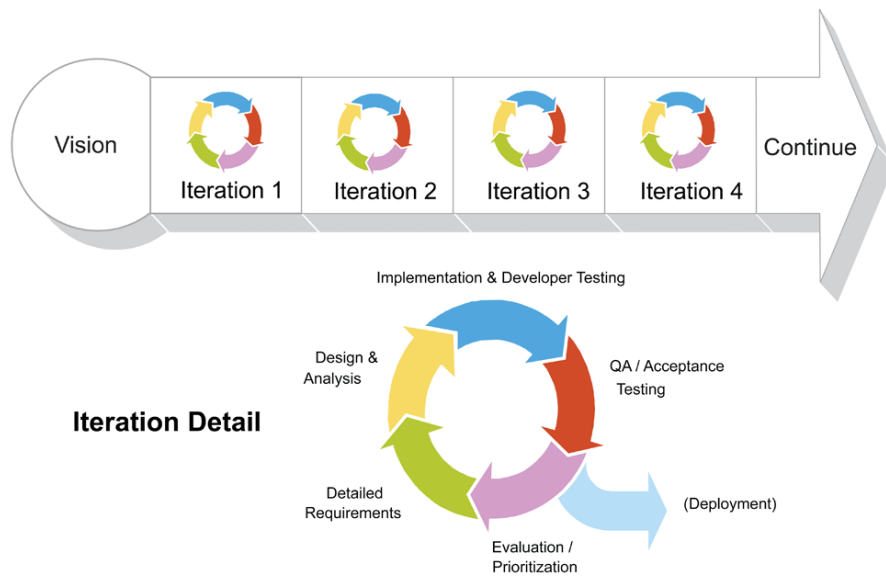
#### 4.1.1 Extreme Programming Methodology



*Figure 17: Extreme Programming Methodology*

The Extreme Programming Methodology (Figure 17) is part of the Agile software development Framework. It advises on frequent releases of the system in short development cycles. It focuses a lot on code reviews and continuously updated already deployed software. “They deliver the system to the customers as early as possible and implement changes as suggested.” [22]

### 4.1.2 Scrum (Agile Methodology)

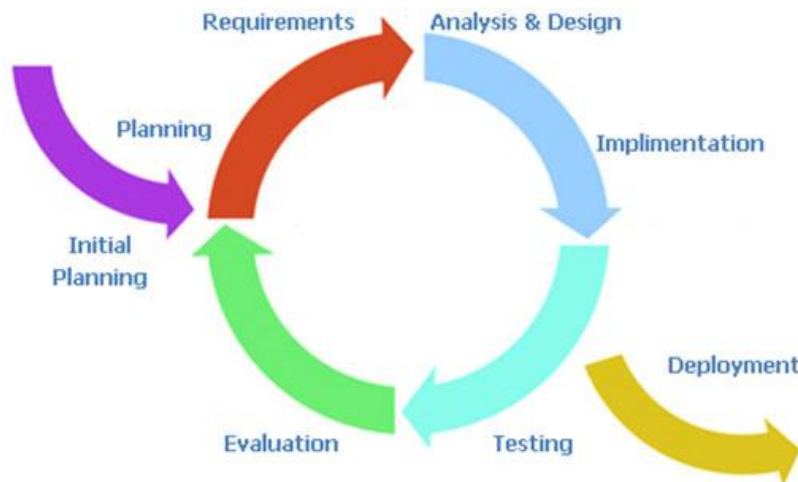


*Figure 18: Scrum Methodology*

Scrum is an Agile Software Development Framework (Figure 18) with Scrum roles for team members must be assigned to properly implement the methodology such as the Product Owner, the Development Team and the Scrum Master.

The reason as to why Scrum was selected was because it is usually recommended as the best software development methodology in terms of game development. Scrum also displayed logical methods in terms of development of a game with its many iterations of the developed product and continuously improved iterations of the product over time, as well daily meetings on what could be improved in the developed product.

### 4.1.3 Iterative and Incremental Method



*Figure 19: Iterative and Incremental Method*

The Iterative and Incremental Method (Figure 19) is very similar to how the waterfall method is implemented, except it is more refined and it uses multiple development cycles. Each part or section of the system is divided up (incrementally) and added piece by piece going through each of the development cycles until completion. This usually means that the core functionality of the system is implemented in the first build and bit by bit features are added onto the build through each development cycle. On each increment, new designs are drawn up and new tests plans are created for testing functionality through each build. [23]

### 4.1.4 Methodology Analysis

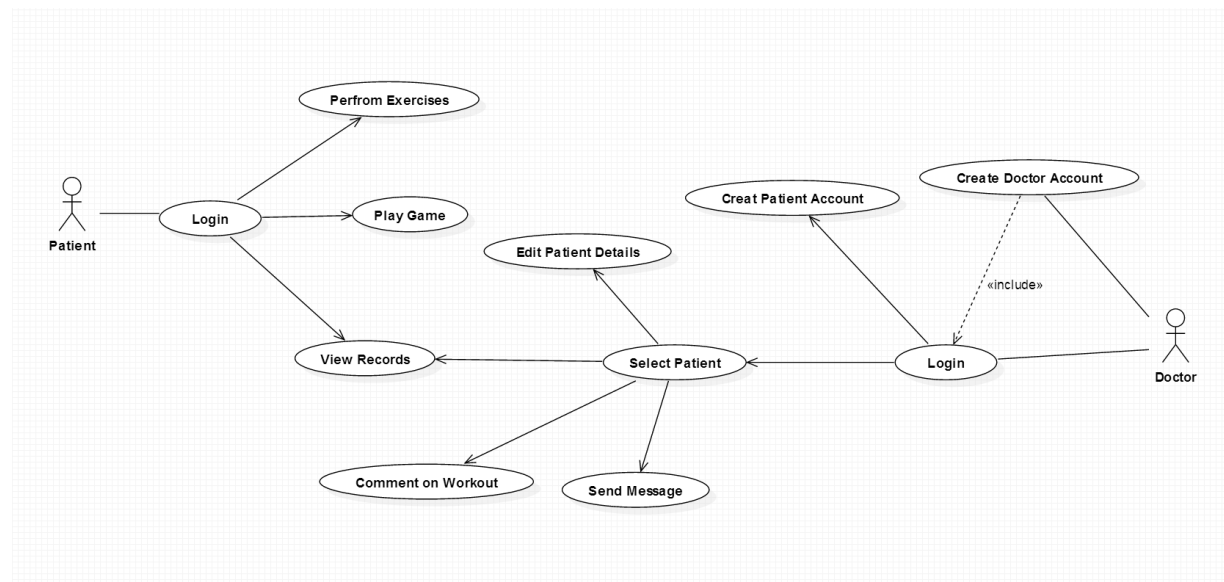
The Scrum Methodology is indeed a very good methodology in terms of game development with a team however the project is an individual project and Scrum mainly describes how a team work through a game development project.

In terms of the project, the whole project can be broken up into many different parts / components. There are already two whole systems (Doctor System, Patient System) to be implemented and these systems can be broken into even smaller parts / components. This is where Scrum and the Iterative and Incremental Method come into use while Extreme Programming does not; Extreme Programming releases deployed builds as opposed to increments of the whole system.

So in conclusion it was found that the best suited methodology for the project was the Iterative and Incremental Method, for many reasons, since the commencement of the project, a lot of what has already occurred before the Software Development methodology was chosen has already been played out in the way the Iterative and Incremental model specifies.

## 4.2 Use-Case

A use-case diagram is a simple representation of a user (users) interaction with the system. It specifies the components to be implementation in a system and is used as a basis for the design of the system.

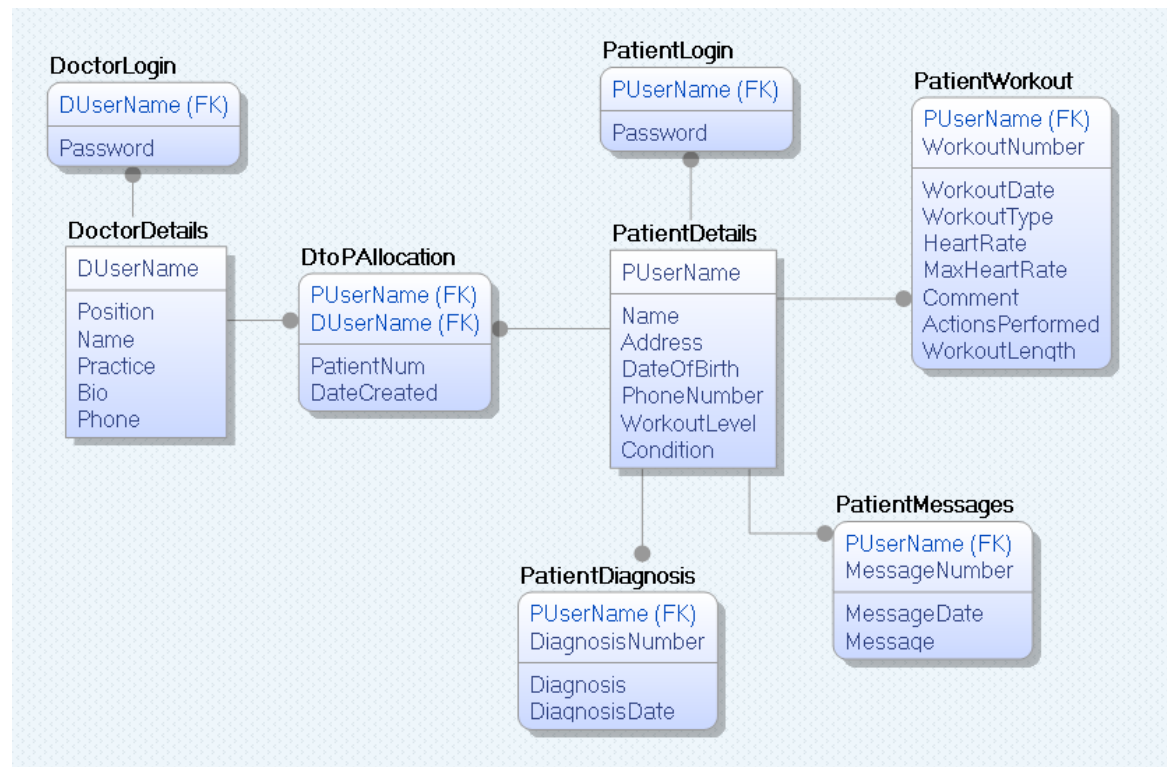


*Figure 20: Use-Case Diagram*

The use case diagram above (Figure 20) illustrates the initial use-case to be implemented in the system. The requirements were gathered through research and how the system will be logically implemented. Based on the methodology chosen, the use-case requirements were subject to change. (Use-cases have not been altered).

### 4.3 Entity Relation Diagram

The Entity Relation Diagram or ERD is a model of mapping relational data sets together and to determine how each data set interacts with one another by means of querying. It is used to as a design to how the database's data set are to be laid out.



*Figure 21: Entity Relation Diagram*

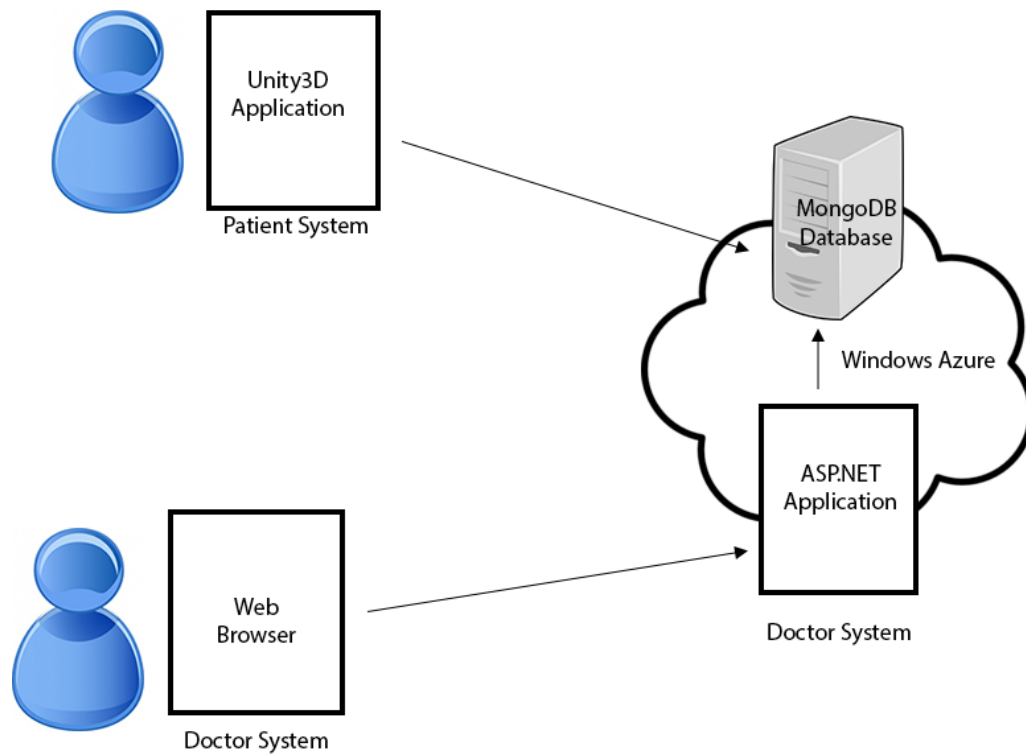
Figure 21 illustrates the ERD (Entity Relation Diagram) for the system. Based on the methodology the ERD may be subject to change.

It is also to note that the Database Management System chosen of MongoDB does not use Entity's and relationships as presented in this ERD, however an ERD is simply a mapping of the data that is intended to store and the relations amongst that data. So this ERD may be used and does apply to the MongoDB database that is to be implemented.



## 4.4 System Architecture Diagram

The system architecture diagram is a representation of the system architecture. The applications involved in the running of the entire system.



*Figure 22: System Architecture Diagram*

Figure 22 illustrate the system architecture for the system with Users connecting to the system by means of being in the role of a doctor or a patient. The patient's role only has access to the Patient system which is a Unity3D application while the doctor's role only having access to the Doctor system which is an ASP.NET application and is accessed by means of a web browser. The ASP.NET application manages requests to-and-from the server with anything the involving the doctors application. The server-side application retrieves data from the MongoDB database. The patients system is connected directly to the MongoDB database in terms of managing requests for data and storing data.

## 4.5 Flowchart

A flowchart is a diagram which can represent an algorithm, workflow, or process. Each step within the flowchart is presented in a box and the boxes are connected by arrows.

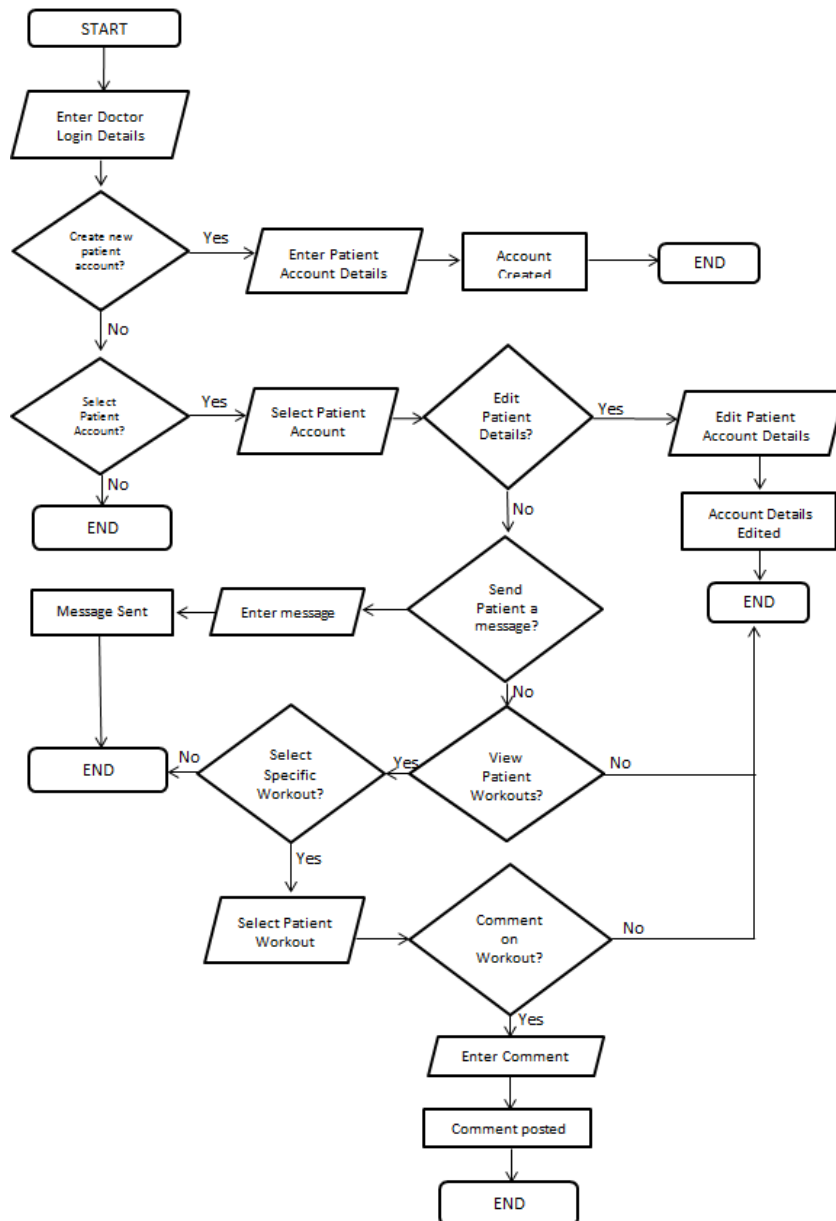


Figure 23: Flowchart

Figure 23 illustrates a flowchart of the processes in the doctors system and displays how certain areas within the system are reached and what must be done to reach those areas. This flowchart is just an example of how the doctor's system will work and will not be fully implemented as such.

## 4.6 User Interface Prototype

The user interface prototype is a hand drawn image of the perceived outlook of how the system user interface will appear on completion of the actual system.

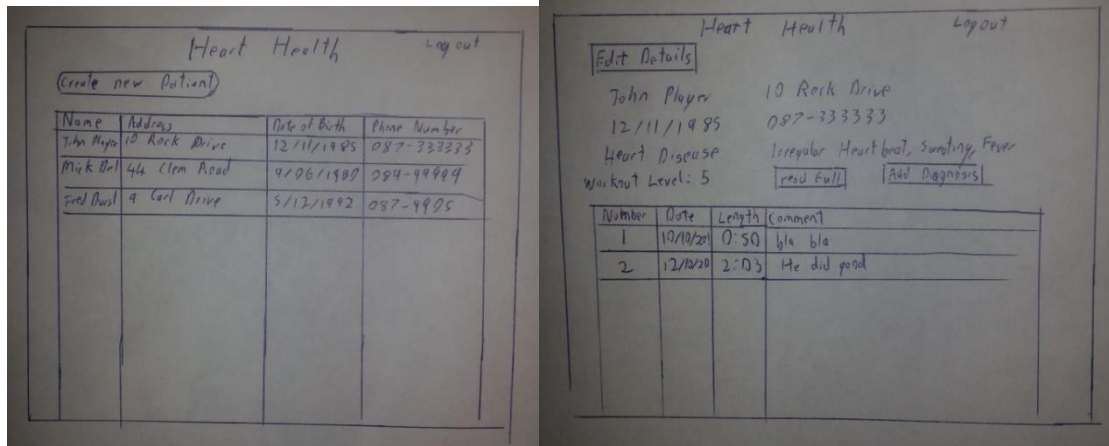


Figure 24: PatientView Prototype

Figure 25: PatientData Screen

User Interface Prototypes were drawn up of two proposed screens from the Doctor's system. The PatientView Screen (Figure 24) details all the patient user accounts created by the doctor, clicking on a specific account will bring the user (doctor) to the PatientData Screen. The doctor can also choose to create another patient account. The PatientData Screen (Figure 25) displays the details of the previously selected patient. The patient's workouts (if any are already performed) are displayed, the doctor has the ability to click into a workout and examine it more clearly.

## 4.7 Conclusion

This chapter has covered the overall design which was used in building the system with what methodology was chosen, the Use-case diagram which was used to display the requirements of the system, the Entity Relation Diagram which was used for the database tables (collections) incorporated into the system, the flowchart which represented a process within the system as well as User Interface Prototypes for the design of specific screens within the system.

## 5 Implementation

This chapter covers the overall implementation of the system and how each component of the system was developed. Implementation of the system was based on each of the design elements and research stated in the previous chapters.

The sections within this chapter:

**System Overview:** The system overview section gives a summary of the two completed systems.

**System Components Implementation:** The system components implementation gives a detailed account of implementation of the overall system.

**Problems Encountered:** The problems encountered section details the problems which were found in each system.

**Conclusion:** Summary of what was done within this chapter.

### 5.1 System Overview

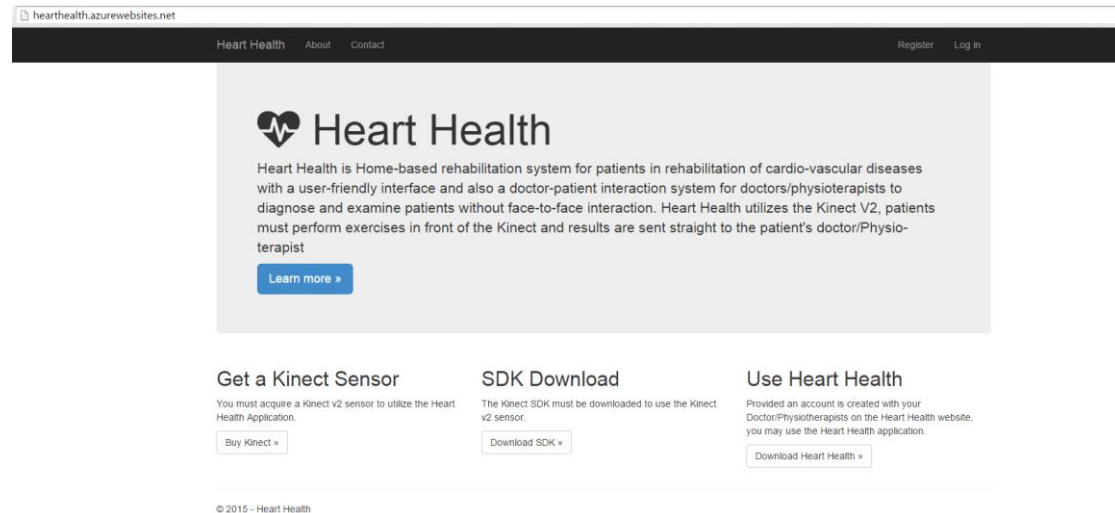
The overall Heart Health system comprises of two systems, the Heart Health Website and the Heart Health Application. Both systems have specific functions to the overall project. The Heart Health Website being the Doctor/Physiotherapists website for creating, viewing and editing patient records and the Heart Health Application which is the application to be used by patient's for their home-based rehabilitation.

#### 5.1.1 Heart Health Website

The Heart Health Website was created using ASP.NET in Visual Studio 2013. It is also hosted on Windows Azure (<http://hearthealth.azurewebsites.net/>). The website is connected to a MongoLab database which is also hosted on Windows Azure.

The website's main function is the Doctor/Physiotherapist administration of their patient's. Doctors/Physiotherapists can register an account on the website, and once their account is created, they have the ability to create Patient Accounts. The purpose of these Patient Accounts is to be used by the Patient of the Doctor/Physiotherapist for their rehabilitation. The Patient uses the Account to log into the Heart Health Application with this account. Each Patient Account created can be viewed, edited and work out details respective of each patient can be viewed. Messages can be also

be sent directly to the patient, (which can be viewed within the application) and a diagnosis of the patient can be made to keep updates on the patients progress which can only be viewed by the doctor. The Doctor/Physiotherapist can also comment on specific workouts to provide feedback to the patient.

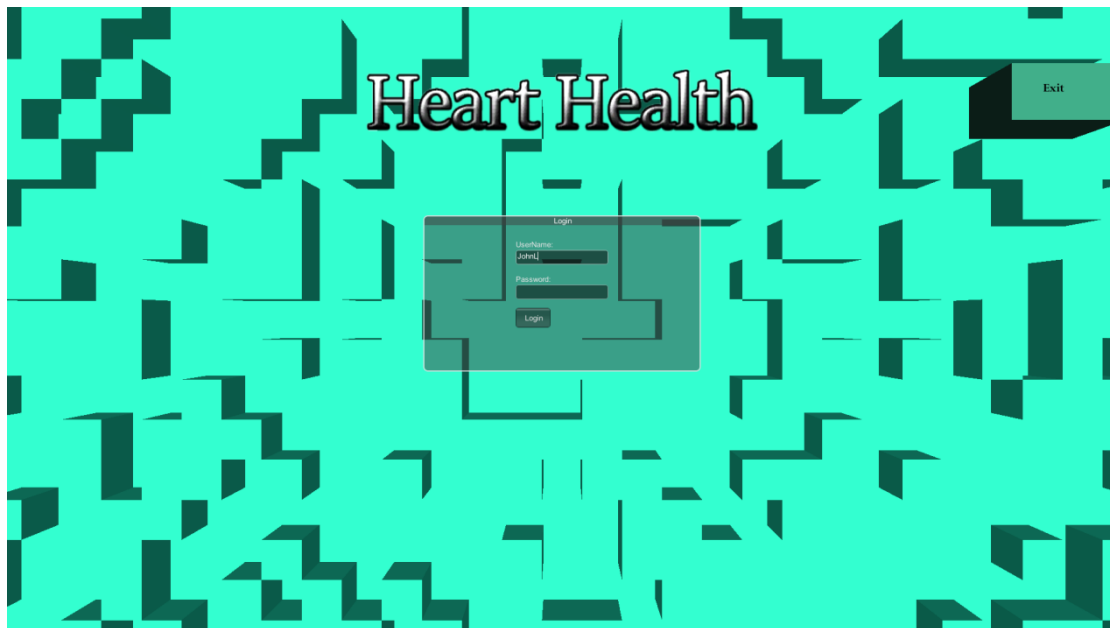


*Figure 26: Heart Health Website Home Page*

### 5.1.2 Heart Health Kinect

The Heart Health Kinect (Application) was created using Unity3D and each class was written in C#. The application itself (as an executable) can be downloaded from the Heart Health Website (<http://hearthealth.azurewebsites.net/>). The application is also connected to the same MongoLab database which is hosted on Windows Azure.

The application's main function is the Patient's rehabilitation. The patient can log into the application with their username created by the doctor on the Heart Health Website (there must be an internet connection to do this), from there they can play any of the four exercise games available, of a Basic Workout, Simon Says, Heart Racer or an Orb Dodger type game. The Patient can also view their specific workouts records, view the Doctor's information, as well as view their own information and edit it. Workout information is recorded from the Basic Workout, Simon Says and Heart Racer games, the Orb dodger game is treated as a warmup type game.



*Figure 27: Heart Health Kinect Login Page*

## **5.2 System Components Implementation**

Following the methodology chosen of Iterative and Incremental Method, the entire system was split into components, the obvious two systems of the Patients system and the Doctors system are seen as components of the entire project however part of these systems are split into sub-components. The three main components of the system are the Database Connection classes, the Website classes (Heart Health Website), the Application classes (Heart Health Kinect).

### **5.2.1 Database Connection Implementation**

The Database Connection classes comprise of the classes used for the database connection to the online MongoLab database, hosted on Windows Azure. Both the Heart Health Application and Website use these classes for database connection.

## Database Classes Setup

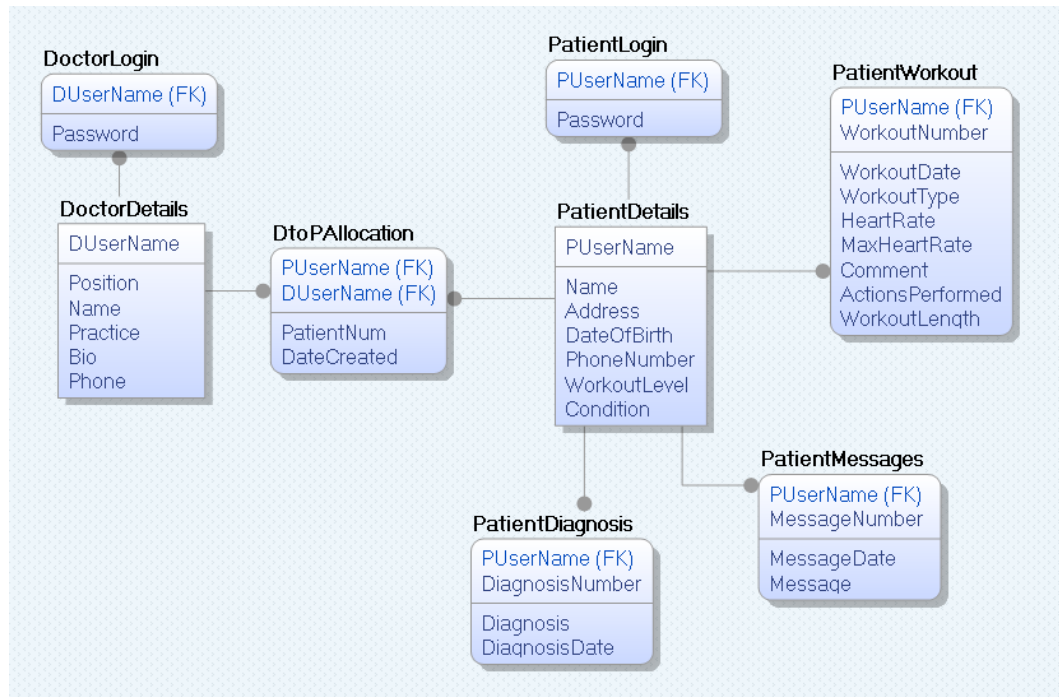


Figure 28: ERD

Each of the database classes created are based off the ERD (Entity Relation Diagram) above, however it should be noted that ERDs are mainly used to represent SQL table relations as MongoDB is a NoSQL type system. However an ERD can be still used to represent the database classes used in MongoDB, MongoDB saves data in a JSON type format (called BSON with MongoDB) which is represented as such:

```

{
  "_id" : ObjectId("55061623ee97182214c35f00"),
  "PUserName" : "SarahF",
  "Name" : "Sarah Farron",
  "Email" : "farron@gmail.com",
  "Address" : "12 Sunnybank Avenue, Dundrum",
  "DateOfBirth" : ISODate("1981-09-25T21:00:00.000Z"),
  "PhoneNumber" : "00353878128789",
  "FitnessLevel" : 3,
  "Condition" : "Ischemic heart disease",
  "WeightKG" : 71,
  "IsMale" : false,
  "Workouts" : [],
  "Diagnosis" : [],
  "Messages" : []
}
  
```

Code 1: PatientDetails BSON

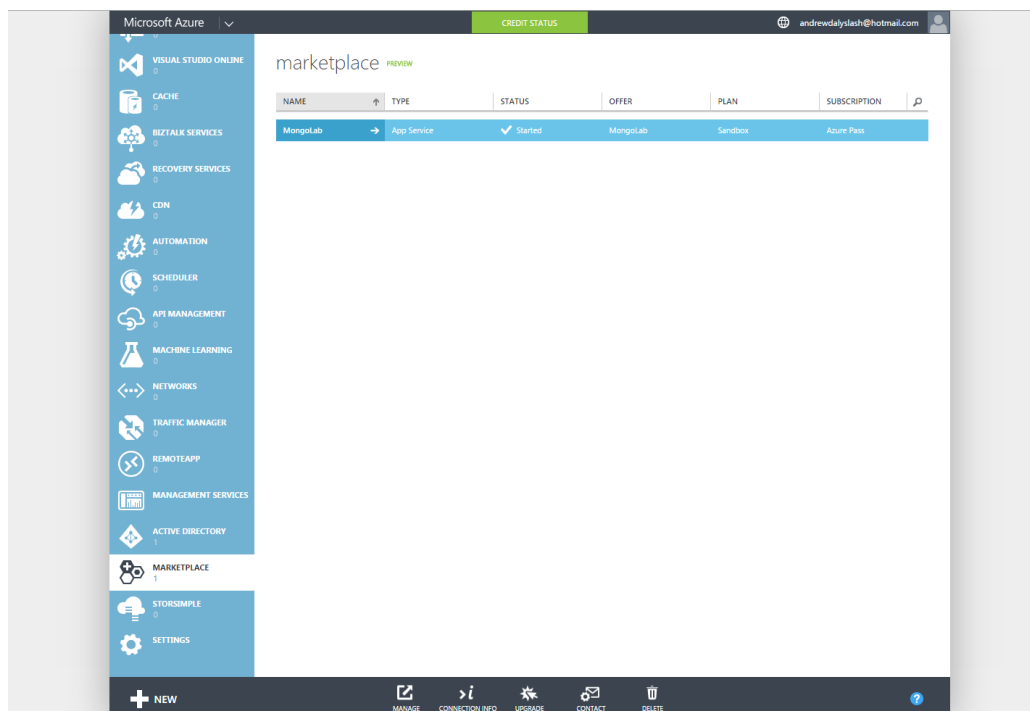
The JSON data display above (PatientDetails BSON), relates to data within the PatientDetails class, which holds the patient's information as well as their Workout information as a list, their Diagnosis information and messages sent to the patient. The class in C# is as follows:

```
public class PatientDetails
{
    [BsonId]
    public ObjectId Id { get; set; }
    public string PUserName { get; set; }
    public string Name { get; set; }
    public string Email { get; set; }
    public string Address { get; set; }
    public DateTime DateOfBirth { get; set; }
    public string PhoneNumber { get; set; }
    public int FitnessLevel { get; set; }
    public string Condition { get; set; }
    public int WeightKG { get; set; }
    public bool IsMale { get; set; }
    public IList<PatientWorkout> Workouts { get; set; }
    public IList<PatientDiagnosis> Diagnosis { get; set; }
    public IList<PatientMessages> Messages { get; set; }
}
```

*Code 2: PatientDetails Class*

## Windows Azure Database Setup

On testing the storing information to the database, localhost can be used to store data locally on the Computer, however a MongoLab database was setup on Windows Azure.



*Figure 29: Windows Azure MongoLab*



Tables (or Collections) and data can be saved onto this MongoLab database, provided there is an internet connection and the connection information is used to connect to the database.

### Database Connection Class

A project was set up to be used as a DLL for database connection called MongoConnectApp, the classes included in this project are in relation to each of the tables listed in the ERD. More classes were created to work with logging into the Heart Health Website such as the MongoMembershipProvider class.

The connection class to connect to the database is as follows:

```
public class MongoHelper<T> where T : class
{
    public MongoCollection<T> Collection { get; private set; }

    public MongoHelper(string connectionString)
    {
        MongoUrl mongoUrl = new MongoUrl(connectionString);
        MongoClient client = new MongoClient(mongoUrl);
        MongoServer server = client.GetServer();
        server.Connect();
        MongoDBDatabase provider = server.GetDatabase(mongoUrl.DatabaseName,
        WriteConcern.Acknowledged);
        Collection = provider.GetCollection<T>(typeof(T).Name.ToLower());
    }
}
```

#### *Code 3: MongoHelper Class*

This simple class set ups the database connection as well as retrieves the collection stated and works as such:

```
MongoHelper<PatientDetails> patientdetails = new MongoHelper<PatientDetails>(ConnectionString);
```

#### *Code 4: MongoHelper Variable*

### Database Management Functions

Using this patientdetails class, a new Patient can be created as such:

```
public void Create(PatientDetails details)
{
    patientdetails.Collection.Save(details);
}
```

#### *Code 5: PatientDetails Collection Creation*

Data received from the database, works as such:

```
var details = patientdetails.Collection.Find(Query.EQ("PUserName",
username)).SetFields(Fields.Exclude("Workouts", "Diagnosis", "Messages")).Single();
```

*Code 6: PatientDetails Query*

This same method was done for the rest of the database classes connecting to the MongoLab database.

These classes are applied to both the Heart Health Website as well as the Heart Health Application, however on they are added as just classes in the Heart Health Application project as there was an issue with using the MongoConnectApp dll in the Unity project. The MongoConnectApp dll is used within the Heart Health Website project however.

### **Secure Login Class**

Another class created within the MongoConnectApp as mentioned before is the MongoMembershipProvider (SimpleMongoProvider) which manages the login in the Heart Health Website. It does this by using the WebMatrix.WebData and System.Web.Security DLL classes. It also uses functions which extend from the ExtendedMembershipProvider class (WebMatrix.WebData).

Within the Web.Config file, located within the Heart Health Website project, the MongoMembershipProvider must be stated to ensure that logins and any other account handling functions (Account Creation, Change Password) are used within the website:

```
<add name="MongoMembershipProvider" type="MongoConnectApp.MongoMembershipProvider,
MongoConnectApp" connectionStringName="mongodb" useAppHarbor="true" />
```

*Code 7: Membership Provider String*

Logins are handled within the AccountController class within the website and then properly analysed within the MongoMemberProvider as such:

```
string hashedPassword = GetHashedPassword(username);
bool vertificationSuccess = (hashedPassword != null &&
Crypto.VerifyHashedPassword(hashedPassword, password));
var account = accountlogin.GetLogin(username);
if (account != null)
{
}
}
```

*Code 8: Secure Login Code*

The accountlogin class stated inside the code above is the database class which connects to the MongoDB database:

```

        public DoctorLogin GetLogin(string username)
        {
            if (CheckUserName(username))
            {
                var details = doctorlogin.Collection.Find(Query.EQ("DUserName",
username)).Single();
                return details;
            }
            else
            {
                return null;
            }
        }
    }

```

*Code 9: Login by Querying Database*

### Database Class Changes

A few changes were made to the tables (collections) stated in the ERD within the Doctor Login table more fields were added to comply with the ExtendedMembershipProvider functions:

```

public class DoctorLogin
{
    [BsonId]
    public ObjectId Id { get; set; }
    public string DUserName { get; set; }
    public string Password { get; set; }
    public DateTime CreateDate { get; set; }
    public bool IsConfirmed { get; set; }
    public string ConfirmationToken { get; set; }
    public DateTime LastPasswordFailureDate { get; set; }
    public int PasswordFailuresSinceLastSuccess { get; set; }
    public DateTime PasswordChangedDate { get; set; }
    public string PasswordVerificationToken { get; set; }
    public DateTime PasswordVerificationTokenExpirationDate { get; set; }
    public DateTime LastLoginDate { get; set; }
}

```

*Code 10: DoctorLogin Class*

As well as that changes were made to the PatientDetails class such as including a field for the patients gender, their weight and their fitness level, the reason for this is explained within Application Implementation sub-chapter.

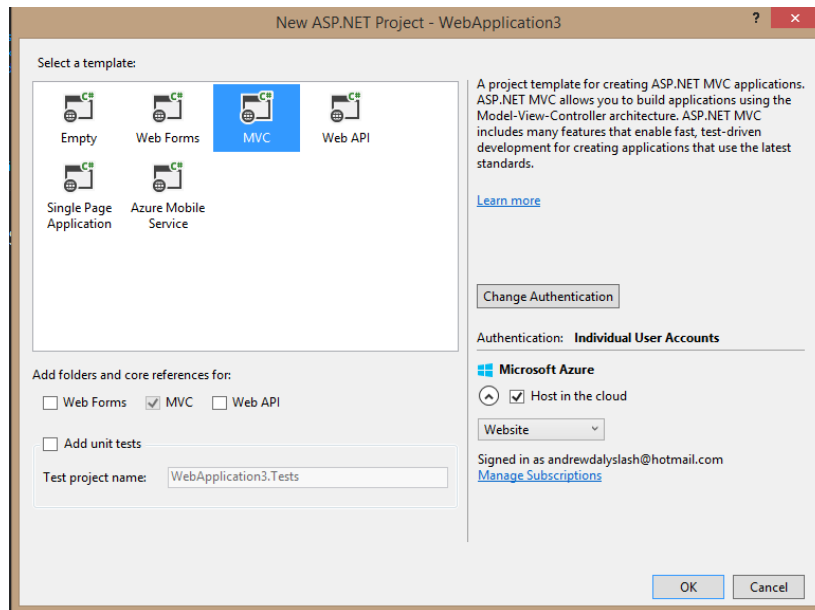
### 5.2.2 Website Implementation

During the start of the Website Implementation it was planned to build the website using Python with Django in Visual Studio. However since the Database Connection classes were already created in C#, difficulties arise in using the C# created DLLs in Python. It is indeed possible to use DLLs within Python however by using either Iron

Python or ctypes, however difficulty arose in doing. It was decided that ASP.NET would be the best option in building the website as ASP.NET with the new Razor Engine fully supports C# and the Database Connection DLLs created.

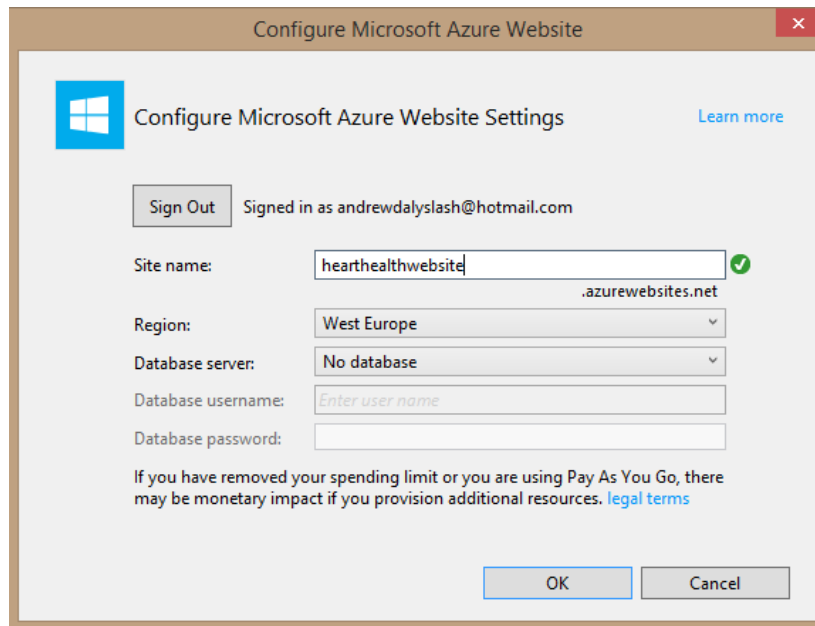
## ASP.NET Project Setup

Visual Studio 2013 comes with many different ASP.NET project templates:



*Figure 30: ASP.NET Project Creation*

It was decided that MVC 4 would be used as the project template. Beforehand the Windows Azure SDK was downloaded and installed into Visual Studio 2013 which provided handy features in quick deployment of the website to the cloud such as immediate setup of the site name, which allowed the website created to be published immediately.



Configure Microsoft Azure Website

Sign Out Signed in as andrewdalyslash@hotmail.com

Site name:  ✓  
 .azurewebsites.net

Region:

Database server:

Database username:

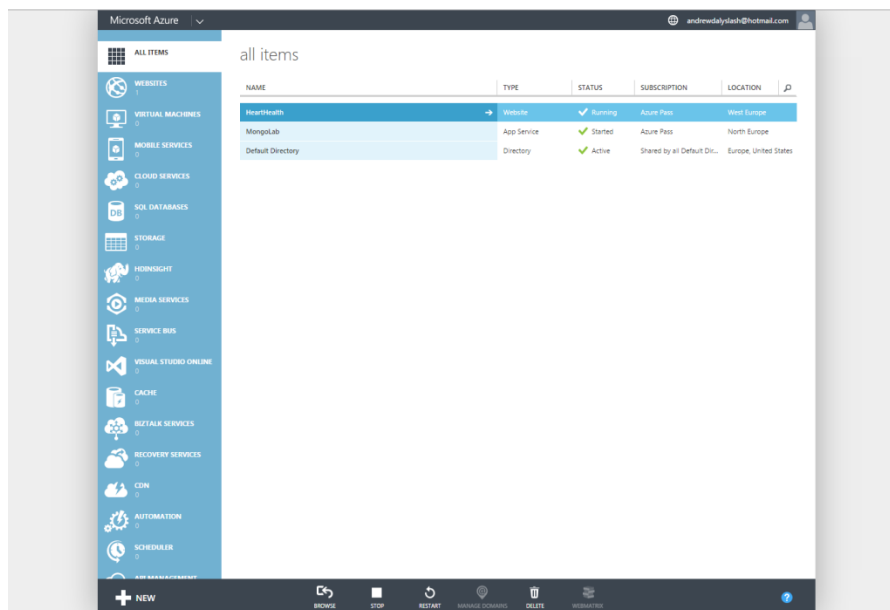
Database password:

If you have removed your spending limit or you are using Pay As You Go, there may be monetary impact if you provision additional resources. [legal terms](#)

OK Cancel

*Figure 31: Windows Azure Site Creation*

On the project creation the website was created along with it, it could also be published immediately with the click of a button (Figure 16) within the solution:



*Figure 32: Windows Azure All Items*

## Doctor Login Management Class Created

The database connection DLL was added to the project and the AccountController class was created for Account Registration, and Account Logins. Within the AccountController, each page within the website referred to as a View requires its own Controller function, for the AccountController class, all views which related to Logins were included. All the Account Controller views created (Login, Register, Manage, AccountDetails, EditDetails) related to the both the DoctorLogin and DoctorDetails database classes. Views are .cshtml files, which are specific only to the Razor View Engine. .cshtml files allow for C# code to be run from within views:

```
@using MongoConnectApp.HeartHealthConnect
@model PatientDetails

@{
    if (Model.IsMale)
    {
        ViewBag.Title = "Mr " + @Model.Name;
    }
    else
    {
        ViewBag.Title = "Ms " + @Model.Name;
    }
}
```

*Code 11: Sample cshtml code*

## Patient Management Class Created

C# code within these files are done starting with “@”, the rest is html.

Another controller was made to manage the patient database classes of PatientController. PatientController contained views which allowed for the creation of patient accounts to be used within the heart health application (CreatePatient, DiagnosisListing, EditPatient, MessageListing, PatientData, PatientView, WorkoutData).

The hand-drawn prototype shows a header with 'Heart Health' and a 'Log out' link. Below the header is a button labeled '(Create new Patient)'. The main content is a table with four columns: Name, Address, Date of Birth, and Phone Number. The table contains three rows of patient data.

Name	Address	Date of Birth	Phone Number
John Mayo	10 Rock Drive	12/11/1985	087-333333
Mick Del	44 Clem Road	9/06/1980	084-99999
Fred Durs	9 Carl Drive	5/12/1992	087-9905

*Figure 33: PatientView Drawn Prototype*

The PatientsView screen (Figure 34) was based off the User Interface prototype (Figure 33) created during Design phase of the project. The PatientView screen gives an overview of all the patient accounts created by the overall account owner (Doctor/Physioterapist). It is also to be noted that Bootstrap.css was used for the design which came with the MVC template.

The screenshot shows the 'Patient Overview' section of the 'Heart Health' application. It includes a 'Create Patient' button and a table listing patient details. The table has columns for Username, Name, Email, Address, and PhoneNumber. Each row has a 'View' button next to the Username column.

Username	Name	Email	Address	PhoneNumber
<a href="#">View</a> SarahF	Sarah Farron	farron@gmail.com	12 Sunnybank Avenue, Dundru	00353876128789
<a href="#">View</a> BartGuz	Bartek Guzowski	guzo1900@gmail.com	12 Dundalk	08777655464
<a href="#">View</a> Bryan	Bryan Duggan	bryand@gmail.com	79 Happy Lane	0449378904
<a href="#">View</a> JohnL	John Lucey	johnlucey@gmail.com	10 Brookfield Mullingar	00353875090901
<a href="#">View</a> JackyRamone	Jack Creagh	jackeyramones@gmail.com	7 David Road, Drumcondra, D7	+353871849090

© 2015 - Heart Health

*Figure 34: PatientView Screen*

Number	Date	Length	Comment
1	10/10/20	0:50	bla bla
2	12/10/20	2:03	He did good

Figure 35: PatientData Drawn Prototype

The PatientData screen (Figure 36) is also based off another User Interface prototype (Figure 35) created during Design phase of the project. The PatientData screen gives an overview of the patient's information, which can also be edited by the Doctor. It also gives an overview of all the workouts performed by the patient. The workout information can only be received from the Heart Health Application.

Mr John Lucey

John L.  
10 Brookfield Mullingar  
johnlucey@gmail.com  
00353879090901

10/12/1992 Age: 22  
Weight: 65 KG  
Fitness Level: 1  
Ischemic heart disease

Number	Date	Length	Comment
1	3/18/2015 6:01:09 PM	19	Great Workout!
2	3/19/2015 12:10:16 PM	108	Your doing great!
3	3/19/2015 12:11:11 PM	43	Keep that heart rate up!
4	3/19/2015 12:30:22 PM	59	
5	3/20/2015 4:57:13 PM	149	Musta been a fun game eh?
6	3/22/2015 2:28:48 PM	16	hey
7	3/24/2015 12:43:56 AM	155	
8	3/24/2015 11:30:39 AM	19	
9	3/24/2015 11:31:26 AM	25	
10	3/24/2015 11:32:06 AM	20	
11	3/24/2015 12:20:50 PM	15	

Figure 36: PatientData Screen

From the PatientData screen (above), Doctor/Physiotherapists also have the ability to send messages (PatientMessages database class) directly to the patient to be viewed from within the Heart Health Application. A patient diagnosis can also be made, only viewable by the Doctor/Physiotherapist (PatientMessages database class).

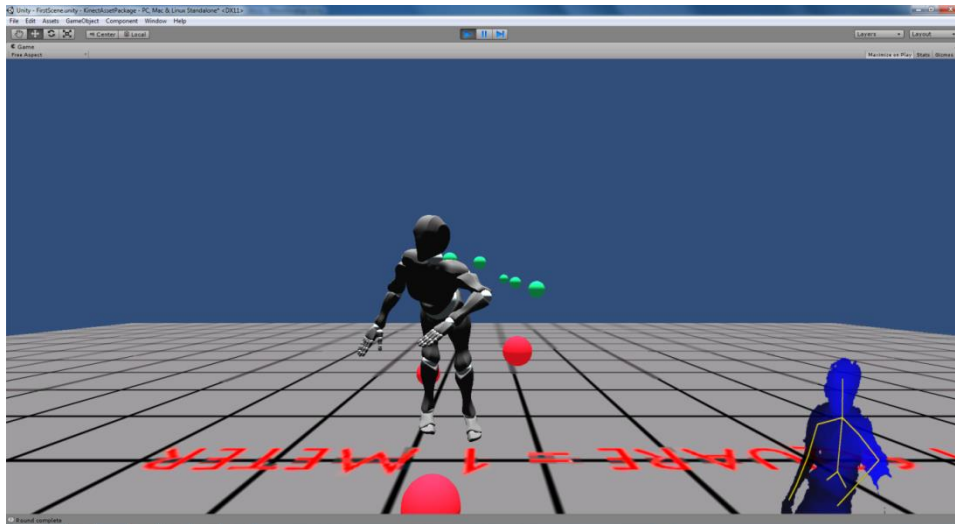


### 5.2.3 Application Implementation

The entire Heart Health Application (Patients System) was built in Unity. The Kinect V2 does not always need to be connected for the system to work; it is only needed for the gameplay elements of the system. An internet connection however is required for the Heart Health Application to work properly.

### Prototype Development

Before starting on the Application, a prototype was created using the Kinect v1 sensor, to test the motion capture capabilities of both sensors (Kinect v2).



*Figure 37: Kinect v1 Prototype*

For the Prototype, Kinect wrapper files available on the Unity asset store were downloaded, (<https://www.assetstore.unity3d.com/en/#!/content/7747>) and used to allow Kinect v1 compatibility with Unity. Models were downloaded from the site mixamo (<https://www.mixamo.com/3d-characters>).

### Kinect v2 Wrapper Files

As for the project itself, the same Kinect wrapper except for the Kinect v2 was downloaded (<https://www.assetstore.unity3d.com/en/#!/content/18708>), during the beginning it was decided that personal wrapper files would be created however due to time constraints as well as many differences in Kinect v2 initialisation, it was decided these will be used.

## Workout Screen

The first scene which was developed was a basic workout scene; it was initially thought to include an avatar to represent the player within the scene, however there wasn't actually a need for one as the scene just required the player to perform workouts in view of the Kinect.

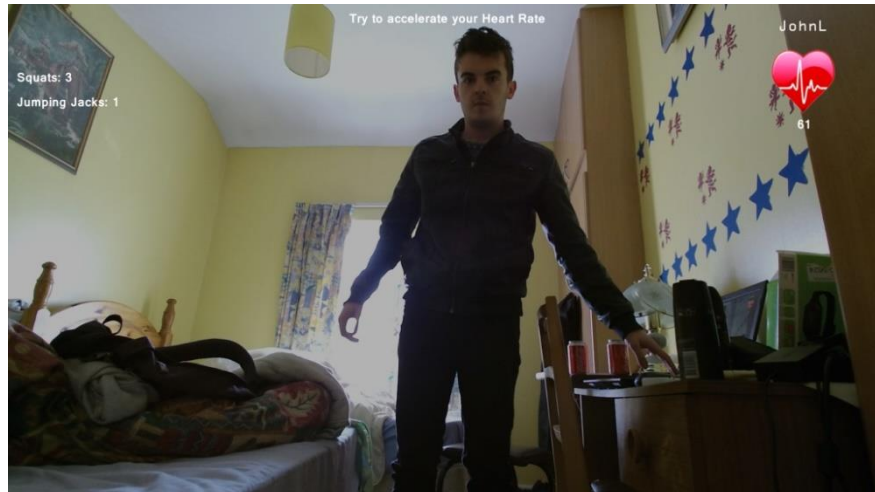


Figure 38: Basic Workout Screen

The scene was set up in Unity with the default Main Camera, as well as a Plane, with which the User Image is drawn onto. On the Main Camera the Kinect Manager Script is attached, which must be attached to all Main Camera where the Kinect is to be used:

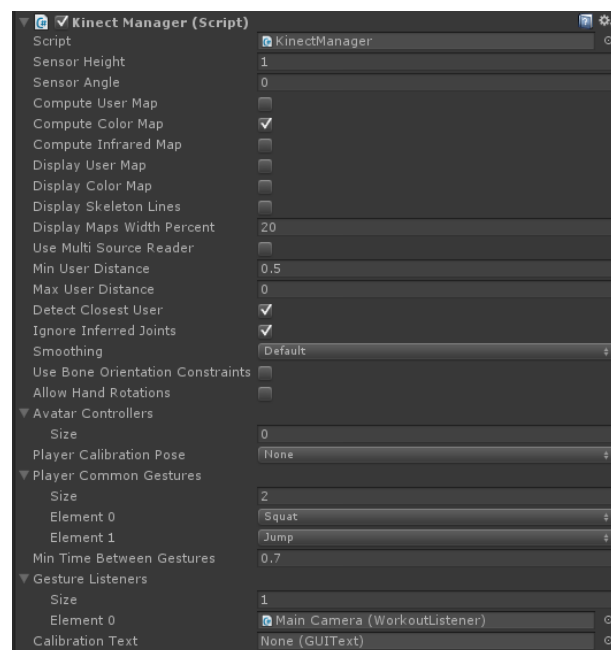


Figure 39: KinectManager Attached Script

## User Image

To project the User Image onto the Plane within the scene a new script was created, called ColourView. A ready-made class was made within the Kinect Manager to show the User map. The ColourView script was also attached to the camera.

```
KinectManager manager = KinectManager.Instance;

if(manager && manager.IsInitialized())
{
    if(ImageWall)
    {
        Texture2D UsersClearTex = manager.GetUsersClrTex();
        ImageWall.renderer.material.mainTexture = UsersClearTex;
        float pos = (Camera.main.nearClipPlane + 0.01f);

        ImageWall.transform.position = Camera.main.transform.position + Camera.main.transform.forward
        * pos;
        ImageWall.transform.LookAt (Camera.main.transform);
        ImageWall.transform.Rotate (270.0f, 180.0f, 0.0f);
        float h = (Mathf.Tan(Camera.main.fieldOfView*Mathf.Deg2Rad*0.5f)*pos*2f) /10.0f;
        ImageWall.transform.localScale = new Vector3(h*Camera.main.aspect,1.0f, h);
    }
}
```

*Code 12: ColourView Code*

## Workout Gesture Listener

Another class was to be created to detect workouts; the main two workouts which are to be used within the project are Squats and Jumping Jacks. The class created WorkoutListener inherited from the KinectGestures.GestureListenerInterface class. The KinectGestures.GestureListenerInterface contained the list of gestures which could be detected by the Kinect.

```
if (gesture == KinectGestures.Gestures.Squat) {
    SquatNum++;
    Camera.main.transform.SendMessage ("SquatPerformed");
}
else if (gesture == KinectGestures.Gestures.Jump) {
    JumpNum++;
    Camera.main.transform.SendMessage ("JumpPerformed");
}
```

*Code 13: Gesture Detection Code*

The line of code of `Camera.main.transform.SendMessage("SquatPerformed")`, sent a message to the Camera which also contained the `HeartRateEstimation` script, to run a function within that script. (If no function called `SquatPerformed` was found, nothing would be done).

### **Heart Rate Estimation**

It was intended to incorporate the Kinect v2 capability at heart rate monitoring into the project, this was to be done by using the Kinects v2 Infrared sensor to detect heat signatures from the users face and as such that data could be used to assume heart rate. However it could not get implemented and as such an alternative had to be made, and as such the `HeartRateEstimation` class was made to estimate the current patient's heart rate based off factors such as the Patients Age, Weight in Kilograms (KG), their Gender, and their Fitness Level, (Athletic, Average, Poor). These fields were added to the `PatientDetails` database class and were taken on the Patient Account created.

For the heart rate estimation, statistics needed to be taken such as finding the average Resting Heart Rate, Target Heart Rate and Maximum Heart Rate for people of Age, Sex, weight, and fitness level. (2.3.1)

## Resting Heart Rate

<i>Resting Heart Rate for MEN</i>							
Age	18-25	26-35	36-45	46-55	56-65	65+	
Athlete	49-55	49-54	50-56	50-57	51-56	50-55	
Excellent	56-61	55-61	57-62	58-63	57-61	56-61	
Good	62-65	62-65	63-66	64-67	62-67	62-65	
Above Average	66-69	66-70	67-70	68-71	68-71	66-69	
Average	70-73	71-74	71-75	72-76	72-75	70-73	
Below Average	74-81	75-81	76-82	77-83	76-81	74-79	
Poor	82+	82+	83+	84+	82+	80+	
<i>Resting Heart Rate for WOMEN</i>							
Age	18-25	26-35	36-45	46-55	56-65	65+	
Athlete	54-60	54-59	54-59	54-60	54-59	54-59	
Excellent	61-65	60-64	60-64	61-65	60-64	60-64	
Good	66-69	65-68	65-69	66-69	65-68	65-68	
Above Average	70-73	69-72	70-73	70-73	69-73	69-72	
Average	74-78	73-76	74-78	74-77	74-77	73-76	
Below Average	79-84	77-82	79-84	78-83	78-83	77-84	
Poor	85+	83+	85+	84+	84+	84+	

Figure 40: Resting Heart Rates[24]

The following resting heartrates provided a close estimation as to a user's resting heart rate based on their age, sex and fitness level and as such a function was made within the HeartRateEstimation class:

```
private int GetRestingHeartRate (int Age, int FitnessLevel, bool IsMale)
{
    int RestingHeartRate = 0;

    if (IsMale)
        RestingHeartRate += 5;
    else
        RestingHeartRate += 8;

    if (FitnessLevel == 1)
        RestingHeartRate += 48;
    else if (FitnessLevel == 2)
        RestingHeartRate += 59;
    else if (FitnessLevel == 3)
        RestingHeartRate += 69;

    if (Age < 25)
        RestingHeartRate += 4;
    else if ((Age > 25)&&(Age <= 35))
        RestingHeartRate += 5;
    else if ((Age > 35)&&(Age <= 45))
        RestingHeartRate += 5;
    else if ((Age > 45)&&(Age <= 55))
        RestingHeartRate += 7;
    else if ((Age > 55)&&(Age <= 65))
        RestingHeartRate += 6;
}
```

```

        else if (Age > 65)
            RestingHeartRate += 7;

    return RestingHeartRate;
}

```

*Code 14: GetRestingHeartRate Function*

The results of Figure provided a rough estimation for how heart rates were estimated in the GetRestingHeartRate function.

## Maximum Heart Rate

In finding an estimation of a patient's Maximum Heart rate (MHR), there are three different formulas available to measure Max heart rate based on different measurements. These formulas consist of the Fetal Heart Rate method which is the most basic, Karovonen Method and the Heil Method.

The Fetal Heart Rate Method (FHR) [25] bases a person's maximum heart rate at either 220 for men or 226 for women, and their age is subtracted from that and as such their Max heart rate is calculated (With FHR being 220 or 226):

$$\text{MHR} = \text{FHR} - \text{age}$$

*Equation 1: Fetal Heart Rate*

This method can be further elaborated on to include a user's fitness level: [26]

Male:

$$\text{MHR} = 205 - \text{Age}/2 \text{ (Athletic)}$$

$$\text{MHR} = 220 - \text{Age} \text{ (Average)}$$

$$\text{MHR} = 214 - 0.8 * \text{Age} \text{ (Poor)}$$

Female:

$$\text{MHR} = 211 - \text{Age}/2 \text{ (Athletic)}$$

$$\text{MHR} = 226 - \text{Age} \text{ (Average)}$$

$$\text{MHR} = 209 - 0.7 * \text{Age} \text{ (Poor)}$$

*Equation 2: Futher Elaborated Fetal Heart Rate*

The Karvonen Heart Rate Method [25] is known to be more accurate in determining a person's Target Heart Rate (THR), it does this by using the same method as the Fetal Heart Rate Method to find the Maximum Heart Rate (MHR), however it also requires the person's Resting Heart Rate (this formula was not used within the system):

$$80\% \text{ THR} = (\text{MHR} - \text{RHR}) * 0.8 + \text{RHR}$$

*Equation 3: Karvonen Heart Rate*

The final formula was developed by Dr. Dan Heil as a study at the University of Massachusetts observing 1500 walkers [25] and is known as the Heil Method. It calculates Maximum Heart Rate based using a person's body weight (in Lbs). This formula differs for men and women in the case that the 4.5 at the end of the formula is removed for women:

Men:

$$\text{MHR} = 211.415 - (0.5 * \text{age}) - (0.05 * \text{weight in lbs}) + 4.5$$

Women:

$$\text{MHR} = 211.415 - (0.5 * \text{age}) - (0.05 * \text{weight in lbs})$$

*Equation 4: Heil Heart Rate*

Of the three methods, both the Fetal and Heil Method were used to most accurately acquire the maximum heart rate. Within the system both method are average to get the most accurate result:

```
private int GetMaximumHeartRate (int Age, int FitnessLevel, bool IsMale) {
    int MaxHeartRate = 0;
    if (IsMale) {
        if (FitnessLevel == 1) {
            MaxHeartRate = 205 - Age / 2;
        } else if (FitnessLevel == 2) {
            MaxHeartRate = 220 - Age;
        } else if (FitnessLevel == 3) {
            float num = 0.8f * Age;
            MaxHeartRate = 214 - (int)num;
        }
    }
    else {
        if (FitnessLevel == 1) {
            MaxHeartRate = 211 - Age / 2;
        } else if (FitnessLevel == 2) {
            MaxHeartRate = 226 - Age;
        } else if (FitnessLevel == 3) {
            float num = 0.7f * Age;
            MaxHeartRate = 209 - (int)num;
        }
    }
    return MaxHeartRate;
}
```

*Code 15: GetMaximumHeartRate Function*

Within the Heil Heart Rate function, the formula is based off Lbs, as such a function was made to convert KG to LB:

```
private int GetMaxHeartRateHeil (int Age, int WeightKG, bool IsMale) {
    float Lbs = 0.0f;
    float TargetRate = 0.0f;

    if (IsMale) {
        Lbs = ConverKgToLb ((int)WeightKG);
        TargetRate = 211.415f - (0.5f * Age) - (0.05f * Lbs) + 4.5f;
    }
    else {
        Lbs = ConverKgToLb ((int)WeightKG);
        TargetRate = 211.415f - (0.5f * Age) - (0.05f * Lbs);
    }
    return (int)TargetRate;
}
```

*Code 16: GetMaximumHeartRateHeil Function*

## Target Heart Rates

In terms of exercising and why monitoring your heart rate is beneficial to exercise and finding out if (based on heart rate) a person is doing too much or not enough in a work out, Target Heart Rates are important. Target Heart Rates are found by getting a percentage of the Maximum Heart Rate, such as some people (or personal trainers) consider for a workout the Target Heart Rate Zone of between 50% and 85% of the Maximum Heart Rate.

In terms of the Heart Health system Target Heart Rate is split up into 3 different Workout levels or Zones, based on the patient's own preference as to how hard they want to work:

- Zone 1 (50% - 65% of MHR)

Zone 1 is ideal for people who are new to exercise or are staring again after a break.

- Zone 2 (65% - 80% of MHR)

Zone 2 is ideal for people who want to lose weight. At this level, fat is the source of energy for the body. It will also improve a person's fitness and endurance.

- Zone 3 (80% - 95% of MHR)

Zone 3 will help get a person's body used to exercising at a faster pace. Lung capacity will improve. This level of intensity will also help with weight control.



Target Heart Rates are displayed within the system to encourage users to work harder within their workout. It also noted that once a user's works towards their target heart rate it gets more difficult to raise it, this is taken into consideration within the heart health system.

### **Heart Rate Simulation**

As the system uses a Heart Rate estimator, accurate measurements had to be taken of how quickly heart rate increased depending on specific exercises (Squats Jumping Jacks) as well as how quickly it cooled down, and as such a heart rate monitor was purchased of the Kinetik Medical Heart Rate Monitor:



*Figure 41: Kinetik Heart Rate Monitor*

Included with the Kinetic heart rate monitor was a chest strap and watch for displaying the current heart rate. The watch also provided a clock as well as a stop watch. On setup of the watch, it requires the person's date of birth as well as their weight (in either KG or LB), and a training zone (Target Heart Rate Zone).

The Kinetic Heart Rate Monitor was used to get an accurate reading of heart rates which will be implemented into the system. From usage of the Heart Rate Monitor a few things were found which came into implementation within the HeartRateEstimation class.

- At the beginning of the workout the heart rate raises quite slowly, doesn't properly raise normally until around 5 or more actions performed (Squats or Jumping Jacks).
- Heart rate increases faster if actions are performed in succession. Actions are performed immediately after one another.
- Cool down decreases slowly but after a while of decreasing, it decreases fast.
- Once the Heart rate reaches the Target Heart Rate depending on the user's fitness level, it increases more slowly, which means more work must be done to increase which puts strain on the body.
- The raises to the heart rate depending on the action performed are near estimates of how the heart rate was raised:

```

void SquatPerformed()
{
    if (HeartRate <= RestHeartRate)
        StartingWorkout = true;
    int Level = 0;
    if (FitnessLevel == 1)
        Level = 3;
    else if (FitnessLevel == 2)
        Level = 2;
    else if (FitnessLevel == 2)
        Level = 1;
    if (StartingWorkout)
        HeartRate += 1.95f;
    else {
        if (HeartRate >= GetTargetHeartRate(GetCurrentMaxHeartRate(), Level,
3))
        {
            HeartRate += 1.1f;
        }
        else
        {
            if (TimeSinceLastAction <= 2.0f)
            {
                HeartRate+=3.8f;
            }
            else
            {
                HeartRate += 2.1f;
            }
        }
    }
}

```

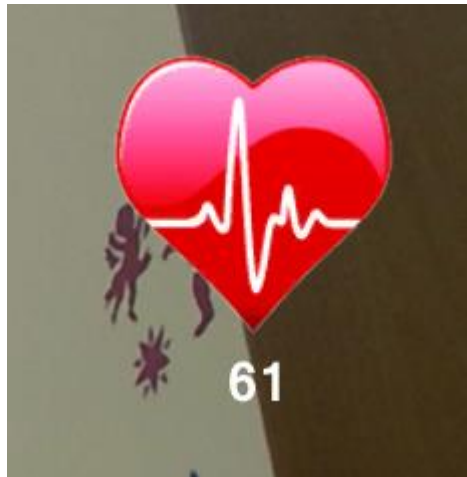
*Code 17: SquatPerformed Function*

## Heart Beat Animation

With the HeartRateEstimation class made, a stable Heart Rate was displayed on screen to the user, a new class made was HeartAnimate. HeartAnimate which requires two images (Textures) would swap both textures each time the heart was to beat:

```
currentRate = Camera.main.GetComponent<HeartRateEstimation> ().GetCurrentHeartRate ();
rate = 60.0f/currentRate;
```

*Code 18: Heart Beat Animation Code*



*Figure 42: Heart Beat Animation*

## Saving Workout Data

Workout data is always saved once the user decides to exit the game/workout in progress. Workout data is not saved however if, the user was not detected by the Kinect (User calibration). Workout data is not saved in the Orbs game.

```
if (Input.GetKey(KeyCode.Escape)) {
    if ((CaptureData) && (WorkoutStarted)){

        PatientWorkout workout = new PatientWorkout{
            WorkoutNumber = detailsdata.GetNumberOfWorkouts(PlayerPrefs.GetString("CurrentUser"))+1,
            WorkoutDate = DateTime.Now,
            WorkoutType = PlayerPrefs.GetString("GameMode"),
            HeartRate = this.GetComponent<HeartRateEstimation>().GetAverageHeartRate(),
            MaxHeartRate = this.GetComponent<HeartRateEstimation>().GetCurrentMaxHeartRate(),
            Comment = "",
            SquatNum = this.GetComponent<WorkoutListener>().Squats(),
            JumpNum = this.GetComponent<WorkoutListener>().Jumps(),
            WorkoutLength = (int)this.GetComponent<WorkoutListener>().WorkoutTime();

            detailsdata.AddWorkout(PlayerPrefs.GetString("CurrentUser"), workout);}
```

```

PlayerPrefs.DeleteKey("GameMode");
Destroy (Camera.main.gameObject);
Destroy(this);
Application.LoadLevel("MenuScene");}

```

*Code 19: Game Exit and Workout Data Save Code*

## Simon Says Game

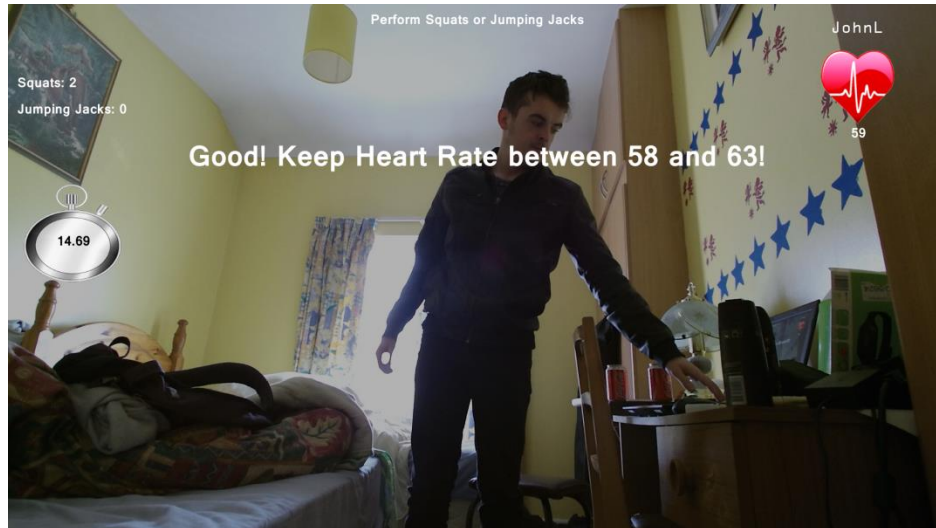
A simple “Simon Says” game was also created, where gestures must be performed on time and correctly to progress through the game. Another Gesture Listener (SimonSaysListener) was created for this as more gestures are included other than the basic workout gestures. (Wave, RaiseRightHand, RaiseLeftHand)



*Figure 43: Simon Says Game Screen*

## Heart Racer Game

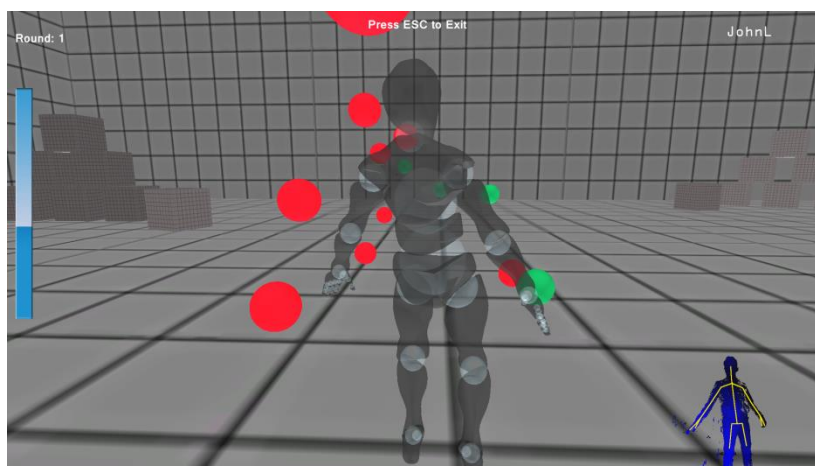
A Heart Racer game was also created, the premise of the game is to accelerate the heart rate to a certain range within a short amount of time. This is done using the HeartRateEstimation class.



*Figure 44: Heart Racer Game Screen*

## Orbs Game

The final game available is an Orb Dodger type game based off the prototype game made (Figure 37). The game is made specifically for a warmup and does not record workout data. It uses the AvatarController class to control an Avatar and dodge incoming red orbs and acquire green orbs.



*Figure 45: Orbs Game Screen*

## Application Login

On start-up of the application, the login screen is introduced where the patient must login with their username provided by the Doctor/Physiotherapist. If it is the patients first login, they do not have to enter a password, just their username, once entered they choose their own password. This login screen isn't as secure as the website login. Passwords are not hashed. Once a patient is logged in, any time they exit the application their account is logged in, unless they chose to log out.

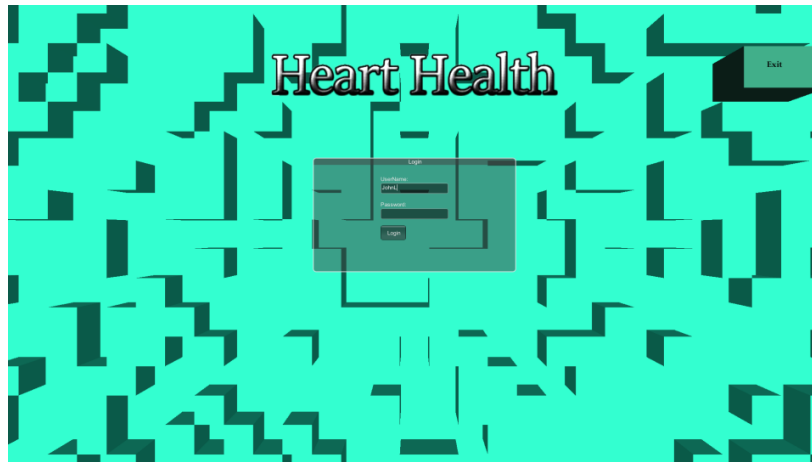


Figure 46: Application Login Screen

## Account Information

Patients uses the Heart Health application can also view their Account Details as well as their overseeing Doctor/Physiotherapist's account information. Patients also have the option to edit their some fields within their Account information (PatientDetails database class). Patients also have the option to change their password.

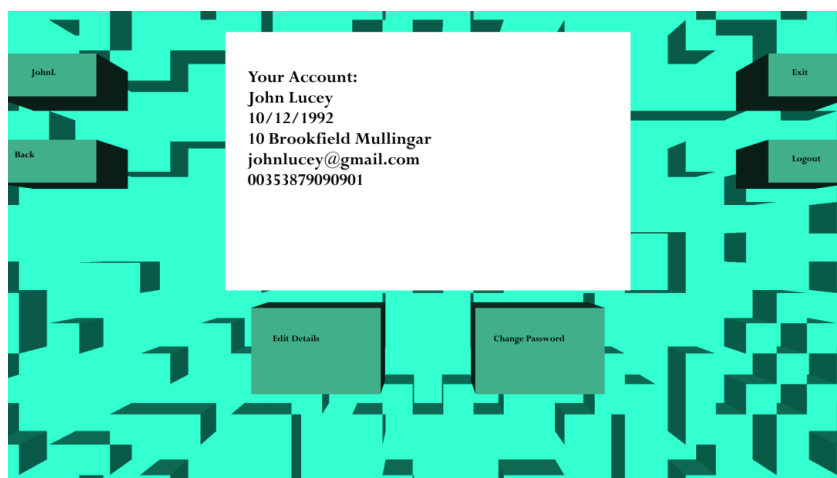


Figure 47: Account Information Screen

## 5.3 Problems Encountered

This section covers the problems encountered during implementation of the three projects and how they were solved.

### 5.3.1 Database Connection Implementation

An issue which kept occurring within the Database connection Implementation (MongoConnectApp project) was that database query must always be correct or the entire system will crash. With simple code such as Code 20, if the username variable was not located within the database, the system would crash.

```
var data = doctorlogin.Collection.Find(Query.EQ("DUserName", username)).Single();
```

*Code 20: Username Query*

Extra measures had to be taken to ensure that this did not happen such as creating a CheckUserName Function, which retrieved a list of all Usernames within the database and checked if the username entered was among them, if not then the query would not be run.

```
public bool CheckUserName(string username)
{
    List<string> userlist = GetAllUsernames();
    if (userlist.Contains(username))
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

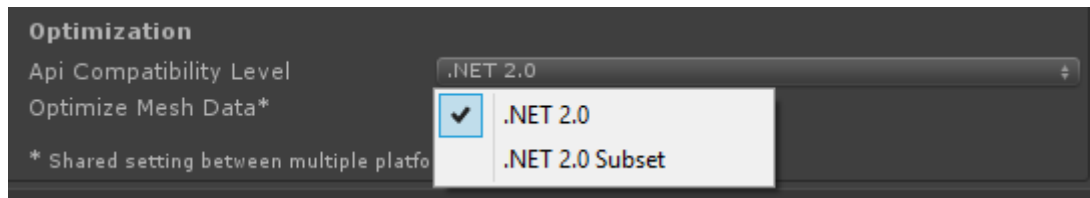
*Code 21: CheckUserName Function*

### 5.3.2 Website Implementation

For the website implementation, it was planned that Python with Django was to be used, to create the website and manage the MongoDB, however during the beginning of the project, the database classes (MongoConnectApp) were already written in C#, this was for Unity support. A DLL was created of the database classes and it was assumed there would be some way to use these classes within Python. There is a way in such it could be done, however it was quite complicated and as such it was easier and a better idea to use a framework which supported these C# classes of ASP.NET, which is mainly in C#.

### 5.3.3 Application Implementation

On implantation of the Heart Health Application in Unity, it was planned that the Database Connection DLL (MongoConnectApp) be used as a reference within the Application. The DLL was referenced however issues arose within Unity where it would constantly crash, saying there was a problem with the DLL, and so as such the Database were put as normal classes into Unity and from there, there was no issues. Another issue which arose was on building the Application as an exe (Executable) within Unity. It would not build. The problem was researched and it had to do with a setting within the Player Project Setting. The API Compatibility Level was set to .NET 2.0 Subset, which do not agree with imported DLLs such as the Mongo.Bson DLL. This was changed to .NET 2.0 and the problem was fixed:



*Figure 48: Unity Issue*

## 5.4 Conclusion

This chapter has covered the entire system implementation as well as issues which had occurred in each part of the system. The system as a whole comprised of three projects;

- The MongoConnectApp, which was the Database Connection classes as a DLL.
- The HeartHealthWebsite, which was the Doctor/Physioterapists system for administration of Patient Accounts.
- The HeartHealthKinect, which was the rehabilitation system for Patients.



## **6 Testing and Evaluation**

This chapter discusses the testing strategies deployed in this project to ensure verification and validation practices are implemented and followed consistently throughout the development life cycle of this system. As the Iterative and Incremental Method was chosen testing is a vital part of the lifecycle and testing is performed at the end of every increment of a cycle.

The testing of this system is separated into many different types of testing such as unit testing, integration testing and Validation testing. Software testing is important through each increment in this project. Both White Box Testing and Black Box Testing are performed as part of the project.

### **6.1 Testing Methods**

#### **6.1.1 Black Box Testing**

Black Box testing also known as functional testing involves testing where the tester has no previous knowledge of the interior workings of the application and is unaware of the expected outcome of the unit to be tested, so typically a tester will interact with system's user interface by providing inputs and examining outputs without knowing how each component works. This testing mainly covers the system's functionality.

[27]

#### **6.1.2 White Box Testing**

White Box Testing also known as clear box testing involves testing where the user has knowledge of the internal structure of the system and its components. The user is aware of expected outcomes of certain inputs of the system and needs to have a look a source to get a better understanding of components and how each components works.

[27]

#### **6.1.3 Unit Testing**

Unit Testing is where individual units or classes within the system are tested to determine if they are fit for use. Depending on what the class does all its procedures

and functions are tested to determine their functionality without the system and to find errors within the classes. [28]

#### **6.1.4 Integration Testing**

Integration testing is where individual units and classes are combined and tested as a group, this usually occurs after the unit testing stage. Classes which are intended to be used together are tested and all available outcomes are determined to find errors within the classes and how both classes interact with one another. [28]

#### **6.1.5 Verification and Validation Testing**

Verification and Validation testing is testing to determine if the system is built correctly and according to the specification (In the case of the proposed system is it following the User requirements correctly). It determines this by using Verification tests and Validation tests.

Verification tests ask the question of “Are we building the product right?” which means verifying the correct design and code which was written for the system.

Validation tests ask the question of “Are we building the right product?” which means checking and testing the actual implemented system and determining if the system was implemented correctly.

#### **6.1.6 Testing Methods Conclusion**

The testing methods which will be implemented during the development of the project will be Unit Testing, Integration Testing and Verification and Validation testing. These testing methods were chosen as they suited the development cycle of the project and how each increment and iteration of the system is to be tested based on the Software Development Methodology chosen.

### **6.2 Testing Phase**

The tests which have taken place are Test-cases, Unit Tests, Integration Tests and Verification and Validation Tests. For each of the tests done, the results are also provided.

### 6.2.1 Test Cases

A few simple test cases of main parts of how the system is expected to work are detailed in the form of test-case scenarios.

Test Case	Test Case Name	Owner	Description	Pre-Condition	Post-Condition	Failed Condition
1	Doctor Account Creation	Doctor System	Doctor has ability to create own user account in doctor system,		Account created, Doctor account details are added to the database	No Account is created
2	Doctor Account Login	Doctor System	Doctor can log into own account with details entered on account creation	Account was created (test case 1)	Account logged in and doctor has access to doctor system	Login failed with correct login details
3	Patient Account Creation	Doctor System	Doctor can create patient user account	Doctor logged in (test case 2)	Patient account created, and details added to the database, Patient Account can be viewed as list of patient accounts in doctor's system	No patient account is created
4	Patient Account Login	Patient System	Patient can log into patient system	Doctor had created patient account (test case 3)	Account logged and patient has ability to perform exercises, play game and view records in patient's system	Login failed with correct login details
5	Perform Exercises	Patient System	Exercise scenario commences, and patient performs exercises in front of Kinect which are recorded	Patient has logged in (test case 4)	Exercise scenario commences, and exercises performed are recorded	Exercise scenario fails to start or does start but Kinect is not tracking movement
6	Exercise Results are recorded	Patient System	Exercises performed in front of the Kinect are recorded	Patient has chosen to perform exercises (test case 5)	Exercises which are performed by the patient are recorded are inserted into the database	Exercises are not recorded and there is no data received from the workout performed
7	Heart rate tracking	Patient System	Heart rate is tracked while Exercises which performed	Patient has chosen to perform exercises (test case 5)	Heart rate is tracked and recorded as patient performs workout	No rate tracking does not work and no heart rate results are received from workout performed

Table 8: Test cases

### 6.2.2 Test Case Results

The following are the test case results from Table 8:

Test Case	Test Case Name	Owner	Description	Procedure Steps	Expected Result	Test Result
1	Doctor Account Creation	Doctor System	Doctor has ability to create own user account in doctor system,	Home Screen, click on Register, Enter Username and password	Account created, Doctor account details are added to the database	Account was created and brought to AccountDetails screen
2	Doctor Account Login	Doctor System	Doctor can log into own account with details entered on account creation	Home screen, click on Login, Enter correct username and password	Account logged in and doctor has access to doctor system	Account was logged in, Account username is visible at the top of the screen
3	Patient Account Creation	Doctor System	Doctor can create patient user account	Account is logged in, PatientView Screen, click Create Patient Button, enter patient details	Patient account created, and details added to the database, Patient Account can be viewed as list of patient accounts in doctor's system	Patient account was created
4	Patient Account Login	Patient System	Patient can log into patient system	On Heart Health Application, username entered, if first login, password is not needed	Account logged and patient has ability to perform exercises, play game and view records in patient's system	Since first login prompt to enter new password appears
5	Perform Exercises	Patient System	Exercise scenario commences, and patient performs exercises in front of Kinect which are recorded	Account logged in, select exercise button from main menu, select game	Exercise scenario commences, and exercises performed are recorded	Basic Workout game selected, Basic workout screen appears, text appears asking for calibration
6	Exercise Results are recorded	Patient System	Exercises performed in front of the Kinect are recorded	Exercises are performed and then ESC button is pressed to exit scene	Exercises which are performed by the patient are recorded are inserted into the database	Exercises performed of Squats and Jumping Jacks are recorded
7	Heart rate tracking	Patient System	Heart rate is tracked while Exercises which performed	Exercises and being performed	Heart rate is tracked and recorded as patient performs workout	Heart rate estimation is working, heart rate is estimated during workout

Table 9: Test case results

### 6.2.3 Unit Tests

A summary is provided of all the Unit tests which have taken place:

Test#	Result	Test#	Result	Test#	Result
1.1	Pass	3.5	Pass	5.11	Pass
1.2	Pass	3.6	Pass	6.1	Pass
1.3	Fail	3.7	Fail	6.2	Pass
1.4	Pass	3.8	Pass	6.3	Pass
1.5	Pass	3.9	Pass	6.4	Pass
1.6	Pass	4.1	Pass	6.5	Pass
1.7	Pass	4.2	Pass	6.6	Pass
1.8	Pass	4.3	Pass	7.1	Pass
1.9	Pass	4.4	Pass	7.2	Pass
1.10	Pass	4.5	Pass	7.3	Pass
1.11	Pass	4.6	Pass	8.1	Pass
1.12	Pass	4.7	Pass	8.2	Pass
1.13	Pass	4.8	Pass	8.3	Pass
2.1	Pass	4.9	Pass	8.4	Pass
2.2	Pass	5.1	Pass		
2.3	Pass	5.2	Pass		
2.4	Pass	5.3	Pass		
2.5	Fail	5.4	Pass		
2.6	Pass	5.5	Pass		
2.7	Pass	5.6	Pass		
3.1	Pass	5.7	Pass		
3.2	Pass	5.8	Pass		
3.3.	Pass	5.9	Pass		
3.4	Pass	5.10	Fail		

*Table 10: Summary of Unit Tests (Appendix)*

### 6.2.4 Heuristic Evaluation

As previously stated in Heuristics sub-chapter, a Heuristic evaluation of the entire system (Heart Health Application and Heart Health Website) would take place. Two heuristic evaluations have taken place, one by myself, and another by another student (who has remained anonymous). This was done in order to avoid bias.

Evaluated by Andrew Daly:

	Heart Health Application	Heart Health Website
Visibility of System Status	4	6
Match between System and the Real World	7	8
User Control and Freedom	7	5
Consistency and Standards	8	8
Error Prevention	4	4
Recognition over Recall	7	8
Flexibility and Efficiency of Use	6	5
Aesthetic and Minimalist Design	5	6
Help and Documentation	3	5
Help User Recognize, Diagnose, and Recover from Errors	3	5

*Table 11: Heuristic Evaluation (Andrew Daly)*

Evaluated by Student:

	Heart Health Application	Heart Health Website
Visibility of System Status	4	4
Match between System and the Real World	6	7
User Control and Freedom	5	5
Consistency and Standards	9	10
Error Prevention	5	6
Recognition over Recall	5	7
Flexibility and Efficiency of Use	7	8
Aesthetic and Minimalist Design	8	10
Help and Documentation	4	4
Help User Recognize, Diagnose, and Recover from Errors	5	5

*Table 12: Heuristic Evaluation (Student)*

### **6.3 Conclusion**

This chapter has covered the different testing methods which were implemented throughout the course of the project as well as results from Test cases performed, and results from Unit test. Two Heuristic Evaluations were also performed of both myself another student.

## 7 Conclusion

This chapter will give a final overall reflection of the project journey. It will also detail any future work planned with the project and a personal statement of my own opinion of the overall project and system.

Sections included in this chapter:

**Walkthrough:** This section gives summary of each section in the project.

**Future Work:** This section details possible future work for the project.

**Personal Statement:** This section is my personal statement of how I found the project.

### 7.1 Walkthrough

#### **Research and Analysis**

The Research and Analysis chapter investigates Similar System or Alternatives to the Heart Rate system. Heuristic evaluations are also done of these systems. Research Topics such as Heart Rates are also discussed. Additional Research from interviewing Zachery Tan was included. The Functional Requirements are stated.

#### **Research of Technologies**

The Research of Technologies chapter investigates Alternative technologies such could have been used by the system, these technologies are compared and the best technologies which were in the end used by the system are chosen. An overview of the architecture of the system is also done.

#### **Design**

The Design chapter looks into different Software Development Methodologies and states why the Iterative and Incremental Method is the best for the project. Design Diagrams are also displayed which contributed to the overall design of the system.

#### **Implementation**

The Implementation chapter covers implementation of each component within the project, it also observes the problems encountered and how they solved.



## **Testing and Evaluation**

The Testing and Evaluation chapter cover Testing methods which were used over the course of the project, as well as test cases and results from test cases. It only covers results from unit tests and a heuristic evaluation of the system is done.

## **7.2 Future Work**

This section shall cover possible areas where the Heart Health system could be worked on in the future.

### **7.2.1 Unity Web Player**

Possible future work to the system could be incorporating it into web such as the Unity Web Player. This will be very beneficial to the system as it eliminates the need to download the Heart Health Application, whereas on Web Player, the application can be run immediately once the webpage is accessed.

The unity Web Player and how to create a project with the Unity web player has not been researched, so it is not known by myself how difficult this would be.

### **7.2.2 Heart Rate Monitoring**

Possible future work to the system would actually be to utilize the Kinect 2 heart rate monitoring capabilities and incorporating it into the system instead of the Heart rate estimation. It was planned to do this at first however due to time constraints and lack of material available in utilizes the Kinect 2 thermal sensor, it was decided heart rate estimation would be used instead.

### 7.3 Personal Statement

I very much enjoyed working on the Heart Health system, and am happy I choose to create a system such as this. Motion capture and gaming is an area I am very much interested in, and I believe that a lot more can be done in motion capture games.

In creating this system I wanted experience with many different and new technologies to broaden my experience, which is why I choose to use MongoDB which is new and gaining in popular. And from using it, I can see actually as to why it is so popular. It wasn't planned to use ASP.NET as part of the project, I had intended to use Python with Django to gain more experience with Python and the Django framework, however I am very happy I ended up going with ASP.NET, it vastly improved my knowledge of C# and with it I feel more confident at web development which was an area I wasn't as familiar with.

I do wish however I had more time to work on getting the Kinect 2 heart rate monitoring to work, however researching online for examples on how it can be achieved turned up with little or no results. There was an application for free which did do heart rate tracking by Dwight Goins, however for some odd reason, the application would not work on my computer due to some graphics card incompatibility. This was disappointing as even using the application and seeing the results it returned could of helped in my take of the heart rate monitoring.

In conclusion I very much enjoyed the project, however I wish I had more time to work on it, things got delayed with the many assignments of fourth year, however I am very happy with the progress I made.

All development is hosted on GitHub:

<https://github.com/andydaly>

## References

- [1] "Exercise and Heart Disease." [Online]. Available: <http://www.webmd.com/heart-disease/guide/exercise-healthy-heart>. [Accessed: 25-Mar-2015].
- [2] "Doctor-patient interaction." [Online]. Available: <http://www.noldus.com/human-behavior-research/application-areas/doctor-patient-interaction>. [Accessed: 25-Mar-2015].
- [3] C. C. Abt, *Serious Games*. University Press of America, 1987.
- [4] International Conference on Games and Virtual Worlds for Serious Applications, K. Debattista, Institute of Electrical and Electronics Engineers, and IEEE Computer Society, *2nd International Conference on Games and Virtual Worlds for Serious Applications VS-GAMES 2010: proceedings : 25-26 March 2010, Braga Portugal*. Los Alamitos, Calif.: IEEE Computer Society, 2010.
- [5] "Xbox Fitness | Work Out Smarter with Famous Trainers on Xbox One." [Online]. Available: <http://www.xbox.com/en-IE/xbox-one/games/xbox-fitness>. [Accessed: 25-Mar-2015].
- [6] "Ireland | Lloyds Online Doctor." [Online]. Available: <https://www.lloydsonlinedoctor.ie/>. [Accessed: 25-Mar-2015].
- [7] "UsabilityNet: Heuristic evaluation." [Online]. Available: <http://www.usabilitynet.org/tools/expertheuristic.htm>. [Accessed: 26-Mar-2015].
- [8] "10 Heuristics for User Interface Design: Article by Jakob Nielsen." [Online]. Available: <http://www.nngroup.com/articles/ten-usability-heuristics/>. [Accessed: 26-Mar-2015].
- [9] L. Sherwood, *Human Physiology: From Cells to Systems*. Cengage Learning, 2008.
- [10] "Cardiovascular effects of strenuous exercise in adult recreational hockey: the Hockey Heart Study." [Online]. Available: <http://www.ncbi.nlm.nih.gov/pmc/articles/PMC99308/>. [Accessed: 26-Mar-2015].
- [11] "\* HRmax (Fitness) - Definition - Online Encyclopedia." [Online]. Available: <http://en.mimi.hu/m/fitness/hrmax.html>. [Accessed: 26-Mar-2015].
- [12] "Target Heart Rates." [Online]. Available: [http://www.heart.org/HEARTORG/GettingHealthy/PhysicalActivity/FitnessBasics/Target-Heart-Rates\\_UCM\\_434341\\_Article.jsp](http://www.heart.org/HEARTORG/GettingHealthy/PhysicalActivity/FitnessBasics/Target-Heart-Rates_UCM_434341_Article.jsp). [Accessed: 26-Mar-2015].
- [13] "Kinect V2 Preview SDK now available includes UNITY3D plugin - Microsoft UK Faculty Connection - Site Home - MSDN Blogs." [Online]. Available: [http://blogs.msdn.com/b/uk\\_faculty\\_connection/archive/2014/07/17/kinect-v2-preview-sdk-now-available-includes-unity3d-plugin.aspx](http://blogs.msdn.com/b/uk_faculty_connection/archive/2014/07/17/kinect-v2-preview-sdk-now-available-includes-unity3d-plugin.aspx). [Accessed: 25-Mar-2015].
- [14] "Amazon Elastic Beanstalk Review 2015 | Amazon Cloud Application Platform." [Online]. Available: <http://cloud-hosting-review.toptenreviews.com/amazon-elastic-beanstalk-review.html>. [Accessed: 25-Mar-2015].
- [15] "Rails vs Django: an in-depth technical comparison." [Online]. Available: <https://bernardopires.com/2014/03/rails-vs-django-an-in-depth-technical-comparison/>. [Accessed: 25-Mar-2015].
- [16] "ASP.NET | The ASP.NET Site." [Online]. Available: <http://www.asp.net/>. [Accessed: 25-Mar-2015].

- 
- [17] “Do What You Could Never Do Before | MongoDB.” [Online]. Available: <http://www.mongodb.com/what-is-mongodb>. [Accessed: 25-Mar-2015].
- [18] “What is MySQL? - Definition from WhatIs.com.” [Online]. Available: <http://searchenterpriselinux.techtarget.com/definition/MySQL>. [Accessed: 25-Mar-2015].
- [19] “How Does The Kinect 2 Compare To The Kinect 1?” [Online]. Available: <http://zugara.com/how-does-the-kinect-2-compare-to-the-kinect-1>. [Accessed: 25-Mar-2015].
- [20] “What is the difference between Kinect and Kinect 2? - Quora.” [Online]. Available: <http://www.quora.com/What-is-the-difference-between-Kinect-and-Kinect-2>. [Accessed: 25-Mar-2015].
- [21] “Git - Git Basics.” [Online]. Available: <http://git-scm.com/book/en/v2/Getting-Started-Git-Basics>. [Accessed: 25-Mar-2015].
- [22] “Extreme Programming: A Gentle Introduction.” [Online]. Available: <http://www.extremeprogramming.org/>. [Accessed: 25-Mar-2015].
- [23] C. Larman and V. R. Basili, “Iterative and Incremental Development: A Brief History,” *Computer*, vol. 36, no. 6, pp. 47–56, 2003.
- [24] “Resting Heart Rate Chart.” [Online]. Available: <http://www.topendsports.com/testing/heart-rate-resting-chart.htm>. [Accessed: 26-Mar-2015].
- [25] “Maximum Heart Rate Calculator.” [Online]. Available: <http://nowlin.com/heartrate.htm>. [Accessed: 25-Mar-2015].
- [26] “Target Heart Rate Calculator.” [Online]. Available: <http://www.stevenscreek.com/goodies/hr.shtml>. [Accessed: 25-Mar-2015].
- [27] “Differences Between Black Box Testing and White Box Testing | Software Testing Fundamentals.” [Online]. Available: <http://softwaretestingfundamentals.com/differences-between-black-box-testing-and-white-box-testing/>. [Accessed: 26-Mar-2015].
- [28] “The Difference Between Unit Tests and Integration Tests.” [Online]. Available: <http://www.typemock.com/unit-tests-integration-tests/>. [Accessed: 25-Mar-2015].

## Appendix

### MongoConnectApp

DatabaseDAccount.cs

Function	Test#	Input	Expected Result	Result
DatabaseDAccount {Constructor}	1.1	<connection string>	Connected to database dependant on string	Pass
CreateLogin	1.2	Username=test123 Password=testtest Createdate= datetime.now Isconfrimed=true Confirmationtoken=xxxx Lastpassfailedate=datetime.now Failuressincesuccess=0 Passwordhangeddate=date.now	Login created	Pass
Edit	1.3	Username=test123 Password=testtes Createdate= datetime.now Isconfrimed=true Confirmationtoken=xxxx1 Lastpassfailedate=datetime.now Failuressincesuccess=1 Passwordhangeddate=date.now	Details changed	Fail
EditLastLoginDate	1.4	Username=test123 Date = datetime.now	Lastlogindate changed	Pass
EditLastPasswordFailureDate	1.5	Username=test123 Date = datetime.now	Lastpasswordfailedate changed	Pass
EditPasswordFailuresSinceLastSuccess	1.6	Username=test123 Failures=2	PasswordFailuresSinceLastSuccess changed	Pass
ChangePassword	1.7	Username=test123 Password=hellohello	Password changed	Pass
ChangePasswordByClass	1.8	Username=test123 Password=hellohello	Password changed	Pass
CheckUserName	1.9	Username=test123	Returns true	Pass
GetLogin	1.10	Username=test123	Login class returned	Pass
CheckLogin	1.11	Username=test123 Password=hellohello	Returns true as username and password is correct	Pass
CheckLoginWthClass	1.12	Username=test123 Password=hellohello	Returns true as username and password is correct	Pass
Delete	1.13	Username=test123	Account deleted	Pass

## DatabaseDDetails.cs

Function	Test#	Input	Expected Result	Result
DatabaseDDetails {Constructor}	2.1	<connection string>	Connected to database dependant on string	Pass
Create	2.2	Username= test123 Name=john Email=temp@test Position = hosp Practice = office Bio=hey Phonenumbr=00002	Accountdetails created, login username must be created first however!	Pass
GetDoctor	2.3	Username=test123	Doctordetails returned	Pass
Delete	2.4	Username=test123	Account deleted	Pass
CheckUserName	2.5	Username=test123	Returns true	Fail (previous test was delete)
EditName	2.6	Username=test123 Name=Johnny	Name changed (test123 was remade)	Pass
EditEmail	2.7	Username=test123 temp@host	Email changed	Pass

## DatabaseDToP.cs

Function	Test#	Input	Expected Result	Result
DatabaseDToP {Constructor}	3.1	<connection string>	Connected to database dependant on string	Pass
CreateByStrings	3.2	Pusename = test1 Dusername = test2	Creates allocation collection	Pass
CreateByClass	3.3.	Pusename = test1 Dusername = test2	Creates allocation collection	Pass
GetAllocationByPatient	3.4	Pusename = test1	Returns allocation class	Pass
GetAllocationByDoctor	3.5	Dusername = test2	Returns allocation class	Pass
GetDUserNameByPatient	3.6	Pusename = test1	Returns correct DUserName	Pass
GetPUserNameByDoctor	3.7	Dusername = test2	Returns correct PUserName	Fail
CheckPUserName	3.8	Pusename = test1	Returns true	Pass
CheckDUserName	3.9	Dusername = test2	Reurrns True	Pass

## DatabasePAccount.cs

Function	Test#	Input	Expected Result	Result
DatabaseDToP {Constructor}	4.1	<connection string>	Connected to database dependant on string	Pass
CreateLogin	4.2	Username= test Password=pass Isconfirmed = false	Login account is created	Pass
CreateFullAccount	4.3	PatientLogin var, PatientDetails vat	Both accounts created	Pass
ChangePassword	4.4	Password=pass New =pass1	Password changed	Pass
CheckUserName	4.5	Username= test	Returns true	Pass
CheckLogin	4.6	Username= test Password=pass1	Returns true	Pass
CheckIfConfirmed	4.7	Username= test	Returns false	Pass
AccountConfirmed	4.8	Username= test	Isconfirmed is set to true	Pass
Delete	4.9	Username= test	Account deleted	Pass

## DatabasePDetails.cs

Function	Test#	Input	Expected Result	Result
DatabaseDToP {Constructor}	5.1	<connection string>	Connected to database dependant on string	Pass
Create	5.2	Username = tdawg Name =tester Email = tdawg Address= dublin Dateofbirth=1/1/2012 Phonenumber=00002 Fitnesslevel=1 Condition=sick WeightKG=60 Ismale=false	Creates new collection	Pass
GetPatient	5.3	Username = tdawg	Gets patient account	Pass
AddWorkout	5.4	Username = tdawg Workoutdetails var	Adds workout	Pass
GetWorkout	5.5	Username = tdawg Workoutnum =1	Gets workout	Pass
AddDiagnosis	5.6	Username = tdawg	Adds diagnois	Pass

		PatientDiagnosis var		
GetDiagnosis	5.7	Username = tdawg Diagnosisnum =1	Gets diagnosis	Pass
AddMessage	5.8	Username = tdawg PatientMessage var	Add message	Pass
GetMessage	5.9	Username = tdawg Messagenum =1	Gets Messgae	Pass
EditAllDetails	5.10	Username = tdawg Name =tester1 Email = tdawg1 Address= dublin1 Dateofbirth=1/1/2013 Phonenumber=00003 Fitnesslevel=2 Condition=sick1 WeightKG=62 Ismales=false	Edits details	Fail
Delete	5.11	Username = tdawg	Account deleted	Pass

## HeartHealthWebsite

### AccountController.cs

Function	Test#	Input	Expected Result	Result
Register	6.1	Username=test Password=testpass	Doctor Account created	Pass
Login	6.2	Username=test Password=testpass	Logs in	Pass
Logoff	6.3		Logs Current user out	Pass
Manage	6.4	Current=testpass New=testpass1	Changes password	Pass
AccountDetails	6.5	Name=John Smith Email=test@test Position=Doctor Practice=Dublin Bio=Hello	Details added to database, created (Database Collections)	Pass
EditDetails	6.6	Name=John Smith1 Email=test@test1 Position=Doctor1 Practice=Dublin1 Bio=Hello1 PhoneNumber=00000	Details are changed	Pass



## HomeController.cs

Function	Test#	Input	Expected Result	Result
Index	7.1		Opens index View	Pass
About	7.2		Opens About view	Pass
Contact	7.3		Opens Contact View	Pass

## PatientController.cs

Function	Test#	Input	Expected Result	Result
PatientsView	8.1		Opens PatientsView view and displays table holding all patients	Pass
CreatePatient	8.2	Username=test2 Name=Jane Smith Email=test@test2 Address=Dublin DateOfBirth=1/1/2012 PhoneNumber=0000 FitnessLevel=1 Condition=sick Weightkg=60 IsMale=false	Creates patient account and patient details (Database Collections)	Pass
PatientData	8.3	Username=test2	Opens test2 patientdata page	Pass
EditPatient	8.4	Username=test2 Name=Jane Smith1 Email=test@test21 Address=Dublin1 DateOfBirth=1/1/2011 PhoneNumber=00001 FitnessLevel=2 Condition=sick1 Weightkg=61	Patient details are changed	Pass
DiagnosisListing	8.5	Diagnosisistext = very sick Usrname = test2	Diagnosis added	Pass
MessageListing	8.6	Messgaetext = cool story bro Usrname = test2	Message added	Pass