DUBLIN INSTITUTE of TECHNOLOGY
Institiúid Teicneolaíochta Bhaile Átha Cliath

Dysfunctional Breathing Diagnostic Tool
Final Year Project Report

DT221C
BSc in Computer Science (Infrastructure)

Colm Fitzpatrick

Damian Bourke

School of Computing

Dublin Institute of Technology

08/04/16

Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:

_____

Colm Fitzpatrick

08/04/16

Abstract

This project aims to provide an objective alternative to the current subjective method of detecting predominant chest breathing. The specific goal of this project is to provide an application which uses this method to identify Dysfunctional Breathing and measure the effects of breathing retraining. Research and knowledge in the area of Dysfunctional Breathing require attention as many patients go misdiagnosed. With limited studies available, meetings with respiratory professionals enabled a unique diagnostic method to be implemented in this proof of concept. Through the use of technologies such as Java, Arduino, C, Oracle SQL and two self-constructed respiratory belts, the system attempts to collect and analyse respiratory data to identify the predominant chest breathing pattern.

Acknowledgements

Damian Bourke, Supervisor

I would like to show my deepest thanks to Damian my supervisor for all the help and support he has given me throughout the project. I greatly appreciate and respect the time and effort he has given during the project.

Aisling McGowan, Chief Respiratory Physiologist

My sincere gratitude goes to Aisling for providing invaluable experience, help, and support during the project. Her willingness and desire to support research in her area of expertise makes her an asset to the scientific community.

Damon Berry, Engineer

I would like to thanks Damon for all the guidance and encouragement he has provided to me. He has surpassed the boundaries of his job requirements to provide this project with his knowledge and experience.

Ross McCann and Ryan Aylward

Finally, I would like to show my appreciation to Ross and Ryan for their role in creating the breathing retraining and belt application videos.

Table of Contents

Table of Figures

Table of Tables

1 Project Statement

This project was selected because the history and knowledge surrounding dysfunctional breathing (DB) are vague. Limited studies in this area have caused little to be known on the subject producing poor treatment for patients. As a result, more frequent and extensive studies are required in this area in order to successfully treat and understand the condition. This project aims to build on the limited knowledge available and encourage further research.

Goal

To develop an objective DB diagnostic and breathing retraining tool which measures differences in chest breathing and abdominal breathing and presents relative and accurate data in a structured report which can be used by medical practitioners to assist them in arriving at a diagnosis.

Objectives

- Research DB causes, diagnosis and treatment.
- Compare similar systems as to identify users and usability requirements.
- Research and choose technologies to be used.
- Deploy Nijmegen questionnaire.
- Source medical professional in the area and conduct meetings.
- Identify problem and solution.
- Define functional and non-functional requirements.
- Choose software development methodology.
- Create system architecture, prototypes, and test plan.
- Outline risks, issues and plan for the future.
- Construct chest and abdominal respiratory belts.
- Develop Java application which facilitates all functional and non-function requirements.

2 Background Research

2.1 Introduction

The goal of the background research chapter is to gain the information required to define the functional and non-functional requirements for this project. Relevant literature will be studied and the causes, diagnosis, and treatment of DB will be described to support the definition of functional requirements. Additionally, similar systems will be examined and scored against usability heuristics to help define non-functional requirements.

2.2 Background research

The term DB has been introduced to describe patients who display divergent breathing patterns and have breathing problems that cannot be attributed to a specific medical diagnosis[1]. Since little is known about DB it has caused confusion throughout the medical industry and in many cases has been ignored. This has resulted in a vast number of patients being misdiagnosed or overdiagnosed. Recent studies have had much success in identifying this problem. Shawn claims that one-third of the participants who had received a diagnosis of asthma by a physician had no evidence of asthma when their medications were tapered and when they were evaluated with serial assessments of symptoms, lung function, and bronchial challenge tests[2]. Such large scale overdiagnosis gives insights into the difficulty medical professionals are presented with. One of the factors in the overdiagnosis of DB is anxiety. Anxiety is frequently linked to DB and hyperventilation syndrome, with symptoms presented in both conditions. Additionally, DB and hyperventilation syndrome have also been associated with asthma, which triggers asthma-like symptoms [9]. With these asthma-like symptoms, it is difficult to distinguish between DB and asthma.

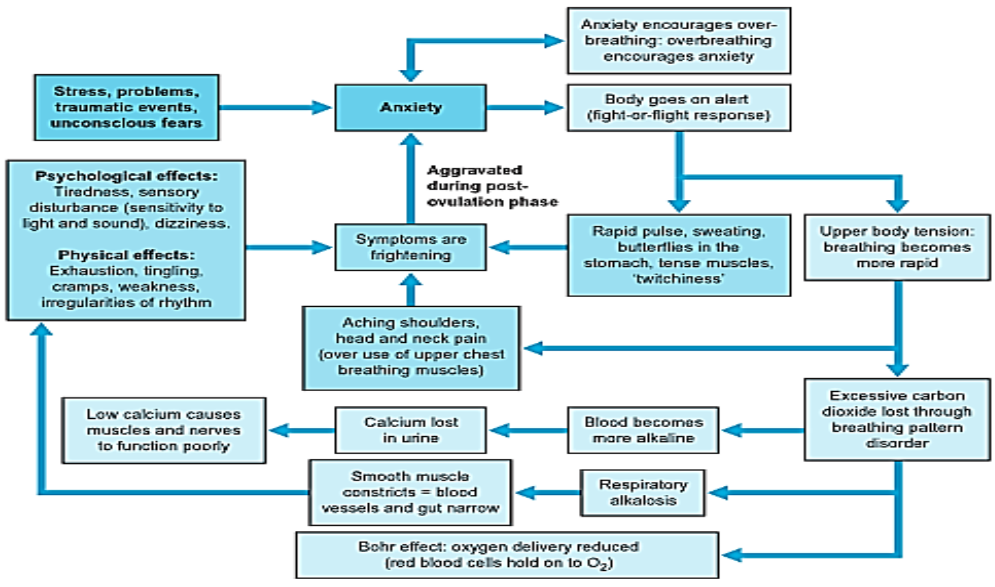


Figure 2-1 Anxiety In Dysfunctional Breathing Flowchart[3]

Figure 2-1 demonstrates the role anxiety plays in DB. By studying this flowchart, it can be seen that anxiety causes patients to over-breath. As a result, excessive carbon dioxide is lost causing respiratory alkalosis. It is respiratory alkalosis which causes the symptoms shared with DB and hyperventilation syndrome.

Diagnosis of DB can, therefore, be difficult; the characteristic symptoms are common to other diseases and there is no standard diagnostic test[4]. Recent studies on DB have produced suggestions or theories as to what it is and how to diagnose it[5][6]. Morgan claims that DB is an "imprecise description of a behaviour and symptom complex which remains unexplained"[5]. Hornsveld suggests that it may be due to a combination of hyperventilation syndrome, hypocapnia, and other less evident factors[6]. Such theories display the confusion surrounding the area.

A patient is said to suffer from hyperventilation syndrome if they frequently over-breathe or have an increased respiratory rate[7]. It is often reported that around 10% of patients in a population are diagnosed with hyperventilation syndrome[8]. Clearly visible hyperventilation is uncommon leading to less severe cases going undiagnosed. Hyperventilation is associated with DB due to large numbers of patients presenting symptoms of both. In 2008, a study to compare patients with DB and patients with asthma was undertaken[9]. The study found that hyperventilation was more frequent in DB patients than in asthmatic patients and DB patients had a higher proportion of emergency room visits. This is a significant statement as little is known about DB and it is contributing to more hospital visits than asthma.

DB can have a variety of negative effects on the body. Hyperventilation occurs when short but frequent breaths are taken. As a result, insufficient air is inhaled into the lungs which decreases gas exchange, leading to increased concentrations of CO2[10]. Hypocapnia is a deficiency of CO2 in the blood resulting from hyperventilation[10]. Additionally, Hypoxia is a reduction of oxygen (O2) supply to tissue[10] and is also a result of hyperventilation.

In theory, patients with hypocapnia should have a low end-tidal CO2 (ETCO2) level. Although some patients with hypocapnia present ETCO2 less than 35 millimeters of mercury (mmHg), most patients with hyperventilation syndrome give normal readings of 35-45mmHg ETCO2 levels. This is due to the body's ability to regulate its blood chemistry. There has been no correlation distinguished between high breaths per minute (BPM) and ETCO2 levels[11].

2.2.1 Diagnosis

Although there is currently no standard diagnostic test for DB, many different approaches and tests exist which are believed to help make a diagnosis.

Nijmegen questionnaire (NQ):

The NQ was introduced over 30 years ago as a screening tool to detect patients with hyperventilation complaints that could benefit from breathing regulation through capnographic feedback[12]. A score of 23 or over in this questionnaire indicates a positive diagnosis of hyperventilation syndrome. Normal scores in the NQ vary depending on

location. For example, data collected in England, Belgium and Holland[13][14] showed normal results of 10-12. However, data collected in China showed subjects score much lower: 4.7±4.6[12]. The sensitivity of the NQ in relation to the clinical diagnosis was 91% and the specificity 95%[15].

Datasets were acquired from two popular studies, carried out by J. Van Dixhoorn and H. Folgering, on the subject[15][16]. 263 patients participated in the first study[16]. The patients were sources from a psychiatric and a lung function laboratory. In the first study patients with symptoms of DB were compared to patients with none. DB Diagnosed non-asthmatic patients were compared to normal breathing non-asthmatic patients in the second study[15].

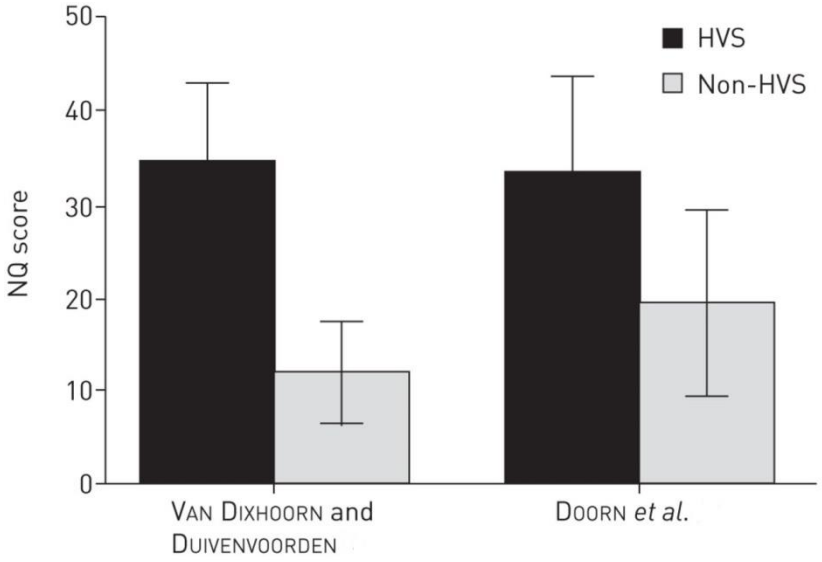


Figure 2-2 Nijmegen Scores From Both Studies[15]

Figure 2-2, displays higher Nijmegen scores for patients with Hyperventilation in both studies. Neither study attempted to differentiate "hyperventilation" complaints from an anxiety disorder or other stress or anxiety-related problems, nor has this been done in later studies[12]. As a result, the data gathered from the studies cannot confirm an exact syndrome but rather indicates an abnormality exists. This makes it an effective tool in the diagnosis process for a wide range of medical illnesses.

Carina Hagman's recent study[1], shows evidence that Nijmegen scores correspond to DB. Her study concluded that through the use of breathing retraining, patients with DB lowered their Nijmegen scores significantly whereas asthma patients did not.

The results from both of these studies demonstrate the importance of the NQ in the research to find a standard diagnosis method.

Quality of Life Questionnaire (QLQ)

Since many patients suffering from DB are overdiagnosed with Asthma, the QLQ can be utilized in the diagnosis of DB. Comparing baseline and post breathing retraining QLQ results can help to provide a diagnosis for DB. In a recent study[1], fourteen patients diagnosed with DB that had previously used asthma medication were tested against

asthmatic patients. For five years, the DB patients stopped taking their asthma medication and instead received and practiced breathing retraining exercises. Over this period of time, the quality of life in the DB patients increased more significantly than that of the asthmatic patients. This comparison between previously misdiagnosed and successfully diagnosed asthmatic patients can help identify patients with DB.

Physical Examination:

- Breath Holding: It is normal for people to have the ability to hold their breath between 25 and 30 seconds. If a patient can only hold their breath for less than 15 seconds it may indicate a low tolerance for CO2.
- Chest & Abdominal Breathing Test: The patient is to lie down and one hand is placed on their chest and the other on their abdomen. Equal upward hand movement of both the chest and abdomen is expected. Low abdominal expansion indicates evidence of hyperventilation disorder.
- Manual Assessment of Respiratory Motion (MARM): Assess and quantify breathing pattern, in particular, the distribution of breathing motion between the upper and lower parts of the rib cage and abdomen under various conditions[17]. It is a manual technique that once acquired is practical, quick and inexpensive[17].

Respiratory Examination:

- Capnography: is a non-invasive continuous analysis of the concentration of carbon dioxide in the respiratory cycle[18]. Low ETCO2 levels can be caused by hyperventilation syndrome.
- Breathing Frequency: Recording breathing pattern to detect irregularity, which is a characteristic of DB.

2.2.2 Treatment

- Pursed lip breathing (PLB) is a technique whereby nasal inhalation is performed and exhalation is performed through a resistance created by the construction of the lips[19]. This technique controls and reduces RR, and reduces shortness of breath.
- Abdominal breathing uses the diaphragm, which is the main respiratory muscle. It is a type of breathing exercise that patients are taught to promote more effective aeration of the lungs, consisting of moving the diaphragm downward during inhalation and upward with exhalation[20]. PLB should be practiced while performing abdominal breathing exercises.
- Other breathing techniques such as Equal Breathing, Alternate Nostril Breathing, etc.

The combination of the above treatments can be defined as breathing retraining. In recent years breathing retraining has had much success in treating patients who suffer from conditions such as Chronic obstructive pulmonary disease (COPD) and asthma. DB may,

however, be responsive to interventions directed at breathing retraining; improvements have been reported in clinical
series[21]. In a recent study[1], nine out of 16 DB patients showed improvements in NQ and QLQ scores compared to two out of fifteen asthmatic patients[1].

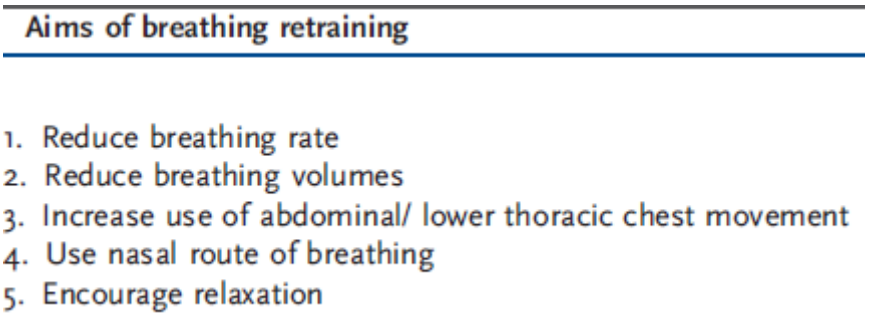


Figure 2-3 Aims of Breathing Retraining[22]

Figure 2-3 displays the aims of breathing retraining which were defined in a study by Mike Thomas in 2014[22]. Here he mentions that breathing retraining sessions conventionally involve face to face interaction between the patient and therapist[22].However, an Australian study which involved breathing retraining delivered as video instructions showed improved health status[23]. Based on the aims of breathing retraining Mike Thomas developed a set of three breathing retraining sessions. These sessions involved educating the patient, teaching nose and abdominal breathing and using controlled breath holds to measure progress[22]. From this, a similar but different breathing retraining session, which will be described further in the paper, was derived for use in this project.

2.3 Alternative Existing Solutions

After much research, no system which helped to diagnose DB was found. During the background research phase, it became evident that people of all ages can be affected by DB, but mostly older patients. Research in software designed for older and novice computer users; and similar systems for intermediate and advanced users began. This was done so that the most effective features each system possessed could be implemented in this project. This was achieved by scoring each of the software out of ten on each of Nielsen's ten Usability Heuristics. Please find the results below.

	Users			
	Novice		Intermediate	Advanced
	Software			
	PointerWare	Eldy	Masimo iSpO2	Nellcor
Visibility of system status	9	8	10	9
Match between system and the real world:	7	8	4	4
User control and freedom	8	7	1	5
Consistency and standards:	7	9	7	8
Error prevention	8	8	5	8
Recognition rather than recall	7	7	8	5
Flexibility and efficiency of use	1	1	1	5
Aesthetic and minimalist design	6	8	7	5
Help users recognize, diagnose, and recover from errors	7	8	5	7
Help and documentation	7	8	8	8

Table 1 Similar Systems Scored Against Nielson's Heuristics

Research began by reviewing the two most popular pieces of software for the novice or elderly user. PointerWave and Eldy are programs designed to simplify a computer to enable users to perform features such as sending emails and surfing the web. Evidently, these two programs are very different to that being developed in this project. However, it is essential that the user interface is developed so that every user, regardless of age or skill, can easily navigate through the program and reach the desired goal. As a result, each high scoring heuristic for the somewhat irrelevant programs carries great significance to the development of this project. Both of the programs studied scored very poorly for the flexibility and efficiency of use heuristic due to the fact more experienced users have no use for the program. However, experienced users potentially will require this program and the flexibility and efficiency of use heuristic indicates how to satisfy these user's requirements.

The next similar system that was examined was the Masimo iSpO2. Through the use of a sensor and iPhone application, it delivers to the user their SpO2, which is the amount of oxygen in the blood, in real time. As evident from Table 1, this system scored very low in the flexibility and efficiency of use. This is due to the fact that the system is very simple. The next system studied was the Nellcor Pulse Oximetry. This is a device used within medical institutions that measures the patient's SpO2, Pulse and Respiratory Rate (RR). The ability for a user to set up specific alarms triggered by certain conditions resulted in this system achieving the highest score in flexibility and efficiency of use. This extra piece of functionality is required due to the consequences of readings being out of a range and quick action is needed.

Heuristic evaluation of online COPD respiratory therapy and education video resource centre[24]

Chronic obstructive pulmonary disease (COPD) is defined as a process characterized by the presence of chronic bronchitis or emphysema that may lead to the development of airways obstruction [2]. COPD can co-exist with DB and much like DB it is most common in older patients.

The Department of Health Education and Behaviour at the University of Florida composed a heuristic evaluation for an online COPD respiratory therapy and education video resource centre called COPDFlix. Since the users of this system experience breathing problems and are mostly elderly it was studied further. The heuristic evaluation was carried out by three experts, who had expertise in Web design and health communication technologies. They were recruited to identify usability violations and to propose solutions to improve the functionality of COPDFlix. The system was evaluated using 18 heuristics. These heuristics were split into the categories of interaction and navigation, information architecture, presentation design, and information design. The results from each expert were assembled together into a single master list.

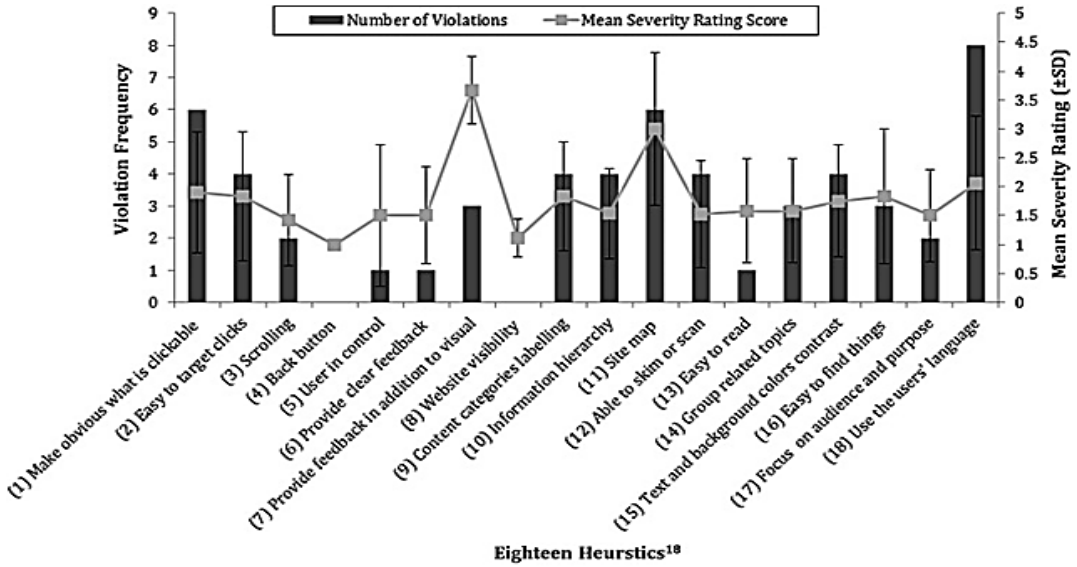


Figure 2-4 Heuristic Evaluation Results[24]

Figure 2-5 displays the eighteen heuristics used by the experts and the corresponding violation frequency. In this evaluation, heuristics were identified which the developers originally overlooked but were found to be of importance by the experts. The top three most frequently violated heuristics were: Make obvious what is clickable, Site map and Use the user's language. Special attention will be given to these heuristics during the development phase in order to learn from COPDFlix's mistakes. Additionally, COPDFlix's achievements will be considered by investigating each of the heuristics with little or no violations. Figure 2.5 displays no violations for the Back button and Website visibility heuristics.

2.4 Conclusion

The background research section has shown that further research and effort is required in the area of DB. The results of this section have made the process of defining functional requirements difficult as it shares symptoms with other illnesses and no standard diagnostic test exists. In the comparison of similar systems, it was concluded that the expected user's skills and experience will have a vast range. During the development of this software, it is essential to keep this in mind. Using the results in Table 1, the user interface can be developed so that to optimise usability and satisfaction. This can be achieved by adopting features from other systems which score highest against each of the heuristics. Special attention will be paid to ensure not to overlook any of the frequently violated heuristics in Figure 2.5, creating a system that satisfies each potential user. The next chapter will research technologies to be used in the development of the system.

3 Technology Research

3.1 Introduction

The technology research chapter aims to define each of the different types of technologies that will be used throughout the project. This is accomplished by researching potential technologies and comparing them against each other using pre-defined requirements.

3.2 Programming Languages

Requirements

- Multi-platform compatibility.
- GUI.
- Database interaction.
- Arduino communication.

Java

The Java programming language is a general-purpose concurrent class-based object-oriented programming language, specifically designed to have as few implementation dependencies as possible[25]. The Java language provides libraries such as Java Swing, JDBC, and RXTX. Java Swing is a GUI library which will be used to develop the projects GUI. JDBC is a Java standard that provides the interface for connecting from Java to relational databases[26]. The JDBC API can be used to enable the Java program and the Oracle database to communicate with each other. The RXTX API provides the Java program with the ability to read data in real-time from the Arduino through the serial port.

C

C is a general-purpose programming language which features economy of expression, modern control flow and data structures, and a rich set of operators[27]. C is not specialised to any distinct applications which makes it advantageous and powerful for many tasks[27]. The GTK+ toolkit could be used to develop the GUI. The Open Data Base Connectivity (ODBC) is an API which could be used to interact with the database. With the use of standard POSIX C, no external libraries are required to read data from the serial ports. However, it is significantly more complex and difficult to read and understand.

Python

Python is the most popular general-purpose dynamic language[28]. This language was developed to with the aim of creating readable code[28]. Tkinter is a standard GUI library for python which could be used to develop the programs GUI. The cx_Oracle extension for python could be used to make a connection from the program to the database. pySerial is the API used in python to communicate with external devices through the serial port.

Conclusion

I have selected the Java programming language for several reasons. I have a vast amount of previous experience developing in Java and it is the language I feel most comfortable using. It is the only language in which I have developed GUIs to a relatively high standard. I have also developed using the JDBC API before and enjoyed working with it.

3.3 Databases & Query Languages

Requirements

- Multi-platform compatibility.
- Graphical interface.

Oracle SQL

Structured Query Language (SQL) is a set of statements used to interact with a database. The purpose of SQL is to provide an interface to a relational database such as Oracle Database, and all SQL statements are instructions to the database[29]. Oracle SQL Developer is an Oracle Database IDE which provides users with a graphical interface for database interaction.[29]. It is possible to use Oracle SQL to run the database on any operating system.

MongoDB

MongoDB is an open-source document database that provides high performance, high availability, and automatic scaling[30]. A MongoDB record is in the form of a document which contains field and value pairs. In contrast to Oracle, MongoDB does not query using SQL but rather by using commands specific to it. This method of querying is referred to as NoSQL. It is possible to run a MongoDB database on any operating system.

PostgreSQL

PostgreSQL is an object-relational database management system (ORDBMS) based on POSTGRES, Version 4.21, developed at the University of California at Berkeley Computer Science Department[31]. Its support for JDBC and its incorporation of a vast number of procedural languages makes it a strong contender in the development of this system. With recent updates, PostgreSQL is becoming more like Oracle and not many differences exist between the two.

Conclusion

Oracle SQL is considered one of the most difficult querying languages[29]. Nonetheless, it is the approach I have decided to take as I have gathered much knowledge in the area through the development of many database systems. I have grown to enjoy developing with Oracle SQL and as a result, the standard of work developed is higher. Additionally, Oracle is the most popular option with many advantages such as it is very stable, offers most support and provides a lot of resources to developers. As a result of choosing Oracle SQL, the database which will store all data generated will be an Oracle relational database.

3.4 Spreadsheet Applications

In order to present users with data and results obtained throughout the program, a spreadsheet application was to be used. Google Docs Spreadsheet was to be used to present this information to users in a structured format, generate graphs to display to medical professionals and provide the ability to securely store and print this report. Due to the nature of the agile methodology, see section 7.2, the report generation approach has been changed. This is due to the fact that real-time graphics are not possible using this method. Instead, a Java library which supports real-time graphics and a report generation library are required.

3.5 Graph & Chart Libraries

Requirements

- Java library.
- Line graph generation and storage.
- Real-time updates.
- Save graph as Portable Network Graphics (PNG) file

JFreeChart

JFreeChart is an open source Java library used to develop charts. It is currently the most popular chart library for Java. The ability to create line charts, which are the standard format for presenting medical data such as heart beats and RR, and the capability to update them once a second is an appealing factor. Another exciting element of JFreeChart is its capability to export charts as PNG files.

GRAL

GRAL, short for graphing library, is a free graphing library for Java. Much like JFreeChart, GRAL offers near real-time graphing. It is possible to generate line graphs using this library. It is considered an easier API to develop with rather than JFreeChart but with the drawback that it is not as powerful. Exporting graphs as PNG files is also possible using this API.

JChart

JChart is another free Java chart library which can create a vast variety of different charts and graphs. The production and exportation of line graphs to PNG files are functionalities which JChart possesses. Regrettably, this library does not possess the ability to plot graphs in real-time, which is a feature essential to this project. Additionally, this library's poor API, documentation, and resources make it the least appealing of a limited selection.

Conclusion

Although I have no previous experience using JFreeChart it is the graphing library I will implement into this solution. It is presently the most popular open source Java graphing library and all other libraries strive to imitate its functionality. It offers the best API, in terms of usability, and provides descriptive documentation and resources.

3.6 Reporting Tools

Requirements

- Java library.
- Structured PDF report generation.

JasperReports

JasperReports is a renowned open source report generation Java library. Its strength lies in its capacity to produce high-quality reports from different data sources. Large sets of sample projects and other resources are available on its official website. Additionally, an active community exists which provides help and support to developers experiencing difficulties. Finally, excellent documentation is provided to the users which precisely describes elements of the library. However, JasperReports requires developers to learn a huge library which is often a deterrent.

ReportMill

ReportMill is a growing open source Java reporting tool which offers many tempting features. A report viewer is provided to view developed reports on the desktop and offers the ability to edit certain data. It is compatible with many file formats such as HTML5, Excel, PDF, and CSV. It boasts an extremely easy to use API which often only requires the use of three lines of code. Large quantities of example projects are currently available through its community on the web. Documentation is educational and provides relevant information to developers in need.

Conclusion

ReportMill is the report tool which will be implemented into the project. ReportMill's simple but effective API is an attractive aspect because no previous experience with either library is possessed. Additionally, a larger number of example projects, tutorials and help exist in the ReportMill online community, compared to JasperReports.

3.7 Minicomputer

Requirements

- Portable.
- Within price range.
- Sensor compatibility.

Raspberry Pi

Raspberry Pi is a small, cheap PC [32]. The price of Raspberry Pi ranges from €30 upwards. In terms of size, this minicomputer is very portable. However, its constant need for a steady power supply and a shutdown process to turn it off hinders its portability. The use of sensors is supported by these minicomputers. But, additional software is required in order to interact with the sensors.

Arduino

The Arduino microcontroller is an easy to use yet powerful single board computer[33]. The price varies from model to model with the average price around €50. It has a low power demand and can sufficiently be powered by a battery pack making it very portable. The use of sensors with the Arduino is much easier than the Raspberry Pi as no additional software is needed to make use of the sensors.

Conclusion

The Arduino has been selected to be used in this project. Although it is more expensive than the Raspberry Pi, it provides greater portability and offers an easier and more enjoyable experience when developing a system which requires sensors.

3.8 Respiratory Effort Sensors

In order for this project to provide users with accurate results, it was planned to make use of Ambu Sleepmate RIPmate™ Inductance Belts. These inductance belts are designed for high sensitivity and patient comfort during measurement of chest and abdominal expansion associated with respiratory effort [34]. Unfortunately obtaining the inductance belts was not possible due to the nature of medical equipment being both hard to source and expensive. This resulted in another approach to measuring respiratory effort being required. After meeting with Damon Berry a new approach was agreed upon. A summary of the meeting can be found in sections 4.5. This new approach was adapted from a recent article by Erin Gee[34], which involves using a conductive rubber cord with a known resistance of 350-400 ohms per inch / 140 - 160 ohms per centimetre that acts as a variable resistor meaning it changes resistance as it is stretched or released[35]. This conductive rubber cord will be used to construct two respiratory effort sensor belts. Using this approach results in sensitivity, quality and durability standards decreasing but enables the project to continue with a promising future. Please reference section 9.6 for the full design and construction method of the respiratory sensors.

3.9 CO2 Sensors

In order to measure ETCO2 for this project the CORIZ CO2 sensor was to be utilized to provide accurate CO2 readings. The COZIR CO2 sensor is a low power, high-performance CO2 sensor which is suited for battery operation and portable instruments[36]. During a meeting with Aisling McGowan, please reference section 4.5, it was concluded that ETCO2 could not be measured as the equipment needed to measure the pressure of the expelled gas could not be sourced. Instead, a new approach involving the combination of this system with a capnograph was agreed upon. The nose piece associated with the capnograph is to be worn by the patient along with the two respiratory belts. During the test, the respiratory professional enters in the ETCO2 values when prompted simply by referring to the value displayed on the capnograph. This method of obtaining the ETCO2 data was agreed to be a viable option as all respiratory wards are required to contain some sort of capnography technology.

3.10 Conclusion

The technology review section studied multiple technologies for each layer of the system. For each layer, the technology which best facilitated the pre-defined requirements was chosen. While following the agile software development methodology changes were made to this section at later stages. The spreadsheet application, respiratory effort sensors and CO2 sensor sections were changed as a result of obstacles encountered during the project. However, alternative solutions were provided and the project continued to progress. The next chapter will display the results of the other forms of research completed during the project.

4 Other Relevant Research Done

4.1 Introduction

This chapter aims to clearly present the findings of all other relevant research undertaken during the project. The results obtained in this section will be used to aid various tasks in the software development process.

4.2 Nijmegen Questionnaire

Forty-one participants aged 20-30 completed an NQ with the following results

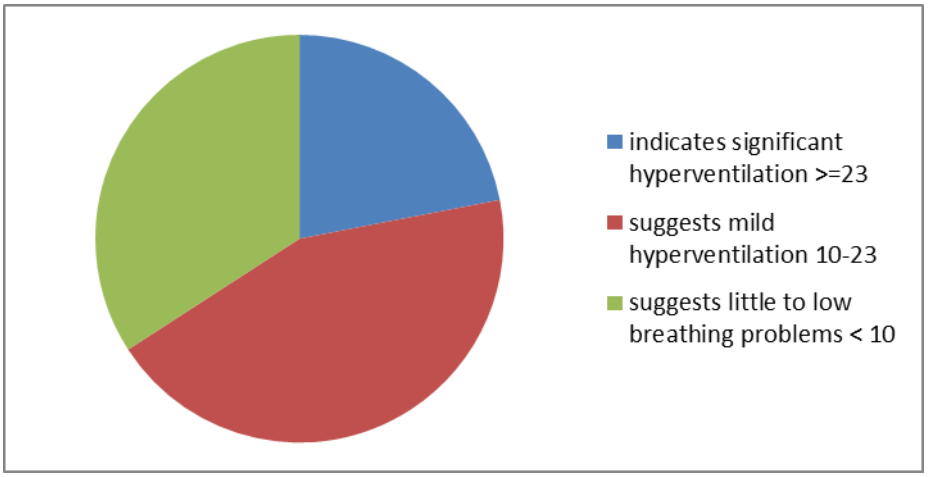


Figure 4-1 NQ Results

- Average score: 16.341
- Percentage of participants scoring 23 or over: 22%
- Percentage of participants scoring in range 10-23: 44%
- Percentage of participants scoring below 10: 34%

Examining Figure 4-1 shows that 22% of people surveyed have a score which indicates significant hyperventilation, 44% have a score which indicates mild hyperventilation and 34% have a score which indicates little to no breathing problems. Perhaps the most significant result, the average score out of all participants indicates mild hyperventilation. These results indicate that less severe cases of hyperventilation syndrome are going undiagnosed and untreated. Without a diagnosis, their condition and symptoms could deteriorate over time. Overall, this proves the importance of hyperventilation syndrome awareness as it would enable patients to manage the syndrome and reduce hospital visits.

4.3 Connolly Hospital Respiratory & Sleep Diagnostics Department Visit

On Thursday the 5th of November a visit was made to the Respiratory & Sleep Diagnostics Department of Connelly Hospital in Blanchardstown, Co. Dublin. Aisling McGowan, Chief Respiratory Physiologist, kindly arranged this meeting. On arrival, Aisling proceeded to explain the functions and uses of the equipment used in the department.

BCI 9004-000 Capnocheck Plus Capnograph

A capnograph is a device which measures CO2 levels and displays the measurements as a waveform. This capnograph is highly recognized for its reliability and accuracy. Its main features are that it provides crisp, clear waveforms and large numbers for maximum visibility with an accuracy of ±2 mmHg [37]. Additionally, RR is measured and clearly displayed as a waveform. A better understanding of the use of this device was gained through a conducted test. Respiratory rate and ETCO2 values were taken every two minutes for twenty minutes. The average of the results was calculated and presented as displayed in Table 2.

mins	CO2 mmHg	RR
2	42	15
4	39	15
6	34	17
8	37	13
10	34	15
12	36	17
14	36	16
16	32	19
18	36	13
20	36	15
Average	36.2	15.5

Table 2 ETCO2 and RR Results

When examining the results, Aisling explained their meaning. With an average ETCO2 reading of 36.2 mmHg, the results were within the normal range of 30 – 42 mmHg. Additionally, with an average RR of 15.5 the results were within the normal range of 12 – 25 BPM.

From this demonstration, the necessary information on capnography needed to implement it in this project was obtained. The importance of displaying data readings in a waveform in conjunction with the numeric display was discovered. Data must be displayed in real-time to medical professionals in a form which they are comfortable and familiar with.

The waveform is the most attractive and beneficial method of displaying ETCO2 and RR. This may be due to its usefulness in displaying the differences between baseline and post-intervention results. Measuring RR through the use of CO2 concentration was another skill learned. When the patient is inhaling CO2 levels are low and when exhaling CO2 levels are high. Using this knowledge it is possible to obtain RR using ETCO2 readings.

Finally, it was discovered that it is essential to conduct these tests for a minimum of five minutes. This is because it is natural for patients to try to control their breathing to obtain "normal" function. This is usually the case for the first few minutes as outliers are often present. Once the patient relaxes and stops trying to control their breathing realistic and accurate readings can be obtained.

Carefusion Masterscreen PFT (Pulmonary function testing)

From observing a patient being tested using the Carefusion Masterscreen extensive knowledge was gained. Firstly, the importance of certain factors such as weight, height, age, smoking status and current medicine intake was noticed. These factors are noted in the first section of the test. The second section of the test was Spirometry which is used to measure the volume of air that the patient is able to expel from the lungs after maximum inspiration[38]. During this test, the patient is seated sitting upright with their lips securely wrapped around the mouthpiece and wearing a nose clip. They are then required to inhale to and exhale to the maximum of their ability. This is repeated until the patient has provided three sets of results with an exhale of at least six seconds. The Third section involved a lung volume test through the use of Helium Dilution. This method involves adding a known concentration of helium to the air in which the patient is to breath[39]. Initially, the patient is asked to breathe normally for at least two minutes. Once a normal breathing pattern has been presented, the patient is asked to inhale deeply, exhale steadily and fully; and resume normal breathing. The lung volume can then be calculated using the before and after helium concentration results, with the dead space of the valve and mouthpiece being taken into account. The fourth section included a test for the diffusing capacity of the lungs for carbon dioxide (DLCO). This is a very important test as it determines how well the lungs exchange gas. Much like Helium Dilution, DLCO involves introducing a known concentration of CO2 to the air in which the patient is breathing[40]. The patient is then asked to hold their breath, for a minimum of ten seconds, and then exhale fully. The diffusion capacity of the lungs is then calculated using the concentration of CO2 in the exhaled air against the known concentration initially introduced.

The combination of Spirometry, Helium Dilution, and Diffusion Capacity currently seems the best practice for testing pulmonary function testing. Each method provided information on primary lung functions. Combining the results of each approach provides data on the patient's lungs as a whole. Thus, providing respiratory experts with data required to come to a diagnosis.

4.4 Aisling McGowan Meeting 10/11/2015

This meeting included a discussion on the limitations of the sensors available for this project. Due to funds and other factors, this project is limited to reading ETCO2 and RR. However, it was identified that currently the first stage in the diagnosis of DB is a physical assessment of the patient's chest and abdominal breathing. Patients with normal breathing functionality breathe with their chest and abdomen equally. Physical assessments can lead to misdiagnosis as it is subjective. It was concluded that the diagnostic tool is to objectively measure abdominal and chest breathing rates, with the added dimension of ETCO2 levels to strengthen results. Depending on funding, effort bands or inductance bands will be utilized to detect these differences. The acquired data will be presented and stored in numeric and waveform format. In the event of significant difference displayed, the user will be required to participate in a breathing retraining session. This session will provide information to educate the user in addition to increasing DB awareness. It will also include breathing exercises to teach the user how to correct their DB and the importance of nasal breathing. Biofeedback

may be used in the process of teaching abdominal breathing. Finally, it will provide breathing control exercises for the patient to practice daily. Once complete, the user must practice the new skills for four weeks before returning to the program. After this period, the user is tested again. The baseline and post breathing retraining data are compared to register/confirm improvements. If improvements are identified, it provides a more accurate and objective result to the first stage of assessment, in relation to the subjective physical assessment.

Newly identified stages of system:

- Stage One: Diagnostic Tool
- Stage Two: Breathing Retraining
- Stage Three: Register/Confirm Improvements

4.5 Aisling McGowan Meeting 18/01/2016

This meeting included a continuation of the previous discussion on sensors and readings to be taken in this project. Aisling presented a range of sensor options which could be used in this project. However, special attention was paid to the Ambu as this company had supplied medical equipment to Connelly Hospital in the past. Ambu is a Danish company which develops and produces medical equipment specialising in products for respiratory support. The Ambu Sleepmate RIPmate™ Inductance Belts and Ambu Sleepmate Piezo were identified as products which were well suited to the needs of the project. Following this Aisling found a flaw in the projects method of measuring ETCO2. Aisling stated that ETCO2 is measured in pressure so some method of measuring the pressure of the expelled gas would also be necessary. Aisling suggested carrying on the project without measuring ETCO2 but instead provide the medical professional the ability to enter in ETCO2 values during the test. She explained this would be a viable option as most respiratory wards contain capnography equipment and the nose piece associated with the device could easily be worn along with the respiratory belts.

4.6 Damon Berry Meeting 11/02/2016

This meeting involved discussing possible methods of measuring respiratory effort. The first method considered involved the use of an accelerometer to measure respiratory effort by measuring the changes in acceleration and converting the change in acceleration to distance using the principle of acceleration equals distance over time. However, this method did not seem best suited as it would require multiple values being processed per second resulting in performance issues. Following this Damon asked for some time to do some research and said he would be back in contact soon.

Later on that day, an email was received from Damon suggesting a method[34] for assembling respiratory effort sensor belts. Upon inspection, this method proved to suitable for the project and an agreement to use this technique for the respiratory effort sensor was made with Damon. Damon kindly agreed to overlook the development process and provided access to the facilities of his lab.

4.7 Damon Berry Meeting 18/02/2016

To begin this meeting Damon presented and explained the different pieces of equipment, required in the building of the respiratory effort sensors, in the electrical engineering lab. Damon explained the importance of resistance and Ohms law in a circuit and informed me that neglecting this property in a circuit could result in hardware damage. Then a simple circuit using an Arduino and a breadboard with a 50Kohm potentiometer and two 220k resistors was put together to take an analog voltage reading through the analog port of the Arduino. However, when adjusting the resistance in the circuit using the potentiometer little change was being made to the output analog voltage. After inspecting and confirming the circuit was correctly assembled, the 220k resistors were replaced with 10k resistors and this resulted in changes in analog voltage proportional to the changes in resistance in the circuit while never reaching 0 or 1023 volts to preventing hardware damage. Damon explained that the potentiometer in the current circuit can be used to replicate the respiratory effort belts. When the respiratory effort sensor belts have been successfully developed, the potentiometer in this circuit can be replaced with the belt.

Next, Damon explained that in order to graph respiratory correctly the Nyquist Sampling Theorem would need to be used to determine a number of readings to be taken per second to successfully detect the necessary points/values in the user' s breathing. He also added that perhaps 10x sampling rate would be best suited as it would give the best results.

4.8 Damon Berry Meeting 22/02/2016

This meeting included a conversation about how to reduce mechanical noise which was negatively affecting the quality of the graph. Damon suggested twisting together the two cables which connect the conductive terminals at each end of the conductive cord to the Arduino. He explained that this reduces mechanical noise as mechanical noise is caused by magnetic fields and when the wires are twisted together parts of the signal are pointing in opposite directions which cause mechanical noise levels to be reduced.

4.9 Chest and Abdominal Respiratory Expansion Study

During the construction of both the chest and abdominal respiratory belts, a length of conductive rubber cord was required. This rubber cord is stretched as the user inhales and only has the capacity to stretch 70% of its length at rest. Any further expansion increases the risk of damage. As a result, the length of the conductive rubber cord needed to be determined. In order to achieve this, measurements for chest and abdominal expansion were taken from ten people breathing at rest. The participants tested varied in age, sex, height and weight. Such a varied test group was selected as the design for the respiratory belts needed to ensure durability and robustness. The results of the test are as follows.

Chest and Abdominal Respiratory Expansion Test		
Participant	Chest Expansion (inch)	Abdominal Expansion (inch)
P1	2	1.2
P2	1.8	1
P3	1	1
P4	2	2.1
P5	1	1
P6	1.5	2
P7	1.5	1
P8	1.6	1.2
P9	1	0.5
P10	1.5	1

Table 3 Respiratory Expansion Results

Table 3 displays the results of the chest and abdominal respiratory expansion test. The goal of this test is to determine the largest stretch the conductive rubber cord is likely to encounter. The largest stretch recorded in the study was 2.1". A recent article[41] gives strength to this results as it claims normal respiratory expansion at rest is in the range of 1.57-1.96". As a result of these findings, both conductive rubber cords needed to be cut to a length which allowed an expansion of at least 2.1" in its 70% stretch range. It was decided that the rubber cord lengths would be precisely cut to a length of 4" as this satisfied the 2.1" stretch requirement and provided the flexibility of another 0.7" before reaching its 70% stretch.

Before the test, each participant was given a consent form which they read, fully understood and signed. Please reference section 15.7 for copies of each signed consent form.

4.10 Aisling McGowan Meeting 15/03/16

This meeting began with a discussion on the patient information required on the "Create an account" page. Aisling stated that the patient's date of birth and address needed to be added for identification purposes. The information gathered from the name, address, and date of birth fields acts as a third id method for verifying a person's identification. Additionally, fields for the sex, current diagnosis, reason for test and medications needed to be added to the registration form. Finally, it was suggested that a body mass index should be calculated using the inputted weight and height values. Following this, a discussion began on the topic of the generated report. Aisling stated that the information entered during the account creation process needed to be displayed in the report. In addition to the user's personal information, a graph for each session of the respiratory test needed to be included in the generated report. Additionally, while examining the graphs Aisling was able to understand and explain the breathing pattern displayed. She stated that the graphs were sufficiently informative as the total lung capacity and the residual volume are synchronized and the data gather was from a steady state, meaning that the inhalation peaks and exhalation peaks did not vary by more than five percent. Finally, it was discovered that a wheeze is a symptom of dysfunctional breathing. This symptom needed to be added to the breathing retraining session.

4.11 Damian Bourke Meeting 05/04/16

This meeting involved presenting the final product to Damian. When presented with the generated report, Damian suggested presenting the overall differences as a percent difference value. He stated that this would be the easiest way for a person to understand that data displayed on the report.

4.12 Conclusion

This chapter presented findings from various methods of research including questionnaires, meetings, and studies. It is clear that more attention and awareness needs to be given to DB as 22% of participants obtained a score of twenty-three or higher. The information gathered during each of the meetings presented enabled the definition of the projects function requirements. Additionally, the data gathered from the chest and abdominal respiratory expansion test aided in the design of the respiratory belts. The next chapter will present the findings of the research phase and, in turn, define the system requirements.

5 Resultant Findings and Requirements

5.1 Introduction

The goal of this chapter is to present and describe the findings of the research phase and use these findings to define the system requirements.

5.2 Findings

Throughout the research phase much has been discovered and changed. Breathing Pattern Disorders (BPD) was the original focus of the project. However, following the Connolly Hospital visit this approach was discouraged due to workload and time factors. BPD is a very broad term which describes a wide variety of lung disease and illnesses. The project's focus then changed to DB as it is related to BPD but is more specific. After much research, the difficulty in DB diagnosis was discovered. This difficulty was discussed with Aisling McGowan during our meeting. In light of the evidence suggesting large quantities of patients with DB are overdiagnosed it was concluded an objective method of detecting abdominal and chest breathing differences was required. The current approach of identifying differences in chest and abdominal breathing involves a physical assessment carried out by a doctor. This was recognized as a potential source of misdiagnosis due to its subjective nature. The importance and benefits of using real-time graphs to present data to the medical professional were explained. These graphs needed to be included in the generated report for examination. These observations gave direction and purpose to the project.

The importance of breathing retraining was discovered. From examining previous studies[1][21], it has been voiced that breathing retraining may help DB patients and there has been evidence that it does, in fact, reduce symptoms. However, there has not been a sufficient amount studies in this area to provide solid evidence of its benefits to DB patients. It has been identified as an area with potential which requires further testing. The NQ is typically used to measure benefits from the breathing retraining sessions and will be implemented in this project.

Vast research and questionnaire results have proven the importance of the NQ in this project. With DB patients presenting lower questionnaire scores post breathing retraining and with 22% of participants in the issued questionnaire indicating positive for hyperventilation it has proven its worth to the DB research community. Much like other studies, this project will use the NQ as a measurement for the benefits of breathing retraining.

The information gathered throughout the research phase has enabled the definition of functional and non-functional requirements. A combination of information obtained from previous studies and Aisling McGowan shaped the end to end process of the program and identified the functional requirements. Through the evaluation of similar systems non-functional requirements were derived. From this study, the best aspects of each system can be observed and implemented in this project. Resulting requirements are presented below.

5.3 Requirements

Functional:

- Create and log in to an account.
- Complete and view results of NQ.
- Record chest and abdominal respiratory expansion data and input ETCO2 values.
- Participate and complete breathing retraining sessions.
- View and store numeric and waveform data in real time.
- Register/confirm improvement.
- Generate, view and print the diagnostic report.
- View help and documentation.

Non-Functional:

- Usability
- Performance
- Reliability
- Accuracy and Precision
- Security
- Documentation
- Portability

Another design feature which became evident throughout this research phase is how the system will cater for disabilities. DB occurs more frequently in older patients so evidently catering to disabilities will be extremely important. Examples of disabilities in which the system potentially will have to adapt to are as follows:

- Bad eyesight.
- Difficulty in staying concentrated.
- Short term memory problems.
- Colour blindness.
- Trembling hands.

5.4 Conclusion

The results of the research phase presented above successfully provided sufficient information to enable the defining of functional and non-functional requirements. In addition to the non-functional requirements, special effort will be made to facilitate the needs of user's which suffer from common illnesses found in older patients. The next chapter will analyse and explain the different stages involved in the Java application.

6 Analysis

6.1 Introduction

This chapter aims to define the stages and environment associated with the system. Each stage of the Java application will be explained while stating the required user state and other external factors which may affect the system.

6.2 Analysis

This system is to be used by the patient in a respiratory professional's office. The respiratory professional must be present to supervise the process but must provide sufficient privacy so that the information provided by the patient is not influenced by their presence.

The first stage in the solution requires the patient to create an account providing necessary personal information. This information will be presented in the generate report and used to aid the medical professional in the diagnosis process. The user account also acts as a security layer in the system to help protect user's personal information. The respiratory professional should be available to answer questions or to measure patient's weight or height if required.

The second stage involves the patient completing an NQ. This is a screening tool used to identify hyperventilation syndrome-related symptoms being experienced by the user. The results of this stage act as a measure of improvement between baseline and post breathing retraining data. Additionally, NQ results indicate if symptoms related to DB are present or not. The patient should be given privacy will completing this questionnaire so results are not hindered.

The third stage is for use by the respiratory professional and makes use of the chest and abdominal respiratory belts. The medical professional is presented with a video guide that informs them how to correctly apply the equipment. Once informed the medical professional applies the chest and abdominal bands to the patient as specified. In the event of no recent ETCO2 value being entered during the registration process, the respiratory professional uses capnography equipment to measure the user's ETCO2 throughout the test. The patient is then checked to confirm that they are rested and that their breathing is not being affected by factors such as fatigue. Once the apparatus is correctly setup and the patient comfortable and breathing at rest, the medical professional triggers the test to start. The test runs for the duration of fifteen minutes but data is only collected for the last ten minutes. This ensures that the patient is breathing naturally at rest and data is collected from a steady state. A steady state refers to a duration of time where the variation of the patient's respiration amplitude is less than 5%. This test will take readings of chest and abdominal expansion and RR. The data will be presented in real-time in the form of a line graph throughout the test. This displayed data is solely for the attention of the medical professional as patients subconsciously control their breathing while watching results. Once the test is complete all numeric and graphical data will be saved for later use.

The fourth stage is determined by the results of the second and third stages. The respiratory professional examines the report generated during the respiratory test. The results of these two stages are then clearly explained to the patient. If a patient obtains any of the below results then they will be advised to advance to the next stage.

- Medium to high NQ score.
- High RR
- Low ETCO2 level.
- A significant difference between chest and abdominal expansion.

If the user does not present any of the previous criteria then they can choose if they would like to progress further to the breathing retraining session.

The fifth stage involves the patient regaining control of the system and consists of a breathing retraining session which will last for a period of approximately 15 minutes. The aim of the session is the teach users how to perform PLB and abdominal breathing. It will consist of the following:

- The user will be asked to sit comfortably, with their knees bent and shoulders, head, and neck relaxed[42].
- Information will then be presented to the user aimed at bettering their understanding of the condition, effects of breathing pattern on symptoms and potential benefits of breathing retraining.
- A combination of textual descriptions, diagrams, and step by step video tutorials will be used to teach the user how to perform PLB and abdominal breathing.
- The user will be given exercises to practice in their day to day life and are advised to perform these exercises in different situations twice a day minimum.
- These new skills should be practiced for approximately four weeks before returning for the second phase of testing.

The sixth stage is the second phase of testing. The user is required to carry out the NQ and respiratory test in the exact same manner as before.

The seventh stage gathers the baseline data and post breathing retraining data and then processes it using the assessment algorithm. Once the data has been processed, the numeric and graphical data is passed to the report generation class. In this class, the data is used to generate a report. This report includes the benefits, if any, of the breathing retraining in regards to Nijmegen scores and the differences in chest and abdominal expansion and RR. This report is ready to print and help aid further diagnosis.

6.3 Conclusion

The analysis chapter clearly explained the process for each of the stages involved in the system. The variables and external factors which may affect the completion of specified stages have been stated and precautions have been made to prevent hindrances. The next chapter will define the software development methodology used and provide reasoning for the choice.

7 Approach and Methodology

7.1 Introduction

The aim of the approach and methodology chapter is to define the software development methodology and the end users which will be used throughout the project.

7.2 Methodology

The Agile software development methodology will be utilized in this project. This methodology has been chosen for several reasons. Being an individual project, the agile methodology was appealing due to its suitability for small team sizes. Two different types of agile development methodologies will be used throughout this project. The first form does not include coding and will be used in the research, design, and planning stage. The information gathered in these stages is testing by asking the project supervisor and the specified users. The second form will start once the initial documentation is complete. In this form design, code and testing will be completed using sprints. Sprints provide the advantage of testing and improving functionalities throughout the development of the project. This is essential as user and project needs change frequently. After each sprint, the developed functionalities will be evaluated by specified users. Please find the specified end-users below.

End-Users

- William Fitzpatrick: novice computer skills, no scientific background.
- Ross McCann: intermediate computer and science skills, UCD science graduate.
- Aisling McGowan: advanced science skills, Respiratory professional.

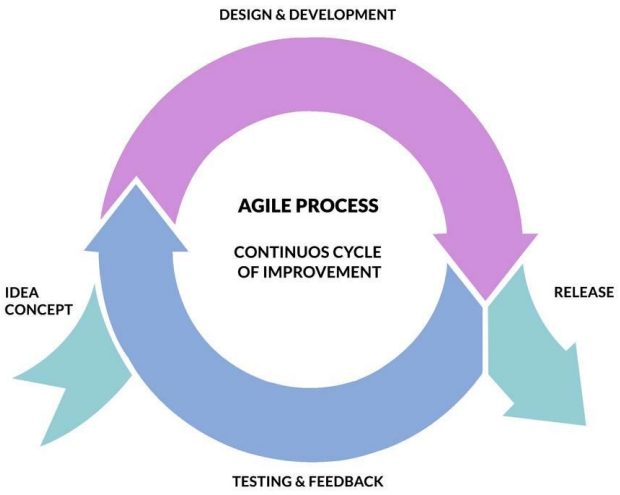


Figure 7-1 Agile Process[43]

Figure 7-1 displays the agile method which is being used in this project. The functionalities will be evaluated by the users through the use of usability testing, which will be explained later in further detail. The feedback is used to make the required changes to the design and

functionalities. Additional types of system testing are then carried out. Once all the functionalities have been developed, evaluated, changed and tested the next sprint can begin. This process is repeated until all functionalities are complete to a standard which satisfies the users.

7.3 Conclusion

The agile software development methodology was chosen due to factors such as the small team size and the inevitability of future user and project requirement changes. Additionally, a set of three end users has been specified. Each user in this set has been selected to create a varied group of end users which represents each sub-section of the target audience. This next chapter is the design chapter which will present that design diagrams developed for the project.

8 Design

8.1 Introduction

The goal of the design chapter is to develop easy to understand designs which simplify the implementation process. These designs will aid the development of the Java application and database as they are referenced throughout.

8.2 Technical architecture diagram:

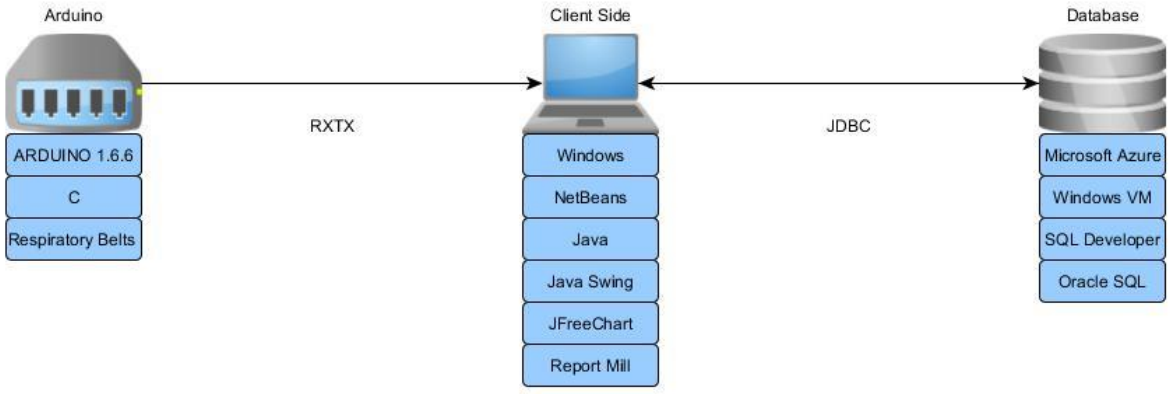


Figure 8-1 Technical Architecture Diagram

Figure 8-1 shows the system architecture for the project. The systems architecture consists of a client-side application, an Arduino system, and a database. Each will be described in further detail below.

The client side application is to be developed using the Java programming language in conjunction with the NetBeans compiler. The Java Swing library is used to create the GUI. Real-time graphs will be built, updated and stored locally by the JFreeChart library. The Report Mill library is used to generate the final report once the final stage of the program has been completed. The RXTX library is used to enable communications between the Java application and the Arduino. In this case, the Java application will use the RXTX library to read data via the serial port. The JDBC library allows communications between the Java application and the external database to exist. In this case, it is used to push and pull data to and from the database.

The Arduino system is programmed using the C programming language within the Arduino 1.6.6 compiler. It will interact with two respiratory belts, one chest and one abdominal. This Arduino system is connected to the computer running the client side application using a USB serial port.

The external Oracle database will be located on a Windows based Microsoft Azure virtual machine. The database will be developed using the Oracle SQL language through the Oracle SQL Developer program. All table creations, insert, update, and deletes will be accommodated by SQL.

8.3 Class Diagram

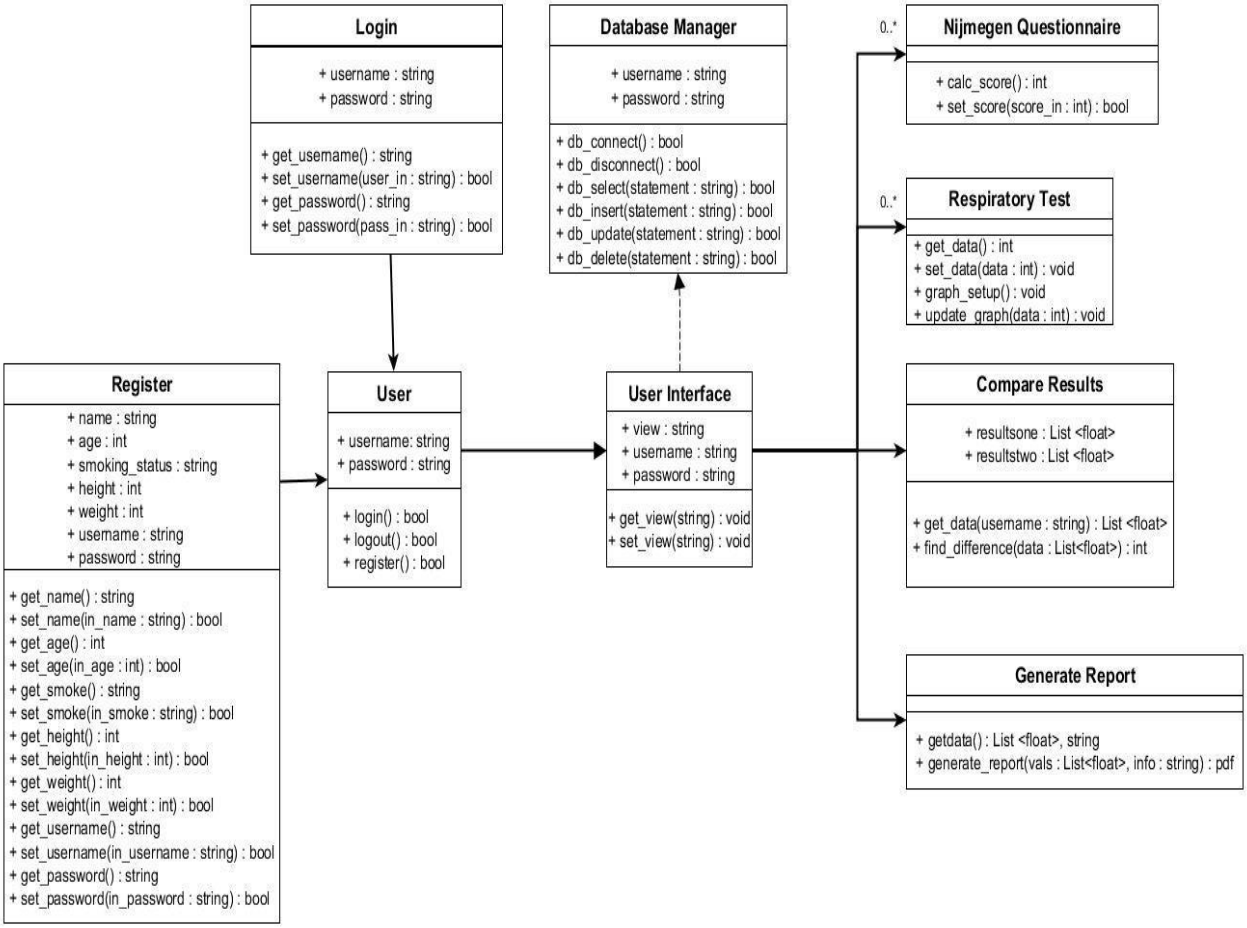


Figure 8-2 Class Diagram

Figure 8-2 is the class diagram which was created for this project to reference during the development phase. The purpose of the class diagram is to specify the structure of how a software system will be written and function, without actually writing the complete implementation[44]. Each future class to be implemented is displayed with the expected variables and functions which will be required within the class.

8.4 Sequence Diagram

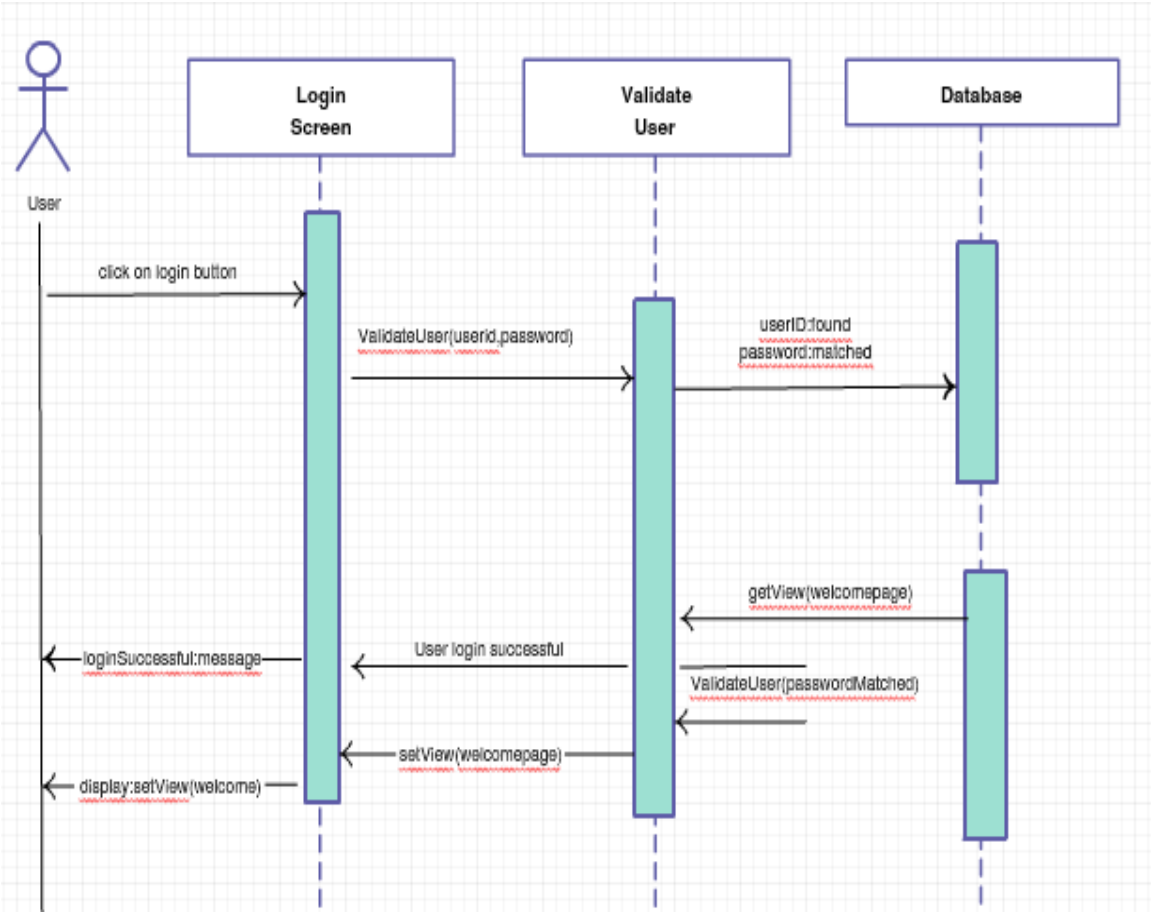


Figure 8-3 Login Sequence Diagram

Sequence Diagrams are interaction diagrams that detail how operations are carried out[45]. Figure 8-3 is an example of a sequence diagram for user login. Firstly the user clicks the login page button to direct him/her to the login page. The user then enters their login details and the system then validates the user details. This is done by querying the database. When the details are successfully validated the view is set to the welcome page and a confirmation message is displayed.

8.5 Use Case Diagram

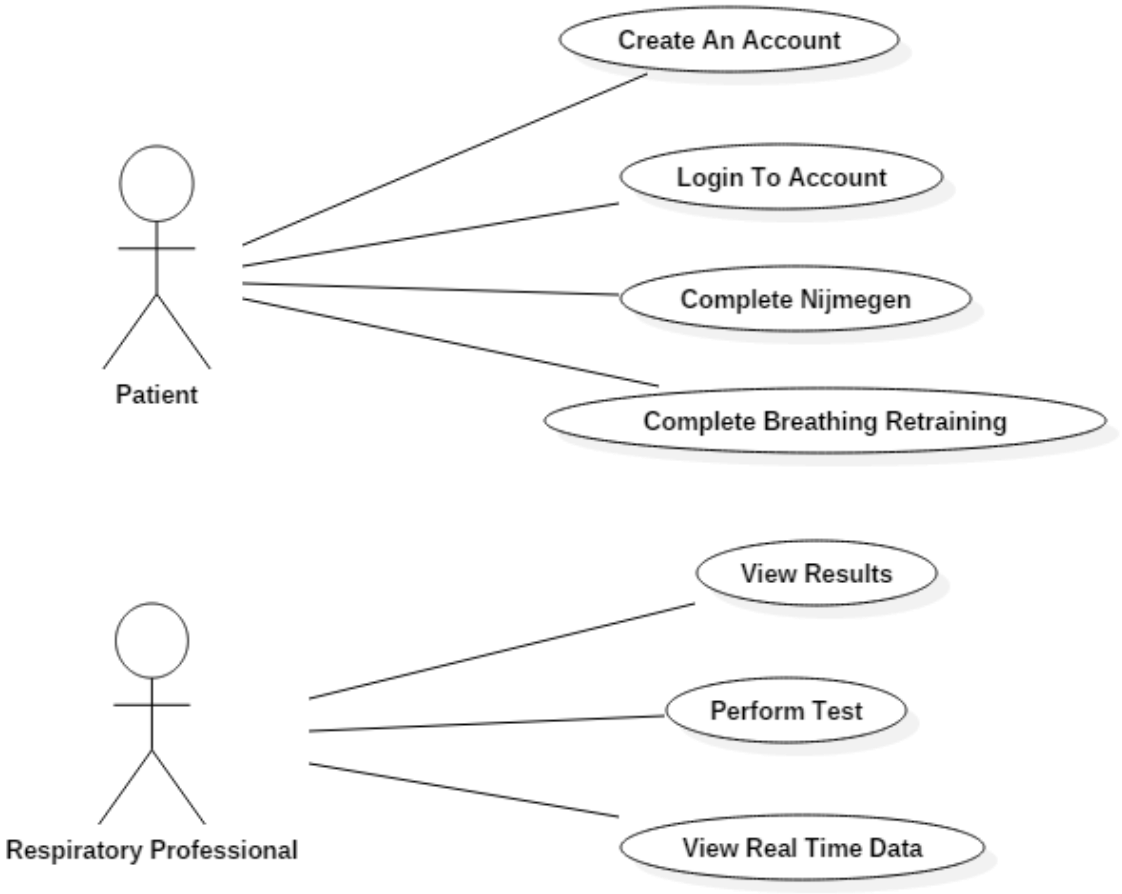


Figure 8-4 Use Case Diagram

Figure 8-4 shows the users interaction with the system through a use case diagram. The patient performs the tasks of creating an account, logging into their account, completing the Nijmegen questionnaire and completing the breathing retraining session independently, unless they require the respiratory professional's assistance with a task such as measuring their weight or height. It is important the patient performs these tasks on their own so that the data they provide is not influenced. The respiratory professional performs the tasks of beginning the respiratory test, viewing the real-time data during the duration of the test and views the results in the generated report. Much like the patient, it is also very important that the respiratory professional performs their tasks independently. For example, if the patient was to view the real-time data during the test they would subconsciously try and breathe differently to make the graphed results look better. This is a known behaviour while watching results and it is common practice to hide data from patients during examinations.

8.6 Entity Relationship Diagram

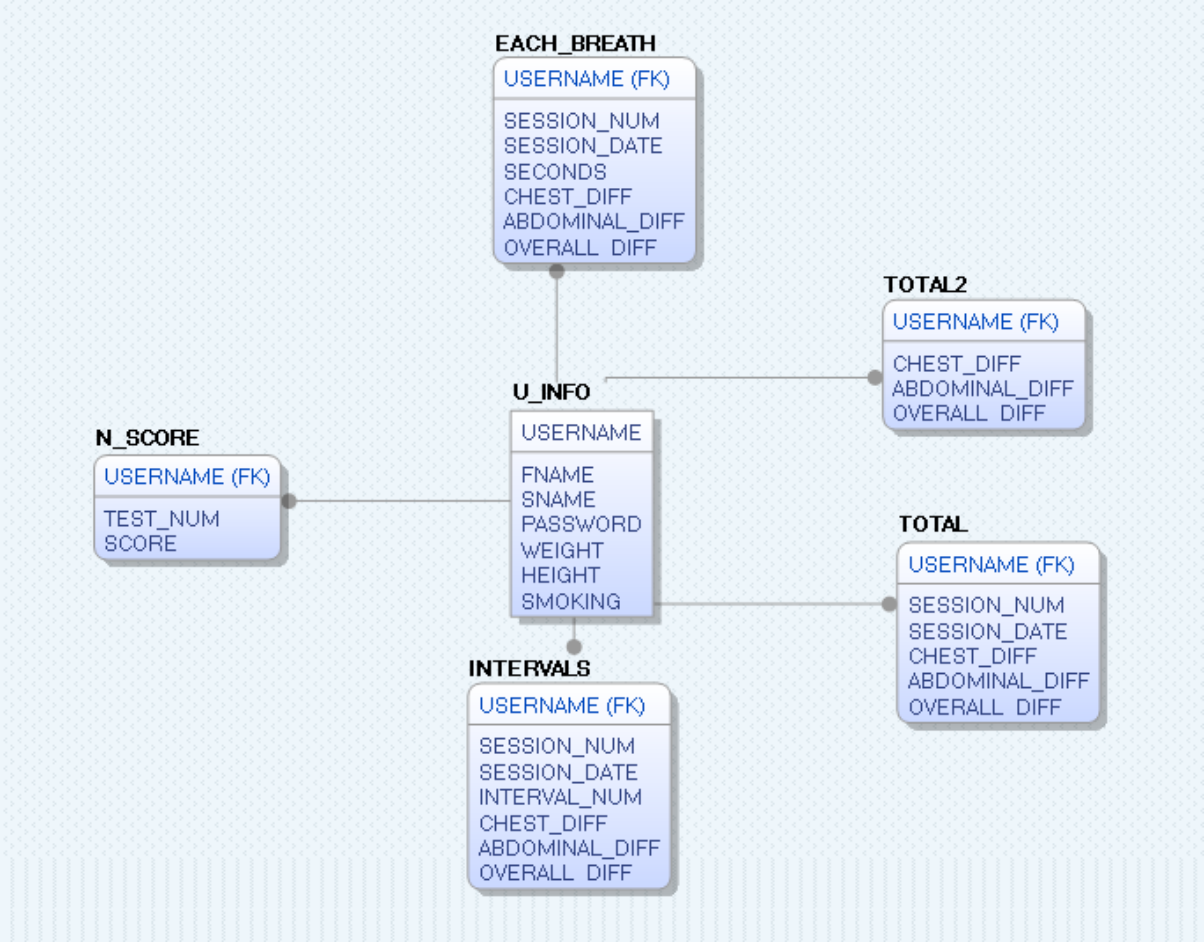


Figure 8-5 Database Entity Relationship Diagram (ERD)

The ERD in Figure 8-5 displays the design of the database which will be used to store the data generated by the program. The "U_INFO" table will be used to store the data obtained during the account creation process and acts as the parent tables to each of five other tables. The "N_SCORE" will be used to store the results of the Nijmegen question obtained by the user. The "EACH_BREATH" table was designed to store the data generated for each breath during the respiratory test. The "INTERVALS" table will store the average data for each of the ten one minute intervals in the respiratory test. The "TOTAL" table will be used to store the average data for the overall session. Finally, the "TOTAL2"table will be used to store the data calculated when comparing the baseline and post breathing retraining session respiratory test data.

8.7 Conclusion

Several designs which are easy to understand have been developed. The different elements of the system and how they interact with each other is displayed in a simple format by the technical architecture diagram. The class diagram presents how to software will be written and structured and will be reference throughout the implementation phase. The sequence diagram displays the sequence of operations which will need to be implemented to develop the login functionality. The use case diagram demonstrates the system's two users and the

tasks each of them performs. Finally, the database design clearly shows how to create the database with details down to the primary and foreign keys. The next chapter is the prototyping and development chapter and it will cover stages involved in the implementation of the system while following the agile software methodology.

9 Prototyping and Development

9.1 Introduction

The prototyping and development chapter aims to describe the stages and prototypes involved in the development of the different technologies used within the system. As per the agile software development methodology, functionalities will be developed in sprints with testing being performed and the end of each sprint.

9.2 Virtual Machine Set-Up

The Azure cloud service was utilised to create a Windows Data Centre virtual machine (VM). Oracle 11g XE was installed on the VM and used to implement the previously described database design, thus creating a web server. Once the firewall settings and VM endpoints had been successfully configured the Java program gained the ability to interact with the database remotely using the JDBC library. The Azure cloud service was used to host the VM for many reasons. Microsoft Azure ensures an impressive 99.95% monthly uptime for VMs hosted on their cloud service[46]. Additionally, Azure by default provides a wide array of security measures such as data encryption and a distributed denial-of-service defence system which provides a high level of security for VMs deployed[47].

9.3 Java Application

This section will describe the stages and prototypes involved in the design and development of the Java application's GUI.

9.3.1 Initial Login and Register Page Prototypes

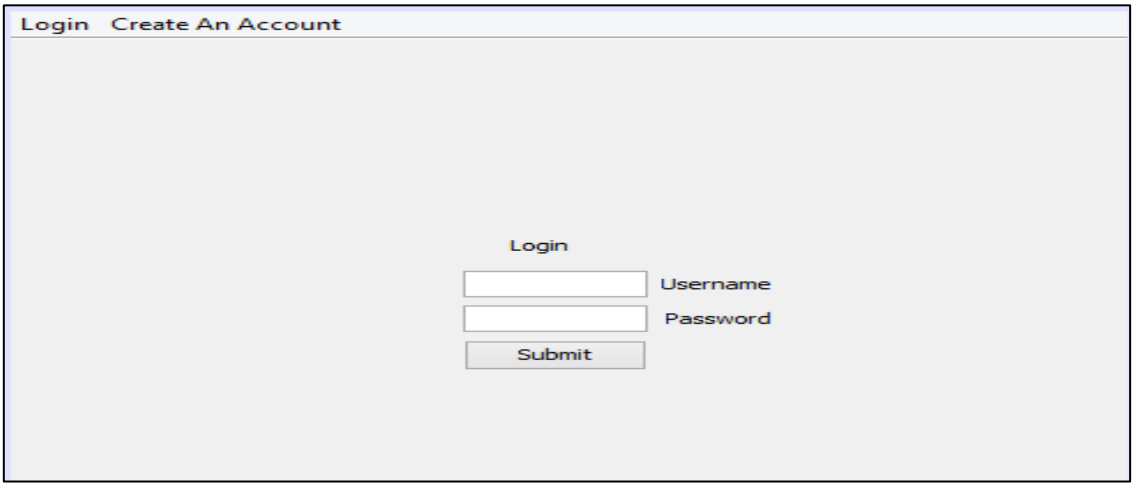


Figure 9-1 Login Page Prototype

Figure 9-1 displays the "login page" prototype developed in Java for this project. It includes a simple login form which requires the user to enter their username and password in order to log in to their account. In the case of a new user, clicking the Create an Account option in the menu bar located at the top of the screen will direct the user to the registry page.

```
public class DBman {
    // JDBC driver name and database URL
    static final String JDBC_DRIVER = "oracle.jdbc.driver.OracleDriver";
    static final String DB_URL =
    "jdbc:oracle:thin:@fypvm2.cloudapp.net:1521:xe";

    //  Database credentials
    static final String USER = "SYSTEM";
    static final String PASS = "MyPassword123";

    static Connection conn = null;


    public static Connection main(){
        // verifying jdbc driver
        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");
        } catch (ClassNotFoundException e) {
            // catch block
            e.printStackTrace();
        }

        // Open a connection
        System.out.println("Connecting to database...");
        try {
            // Creating connection
            conn = DriverManager.getConnection(DB_URL,USER,PASS);
            System.out.println("Connected!!!");
        } catch (SQLException e) {
            // catch block
            e.printStackTrace();
        }
    return conn; // return connection
    }
```

Figure 9-2 Login Page Prototype

In order to verify the submitted data a connection must be made to the database containing the data and then this database needs to be queried. The JDBC library has been utilized to provide database connectivity and interaction capabilities to the Java program. Figure 9-2 presents the database manager class developed to establish a connection to the specified database. Firstly the JDBC driver is checked to confirm that it is correct and available for use. The web server's cloud app URL is used to connect to the database at the SYSTEM user to ensure sufficient privileges.

```
// Create SQL query strings
sql = "SELECT * FROM U_INFO WHERE USERNAME = '"+ username +"'";
sql2 = "SELECT * FROM U_INFO WHERE U_PASSWORD = '"+ password +"'AND
USERNAME = '"+ username +"'";
conn = DBman.main(); // Connect to database

//Execute a query
System.out.println("Creating statement...");
try {
    // Creating statements for both SQL query strings
    stmt = conn.createStatement();
    stmt2 = conn.createStatement();
} catch (SQLException e) {
    // catch block
    e.printStackTrace();
}
try {
// Execute queries and stored results
result = stmt.executeQuery(sql);
result2 = stmt2.executeQuery(sql2);

} catch (SQLException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

Figure 9-3 Login Form Database Query Code

Figure 9-3 displays the code associated with the login form which was developed to connect and query the database using the user submitted values for the username and password fields. Firstly, two SQL queries are created, using the submitted data, so that informative feedback can be displayed to the user in the event of login failure or success. Next, a connection to the database is established and a statement for each SQL query is created using the database connection. Both statements are executed and the corresponding results are stored in ResultSet variables.

```
try {
    if (result.next()){// Check if result was returned
        if (result2.next()){
            correctusername = username;
            JOptionPane.showMessageDialog(null,"You Have Successfully
Signed In");
            usernameVar.setText("");// reset fields
            passwordVar.setText("");
            setVisible(false);
            dispose();
            nijimegen.main(correctusername); // change view
        }
        else {// if password is incorrect
            JOptionPane.showMessageDialog(null,"The Password You Have
Entered Is Incorrect. Please Try Again");
            usernameVar.setText("");
            passwordVar.setText("");
        }
    }
    else {// if username is incorrect
        JOptionPane.showMessageDialog(null,"There Is No User Registered
With This Username. Please Try Again Or Create An Account");
        usernameVar.setText("");
        passwordVar.setText("");
```

```
    }
} catch (SQLException ex) {// catch sql exception
    Logger.getLogger(FYPGUI.class.getName()).log(Level.SEVERE, null, ex);
}

try {
    stmt.close();// close statements
    stmt2.close();
} catch (SQLException e) {
}
try {
    conn.close();// close connection
} catch (SQLException e) {
}
```

Figure 9-4 Login Form Data Verification Code

Figure 9-4 shows the code developed to handle the results of the two previously executed SQL queries. A successful login occurs when both result sets contain a single row which indicates that both the username exists and the password entered corresponds to the username entered. In the event of the first result set containing no data, a message is displayed to the user informing them that the username entered does not exist in the database. In the case of the first result set containing data and the second result set containing no data, a message indicating that the password entered is incorrect is presented to the user. This message is given to the user as the username has been confirmed to be a username stored in the database but the password given is not the same as the password associated with the username stored in the database.

Unit testing was undertaken on the login class, please reference section 10.5.1, to confirm that no bugs exist in the code and no unexpected events occur. This unit test was a success as each test in the unit test case passed.

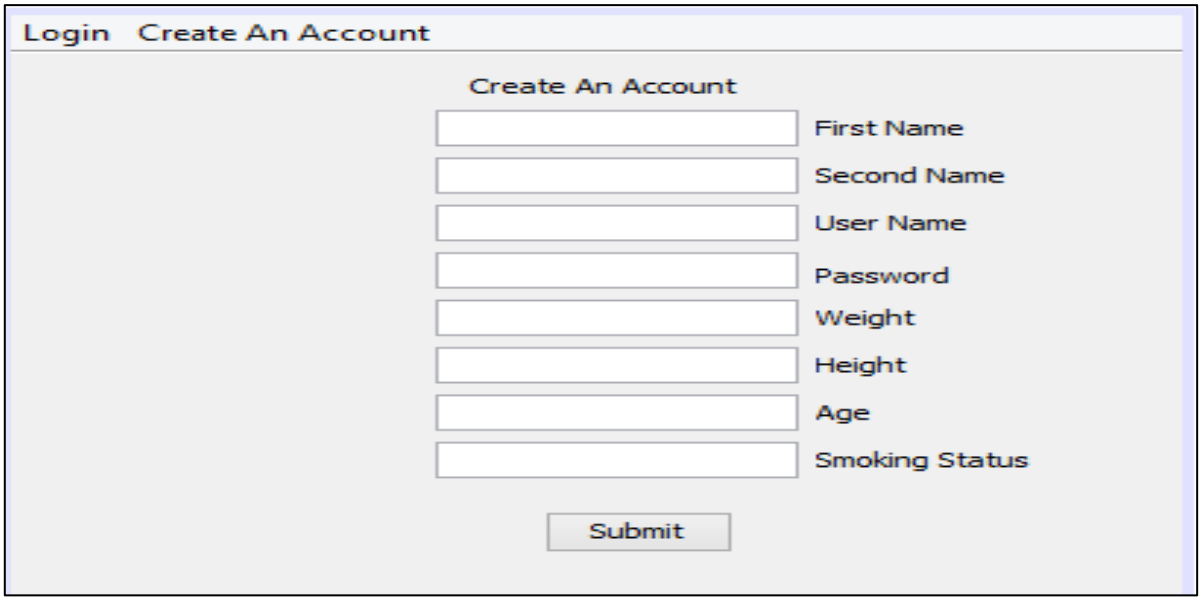


Figure 9-5 Create an Account Page Prototype

Figure 9-5 shows the first prototype for the “Create An Account” page which will be used to obtain user sign-in information and other data which may affect the results if the respiratory test.

```
sql = "INSERT INTO U_INFO VALUES('"+ firstname +"','"+ secondname +"','"+
password +"','"+ weight +"','"+ height +"','"+ age +"','"+ smoker +"')";
conn = DBman.main(); // establish connection to database
try {
    stmt = conn.createStatement(); // create statement
} catch (SQLException e) {// catch block
      e.printStackTrace();
}
try {
    result = stmt.executeUpdate(sql); // execute query and store results
} catch (SQLException e) {
    e.printStackTrace();
}
try { // clean-up environment
    stmt.close();
} catch (SQLException e) {
}
try {
    conn.close();
} catch (SQLException e) {
}

if (result > 0){ // if successful display message and direct to login page
    JOptionPane.showMessageDialog(null,"You Have Successfully Created An
Account");
    setVisible(false);
    dispose();
    new FYPGUI().setVisible(true);
}
else{ // else display error message
    JOptionPane.showMessageDialog(null,"An Error Has Occurred. Please Check
Internet Connection And Try Again");
}
```

Figure 9-6 Registration Page Code

Figure 9-6 presents the code developed for the “Create an Account” page’s registration form. Much like the login form, the JDBC library has been used to connect the registration form to the database. However, in this case the executeUpdate() function is used as data is being inserted into the database. This function returns one if successful and zero when unsuccessful and stores it in the result variable. Next, if the result value contains a positive result a confirmation message is presented to the user and they are then redirected to the login page. In the case of the insert failing, the user is informed of the failure and asked to check their internet connection and try again.

Extensive data validation methods occur prior to the code displayed in figure 9-6. A unit test case was developed to test the data validation methods implemented, reference section 13.2. The results of the unit test were mostly positive with only a small number of failures. A method to fix the problem was decided through examining the failures, please reference section 10.5.2 for further details.

The execution times of each of the functionalities implemented in this sprint were tested using performance testing, reference section 10.6.1. This test case was a success as each test passed.

9.3.2 Improvements and Addition of Nijmegen Questionnaire Page

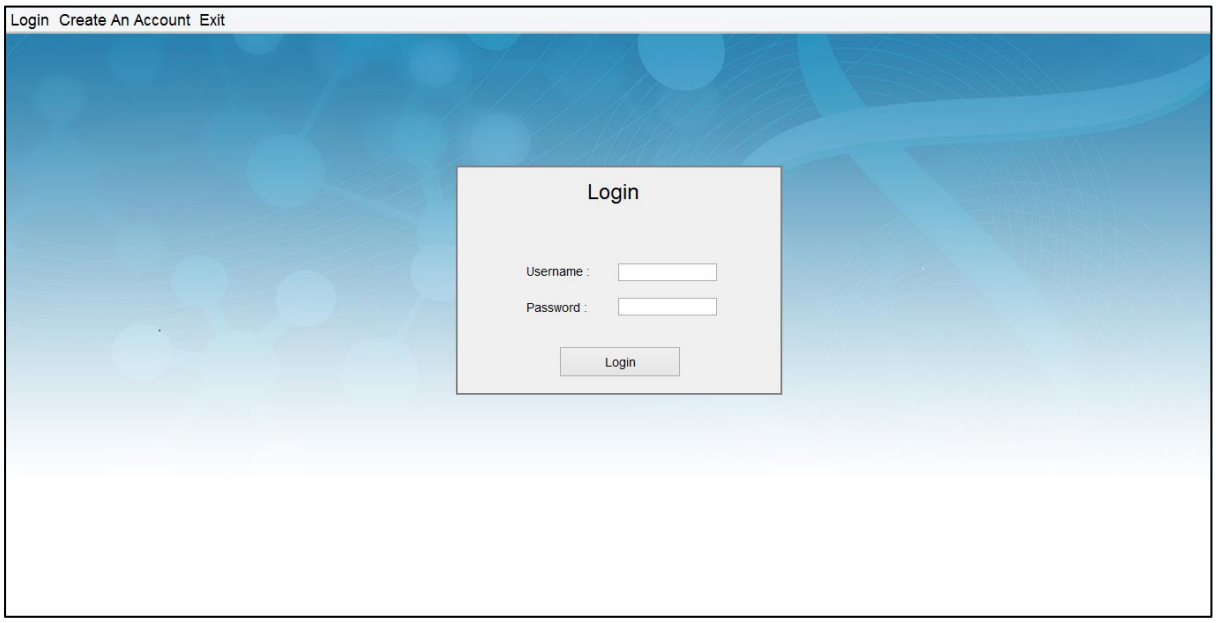


Figure 9-7 Login Page Prototype Two

Figure 9-7 displays the second prototype for the login page. Many changes have been made, when compared to the previous prototype, based on the heuristics brought to attention from the usability test. Please reference section 10.4.2 for details on the usability test. The actions taken have been implemented to try to fulfill the needs of each user.

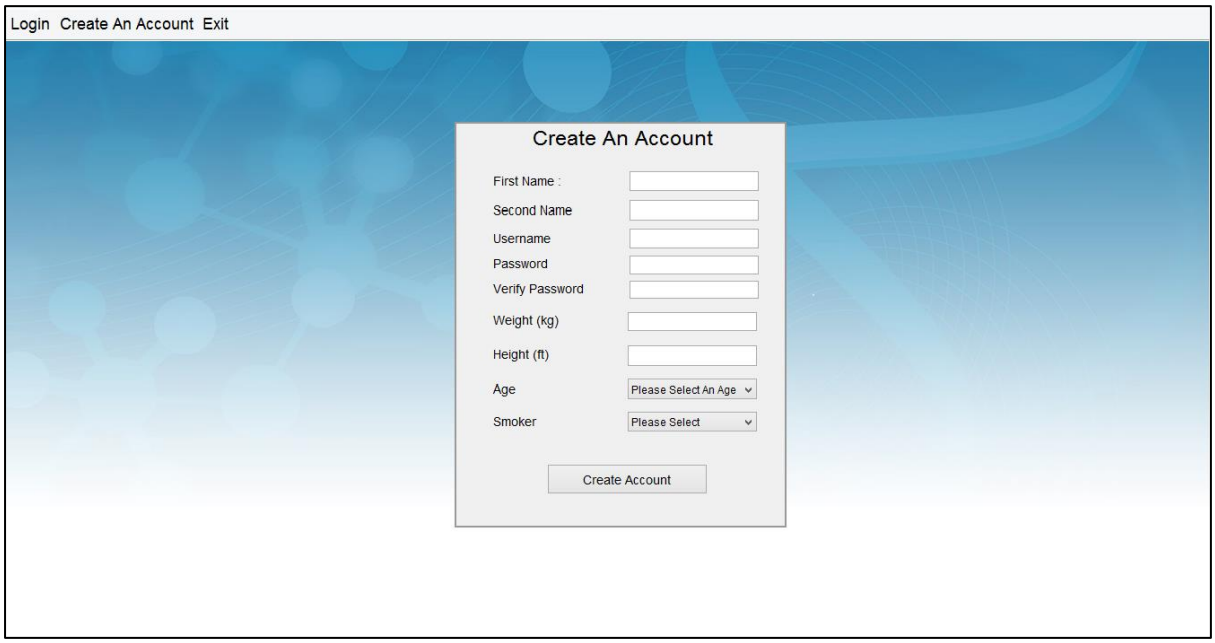


Figure 9-8 Create An Account Page Prototype Two

Figure 9-8 demonstrations the second prototype for the “Create An Account” page. Multiple changes have also been made to this page as a result of usability testing. Please reference section 10.4.2 for details on the usability test. These changes have been made in an attempt to make the registration process as easy as possible for the user. The verify password field has been added to prevent the user from entering a different password than desired. Additionally, units of measurement have been added to the label for the weight and height text fields to keep data consistent and prevent confusion. Finally, pre-defined options are given for both the age and smoking status fields to further simplify the process.

The actions decided from the results of the previous unit test were implemented to try to resolve the encountered problems. A unit test case was then developed to test the newly added data validation methods and features implemented as a result of usability testing, please reference section 13.3. This unit testing phase was a success as all tests in the test case passed.

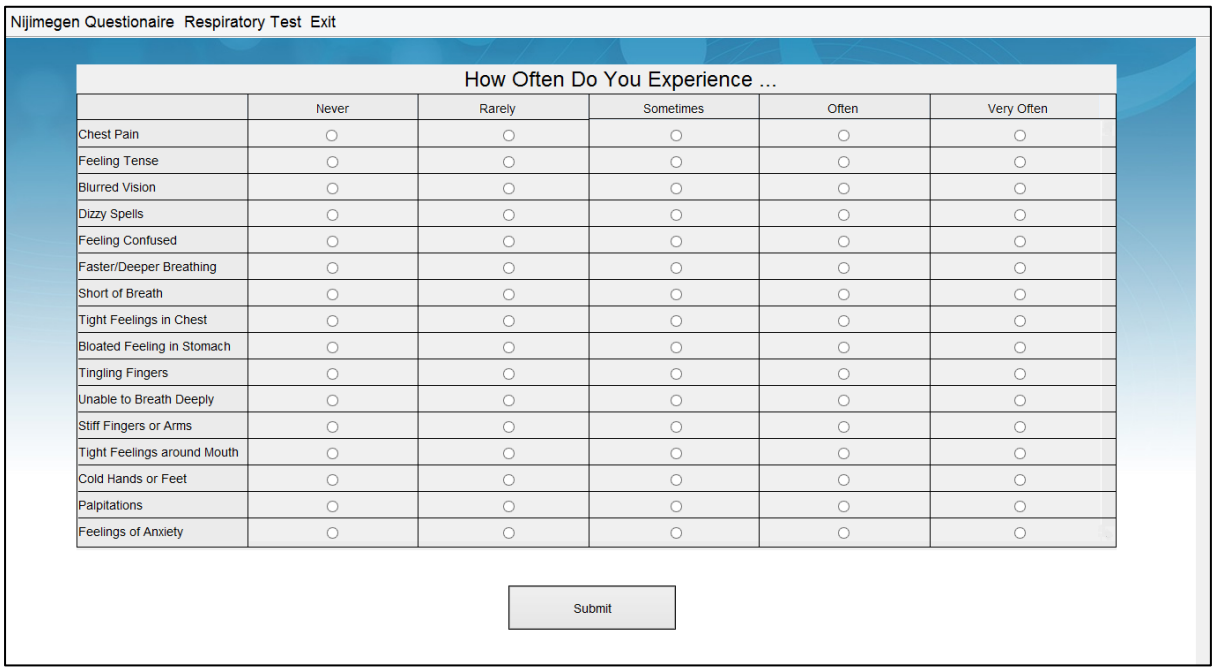


Figure 9-9 Nijmegen Questionnaire Page

Figure 9-9 displays the newly added “Nijmegen Questionnaire” page. The user is required to use the radio buttons to enter a score, from never to always, for how often they experience the specified symptoms. While developing the “Nijmegen Questionnaire” page the low scoring heuristics identified in the “login page” and “registration” page usability test were kept in mind as to prevent repeating mistakes.

```
if (buttonGroup1.getSelection()==null || buttonGroup2.getSelection()==null
|| buttonGroup3.getSelection()==null
    || buttonGroup4.getSelection()==null ||
buttonGroup5.getSelection()==null || buttonGroup6.getSelection()==null
        || buttonGroup7.getSelection()==null ||
buttonGroup8.getSelection()==null || buttonGroup9.getSelection()==null
            || buttonGroup10.getSelection()==null ||
buttonGroup11.getSelection()==null || buttonGroup12.getSelection()==null
                || buttonGroup13.getSelection()==null ||
buttonGroup14.getSelection()==null || buttonGroup15.getSelection()==null
                    || buttonGroup16.getSelection()==null){
    JOptionPane.showMessageDialog(null,"Please select an answer for all
questions"); // Check if option for each button group is selected
}
else { // add all scores together
    int finalResult = result1 + result2 + result3 + result4 + result5 +
result6 + result7 +
    result8 + result9 + result10 + result11 + result12 + result13+ result14
+ result15 + result16;
```

Figure 9-10 Nijmegen Questionnaire Code

Figure 9-10 demonstrations the code implemented to calculate the user's Nijmegen questionnaire score. Every radio button in each button group is given an on click method which enters a value into a global variable associated with that button group. When the submit button is clicked, each button group is checked to verify an option has been selected by the user. When an option has not been selected for any of the questions, a message is presented to the user informing them to ensure that they select an answer for all the questions provided. When an option has been selected for every button group, the scores from each question are added together to produce the final Nijmegen questionnaire score. This value along with the user's username and the session number are then inserted into the appropriate table in the database using the same technique as the registration form.

Each of the functionalities implemented in this sprint were tested using performance testing, reference section 10.6.2. All functionalities executed fully within the allowed time and passed the test.

9.3.3 Addition of Respiratory Test Page

Figure 9-11 Respiratory Test Page Prototype One

Figure 9-11 displays the initial prototype for the "Respiratory Test" page. This page is designed for use by the respiratory professional. When the respiratory professional has successfully prepared the patient for the test they click the "start" button which begins the respiratory test.

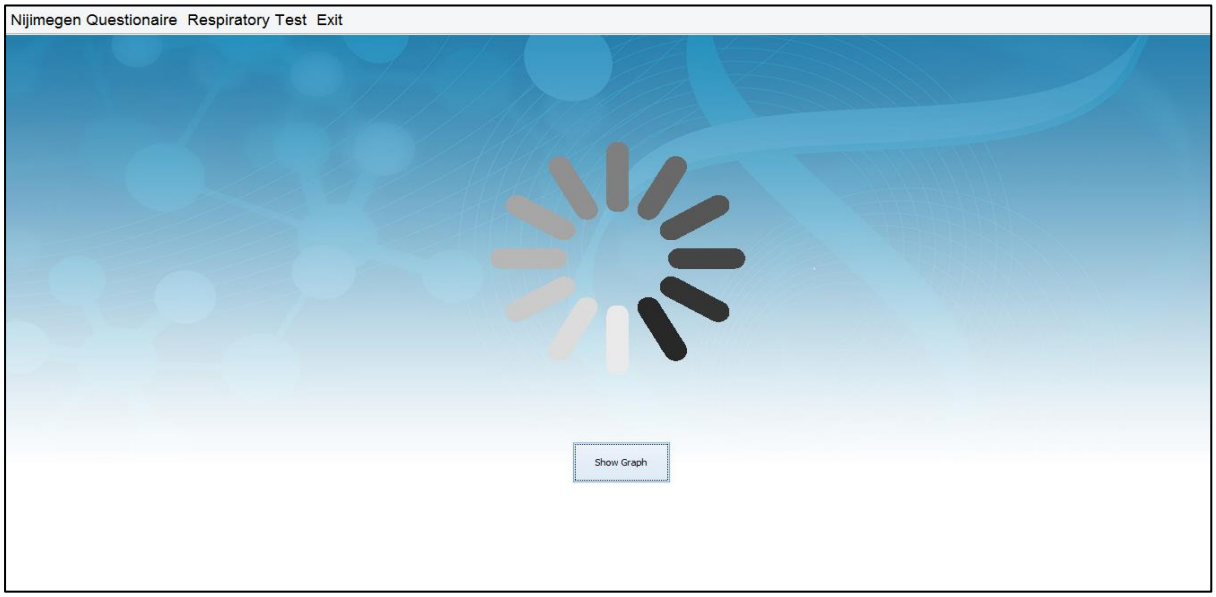


Figure 9-12 Respiratory Test Page Prototype One

Figure 9-12 shows the change to the "Respiratory Test" page in the event of the respiratory test beginning. The start button is replaced with a loading style gif to inform the respiratory professional that the test is in progress. The respiratory professional can now select the "Show Graph" button to display the graph which is plotting the chest and abdominal respiratory data.

```
private void initializeSerial()
{
    CommPortIdentifier portId = null;
    Enumeration portEnum = CommPortIdentifier.getPortIdentifiers();// store
all ports

    while(portEnum.hasMoreElements()) // scan through all ports
    {
        CommPortIdentifier currentPortIdentifier =
(CommPortIdentifier)portEnum.nextElement(); // scan through all ports
        for(String portName : PORT_NAMES)// if port is equal to any of the
ports stored in variable
        {
            if(currentPortIdentifier.getName().equals(portName))
            {
                portId = currentPortIdentifier; // store port id
                break;
            }
        }
    }

    if(portId == null)
    {
        JOptionPane.showMessageDialog(null,"Port not found. Please check
        your Ardunio is connected to the computer using a USB");
        respiratorytest.main(gusername);
        return;
    }

    try {

        serialPort = (SerialPort)portId.open(this.getClass().getName(),
        2000); // open port
        serialPort.disableReceiveTimeout(); // set up port options
        serialPort.enableReceiveThreshold(1);
        serialPort.setSerialPortParams(9600, SerialPort.DATABITS_8,
        SerialPort.STOPBITS_1, SerialPort.PARITY_NONE);

        input = new BufferedReader(new
        InputStreamReader(serialPort.getInputStream())); // setting up
        input stream from arduino

        serialPort.addEventListener(this); // add event listener on port
        serialPort.notifyOnDataAvailable(true); // turn on data
        availability notifications

    } catch (Exception e) {
        System.err.println("Initialization failed : " + e.toString());
    }
}
```

Figure 9-13 RXTX Arduino Serial Port Connectivity

Figure 9-13 displays the initializeSerial method of the respiratorytest4 class which is the first method to be run by the respiratorytest4 constructor when the show graph button is clicked. This method uses the RXTX library to find and open the appropriate port to facilitate communication between the Arduino and Java application. The method begins by obtaining a list of all possible ports which are available and searches this list for of the ports stored in the

PORT_NAMES variable. The PORT_NAMES variables contains the ports "/dev/tty.usbserial-A9007UX1", "/dev/ttyUSB0" and "COM3" which are the ports used by RXTX in the Mac OS, Linux, and Windows operating systems. This provides flexibility to the Java application as it is platform independent. Once the port name is found, the corresponding port id is stored and used to open the port for use. Next, an input stream is set up so that the Java application can read data from the Arduino through the port. Finally, the options for setting up an event listener on the port and enabling data availability notifications are turned on. Enabling these options provides the capabilities for the required real-time graphs by giving the Java application the ability to listen and retrieve data on the port as it is available.

```
public respiratorytest4(String title) {
    super(title);

    initializeSerial();// connect to Arduino through serial port
    float data[];
    timeSeriesCollection = new TimeSeriesCollection();
    seriesX = new TimeSeries("Data");

    timeSeriesCollection.addSeries(seriesX); // add dataset to
    timeseriescollection

    chart = createChart();// create chart
    ChartPanel chartPanel = new ChartPanel(chart); // create chart panel
    chartPanel.setFillZoomRectangle(true);
    chartPanel.setMouseWheelEnabled(true);
    chartPanel.setPreferredSize(new java.awt.Dimension(1000,500));
    setContentPane(chartPanel); // set chart to chart panel
}
```

Figure 9-14 Respiratorytest4 Constructor

Figure 9-14 presents the constructor for the respiratorytest4 class. Once the initializeSerial method has complete, the JFreeChart library is used to create a chart and a chart panel to hold the chart. The JFreeChart TimeSeriesCollection class is used create an object to hold to dataset with a time value against each value. This enables the chart to display the data in the dataset and show the time associated with each value. Next the chart is created with using the createChart() class which sets various options and creates the line chart using the ChartFactory.createTimeSeriesChart() function. Finally, the charts panel's dimensions are specified and the chart is displayed. However, no data has been added to the chart at this stage.

```
@Override
public synchronized void serialEvent(SerialPortEvent event) {
    if(event.getEventType() == SerialPortEvent.DATA_AVAILABLE) // run when
data is available on the connected port
    {
        try
        {
            String inputLine = input.readLine();// read data

            data1 = Float.valueOf(inputLine);
            dataarray[count] = data1;
            count = count + 1;

            float in_x = new Float(data1);

            this.timeSeriesCollection.getSeries(0).add(new
            Millisecond(),in_x); // add data to dataset

            if (count == 9000){ // if 15 mins has pasted
                JOptionPane.showMessageDialog(null,"You Have Successfully
                Completed The Test");
                rchart.setVisible(false);
                analysedata1();// analyse collected data
                close();// close connection on serial port
            }
        }
        catch(Exception ex)
        {
        }
    }
 }
```

Figure 9-15 Respiratorytest4 Serial Event Listener Code

Figure 9-15 displays the code developed to update the graph in real-time. The serialEvent subclass overrides the super class so that it can inherit and change its behaviour. It then listens for events on the open port and checks if data is available when an event occurs. This functionality provides the Java application with the ability to receive, store and graph the ten pieces of data send by the Arduino each second. When data is available, the data is captured using the inputstream and then added to the previously create timeseries held within the timeSeriesCollection object. The respiratory test then runs for the duration of fifteen minutes, collecting ten pieces of data every second for 900 seconds. When the test is complete, a confirmation message is presented to the user and the data is then sent for analysis.

```
int max=0;
int min=1000;

for(c=0;c<dataarray.length;c++){ // loop through all data collected
    if (dataarray[c]<max){
            count1 = count1 + 1;
            if(count1>15){ // if no data found bigger than max for 1.5
            seconds
                in = 1; // inhale complete
            }
    }
    else {
        max = dataarray[c]; // finding peak of inhalation
    }
    if (dataarray[c]>min){
            count2 = count2 + 1;
            if(count2>15){ // if no data found smaller than min for 1.5
            seconds
                out = 1; // exhale complete
            }
    }
    else{
        min = dataarray[c]; // find peak of exhalation
    }
    if(in>0 && out>0){ // if both inhale and exhale are complete
        high[e] = max; // add inhalation peak value
        low[e] = min; // add exhalation peak value
        count3 = (c/10);
        timea[e] = count3;
        e = e + 1;
        rr = rr + 1; // increment respiratory rate
        in = 0; // reset values
        out = 0;
        max = 0;
        min = 2000;
        count1=0;
        count2=0;
    }
}
```

Figure 9-16 respiratorytest4 Analysis Function

Figure 9-16 demonstrations the code implemented to analysis the data collected during the respiratory test. As the loop iterates through the values stored in the array, each value is tested to see if it is larger than the max value and smaller than the min value. In the event of no larger values than max being found for one and a half seconds, the “in” variable is set to one which means inhalation has taken place. In the event of no smaller values than min being found for one and a half seconds, the “out” variable is set to one meaning exhalation has taken place. A complete breath is indicated when both the “in” and “out” variables have a value of one. When this occurs, the respiratory rate variable is incremented by one, the inhalation and exhalation peak values are stored and the other variables are reset. The stored inhalation and exhalation peaks will be used to calculate the respiratory expansion for that breathe.

The execution times of each of the functionalities implemented in this sprint were tested using performance testing, reference section 10.6.3. All the tested functionalities passed the test meaning their performance required no further attention.

9.3.4 Improvement of Respiratory Test Page and Addition of Retraining Page

Figure 9-17 Respiratory Test Page Prototype Two

Figure 9.7 presents the second prototype of the "Respiratory Test" page before the test has begun. Several changes have been made based on the results of a usability test, please reference section 10.4.3. An instruction on how to begin the test has been inserted. The respiratory professional is now presented with the option of watching a video tutorial showing how to apply the chest and abdominal respiratory belts. Additionally, the option to display the graph before the test has begun has also been removed.

Figure 9-18 Respiratory Test Page Prototype Two

Figure 9-18 shows the display change in the "Respiratory Test" page when the respiratory test has commenced. The respiratory professional now has the option to view the real-time graph throughout the duration of the test.

```
timeSeriesCollection = new TimeSeriesCollection();
seriesX = new TimeSeries("Chest Data");
seriesY = new TimeSeries("Abdominal Data");// add extra dataset

timeSeriesCollection.addSeries(seriesX);
timeSeriesCollection.addSeries(seriesY);
```

Figure 9-19 respiratorytest4 Constructor Code Improvements

Figure 9-19 displays the improvements made to the respiratorytest4 constructor. These improvements enable datasets from both the chest and abdominal respiratory belts to be plotted on the graph. This was achieved through adding another timeseries element into the TimeSeriesCollection object.

```
String inputLine = input.readLine();
String [] inputValues = inputLine.split(",");// split data before and after
comma
data1 = Float.valueOf(inputValues[0]);
data2 = Float.valueOf(inputValues[1]);
chestA[b]=data1;
abdominalA[b]=data2;
b = b + 1;
count = count + 1;

float in_x = new Float(data1);
float in_y = new Float(data2);

this.timeSeriesCollection.getSeries(0).add(new Millisecond(),in_x);
this.timeSeriesCollection.getSeries(1).add(new Millisecond(),in_y); // add
data to both datasets
```

Figure 9-20 respiratorytest4 Serial Event Listener Code Improvements

Figure 9-20 presents the changes made to the serial event listener. The inputLine value is split as chest and abdominal respiratory data is being received from the Arduino as a single comma separated string. Next both values are stored and added to the corresponding timeseries element which in turn plots both datasets on the real-time graph.

```
for(int f=0;f<highChest.length;f++){
    chestDiff[f] = highChest[f]-lowChest[f]; // get difference between
inhale and exhale peaks
    abdominalDiff[f] = highAbdominal[f]-lowAbdominal[f];
    if(abdominalDiff[f] > chestDiff[f]){
        each_b_diff[f] = abdominalDiff[f]-chestDiff[f];
    }
    else {
        each_b_diff[f] = chestDiff[f]-abdominalDiff[f]; // get difference
between chest and abdominal expansions
    }
}
```

Figure 9-21 respiratorytest4 Analysis Function Improvements

A vast amount of additional functionality was added to the analysis function of the respiratorytest4 class. Functionality was added to calculate and store the respiratory rate, inhalation peaks and exhalation peaks using the data obtained from both respiratory belts. The previous method of calculating these values was adopted in this case. Figure 9-21 displays another piece of code added to the analysis section. This loop calculates the

difference between the chest and abdominal inhalation and exhalation peaks for each breath. Following this, the absolute difference between the chest and abdominal expansion for each breath is calculated. Additional code was implemented to calculate the average difference and overall difference for ten one minute intervals and the session total.

Four additional methods were implemented with the role of inserting the data generated in the analysis function into the appropriate tables in the database. The fourth method is called when it is the user's second session; otherwise, the report generation class is called.

```
try {
    st = conn.createStatement();
} catch (SQLException ex) {
    Logger.getLogger(report.class.getName()).log(Level.SEVERE, null, ex);
}
resultSet = st.executeQuery("select * from EACH_BREATH where
USERNAME='"+gusername+"'");

// Convert resultset to XML
new RMXMLWriter().writeObject(resultSet, "C:/Users/Colm/Documents/Final
Year Project/reports/Dataset.xml");

// Load Reportmill file template
RMDocument template = new RMDocument("C:/Users/Colm/Documents/Final Year
Project/reports/template3.rpt");

// Generate report from result set and get PDF
RMDocument report = template.generateReport(resultSet);

report.write("C:/Users/Colm/Documents/Final Year
Project/reports/"+gusername+"Report.pdf");
JOptionPane.showMessageDialog(null,"Generated PDF Report");
respiratorytest.success();
```

Figure 9-22 Report Generation Class Code

Figure 9-22 presents the report class which was developed to include the required functionality of report generation. The ReportMill API was utilized in this class to enable the generation of reports with the PDF file type. Firstly, the data to be inserted into the report was fetched from the database and stored in a resultSet variable. Next an XML file is generated as the ReportMill file template requires an XML dataset in order to determine data structure. Following this, the ReportMill template is loaded, the report is generated, written to the specified PDF file and a confirmation message is displayed to the user.

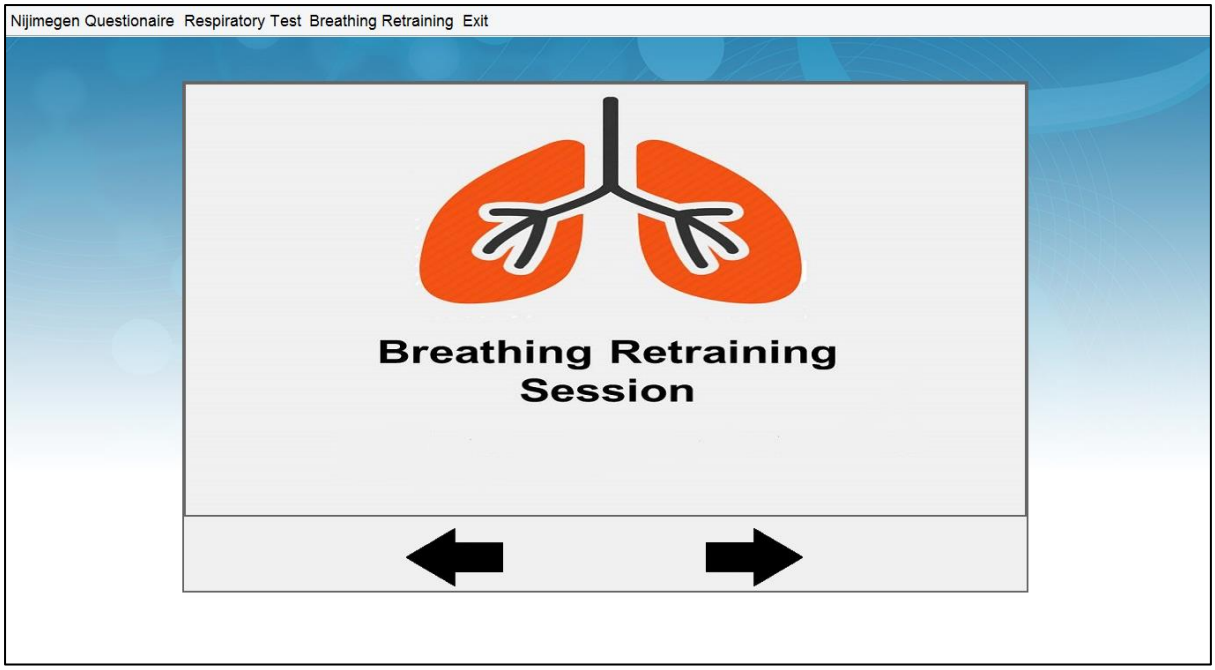


Figure 9-23 Breathing Retraining Page Prototype One

Figure 9.23 shows the first prototype for the "Breathing Retraining" page and displays the first slide in the breathing retraining session. The user is presented with the option to navigate back and forward through the slides using the left and right arrows.

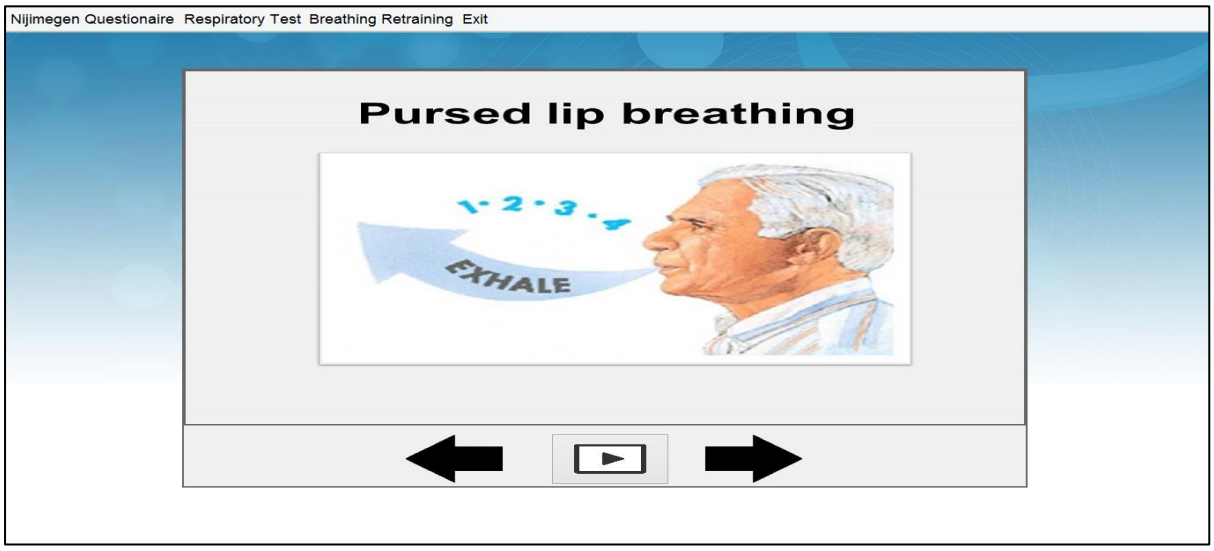


Figure 9-24 Breathing Retraining Page Prototype One

Figure 9.24 presents one of the two slides in the breathing retraining session which contains a video demonstration of the specified breathing technique. The user has the option to begin the video by selecting the play button in between the navigation arrows.

```
m = new ImageIcon[12]; // create array of images
m[0] = new
ImageIcon(getClass().getClassLoader().getResource("pics/Slide1.jpg"));
m[1] = new
ImageIcon(getClass().getClassLoader().getResource("pics/Slide2.jpg"));
m[2] = new
ImageIcon(getClass().getClassLoader().getResource("pics/Slide3.jpg"));
m[3] = new
ImageIcon(getClass().getClassLoader().getResource("pics/Slide4.jpg"));
m[4] = new
ImageIcon(getClass().getClassLoader().getResource("pics/Slide5.jpg"));
m[5] = new
ImageIcon(getClass().getClassLoader().getResource("pics/Slide6.jpg"));
m[6] = new
ImageIcon(getClass().getClassLoader().getResource("pics/Slide7.jpg"));
m[7] = new
ImageIcon(getClass().getClassLoader().getResource("pics/Slide8.jpg"));
m[8] = new
ImageIcon(getClass().getClassLoader().getResource("pics/Slide9.jpg"));
m[9] = new
ImageIcon(getClass().getClassLoader().getResource("pics/Slide10.jpg"));
m[10] = new
ImageIcon(getClass().getClassLoader().getResource("pics/Slide11.jpg"));
m[11] = new
ImageIcon(getClass().getClassLoader().getResource("pics/Slide12.jpg"));
piclab.setIcon(m[0]); // set jlabel icon to first image
```

Figure 9-25 Breathing Retraining Code

Figure 9-25 shows the code implemented to provide the breathing retraining slideshow. Each image included in the ImageIcon array was created using Microsoft PowerPoint. As the user navigates through the slide show using the arrows the different images in the array are displayed. When the user reaches slide number seven or eight a play button appears which can be used to play the video tutorial which accompanies the text and visual description of the specified breathing technique.

The execution times of each of the functionalities implemented in this sprint were tested using performance testing, reference section 10.6.4. This test case was considered a success as all tested functionalities passed the test.

9.3.5 Breathing Retraining, Registration, and Serial Event Improvements

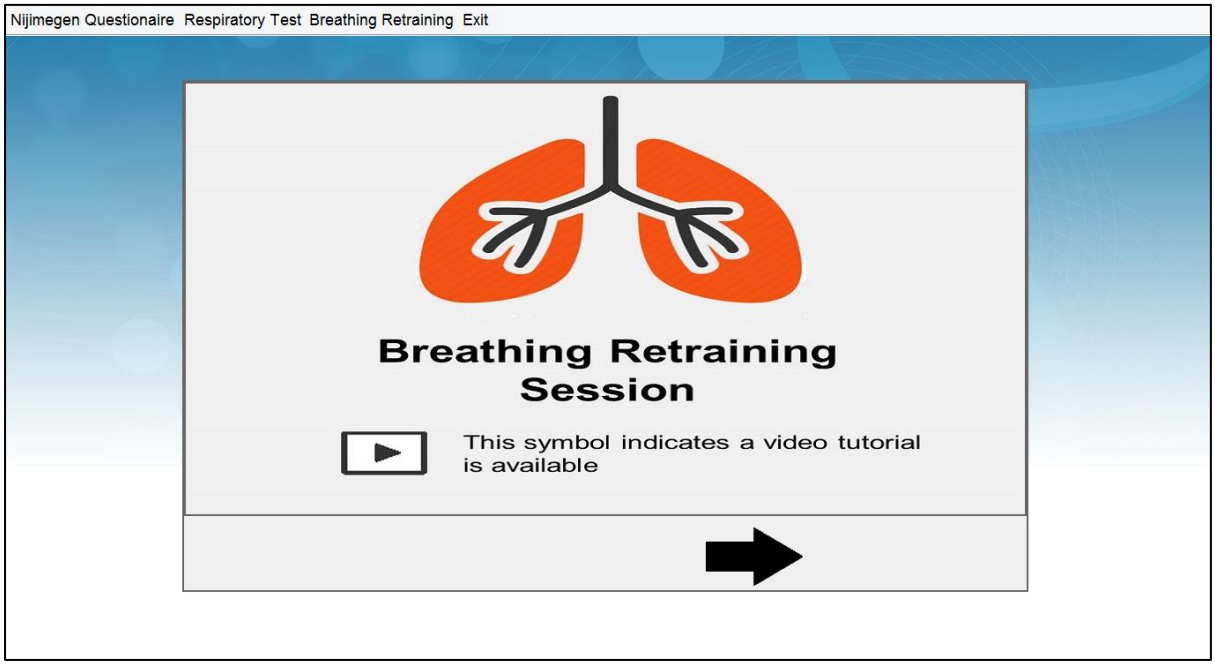


Figure 9-26 Breathing Retraining Page Prototype Two

Figure 9-26 displays the second prototype for the first slide of the breathing retraining session. As a result of usability testing several changes were made, please reference section 10.4.4. The informative text has been added to the page to advise the user to select the play symbol to watch a video tutorial when available. Additionally, the previous slide navigation arrow has been removed as no previous slide exists.

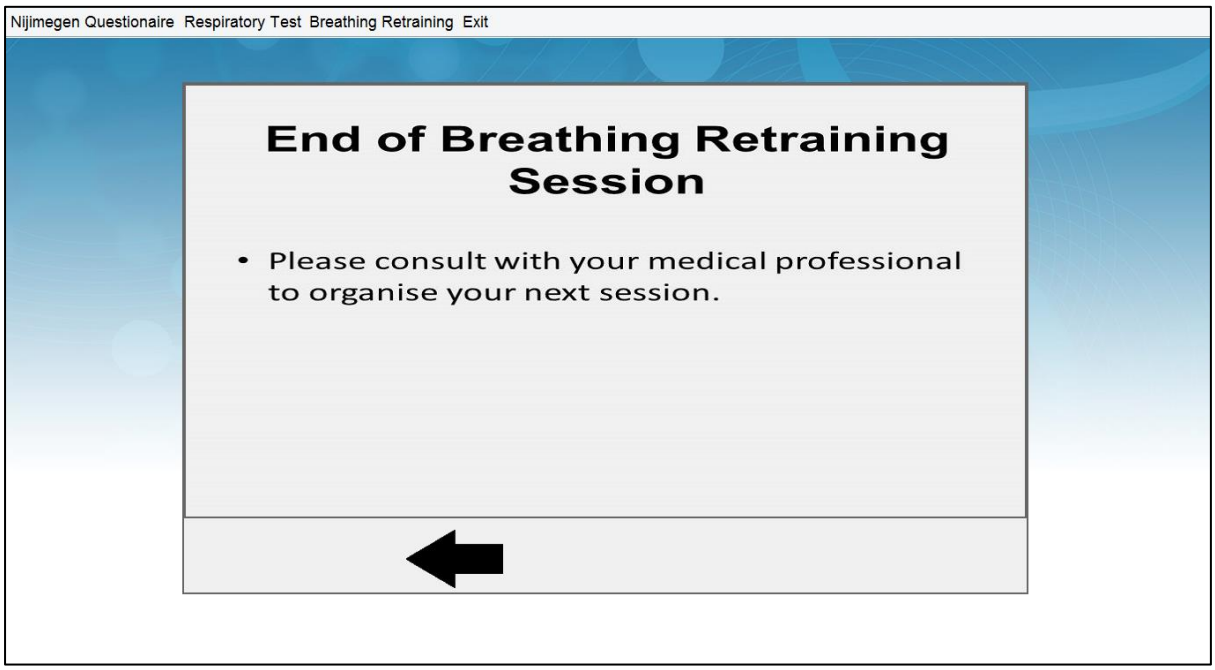


Figure 9-27 Breathing Retraining Page Prototype Two

Figure 9-27 exhibits the breathing retraining session's newly added slide. The change of removing the next slide navigation arrow has been made as a result of the findings in the usability test which can be found in section 10.4.4.

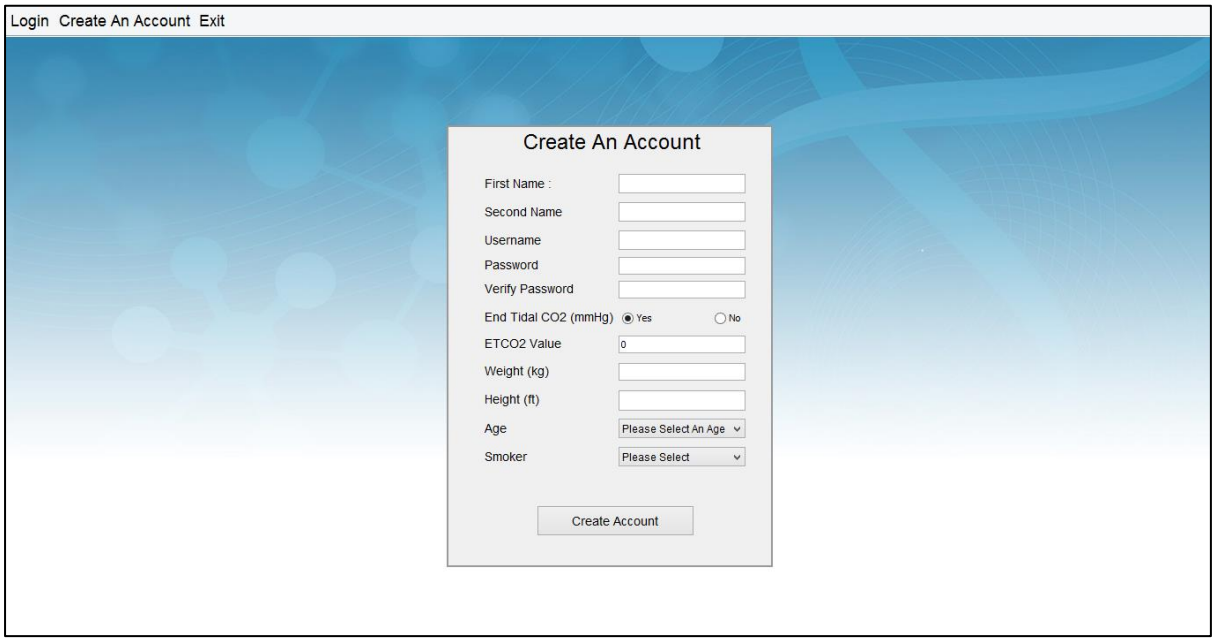


Figure 9-28 Addition of ETCO2 option in registration form

Figure 9-28 presents the changes made to the "Create An Account" page's registration form as a result of a meeting with Aisling McGowan, please reference section 4.10. Radio buttons have been added to enable the user to specify if they do or do not possess a current ETCO2 value. In the event of the user selecting the "yes" radio button, an additional text field is displayed for the user to enter the value. In the event of the user selecting the "no" radio button, the ETCO2 value is then obtained by respiratory professional inputting the ETCO2 value for each of the ten one minute intervals during the respiratory test.

A unit test case was created to test the newly added functionality, reference section 13.4.This phase of unit testing was a success as all tests in the test case passed.

```
if(inputLine.length()>0){ // checking data received is not empty
    String [] inputValues = inputLine.split(",");
    if(inputValues.length>1 && inputValues[0].length()>0 &&
    inputValues[1].length()>0){ // checking no null values are received
        matcher = pattern.matcher(inputValues[0]);
        matcher2 = pattern.matcher(inputValues[1]);
        if(!matcher.find() && !matcher2.find()){// checking data is numeric
            data1 = Float.valueOf(inputValues[0]);
            data2 = Float.valueOf(inputValues[1]);
            if(data1>0 && data2>0){ // verifying no negative values
                chestA[b]=data1;
                abdominalA[b]=data2;
                b = b + 1;
                count = count + 1;
                float in_x = new Float(data1);
                float in_y = new Float(data2);
                this.timeSeriesCollection.getSeries(0).add(new
                Millisecond(),in_x);
```

```
                    this.timeSeriesCollection.getSeries(1).add(new
                    Millisecond(),in_y);
                }
            }
        }
}
```

Figure 9-29 respiratorytest4 Serial Event Improvement

Figure 9-29 shows the improvements made to the respiratorytest4 class's serial event. These changes were made as a result of the finding in the associated unit test case, please reference section 13.5. A new unit test case was developed to test these newly added data validation method, reference section 13.6. The successful results of this unit test case concluded that the actions taken correctly fixed the problems encountered.

9.3.6 Report Generation Improvement and Addition of Help Option

```
Map map = new HashMap();
map.put("Row1", resultSet);
map.put("Row2", resultSet1);
map.put("Row3", resultSet2);
map.put("Row4", resultSet3);
map.put("Row5", resultSet4);
map.put("Row6", resultSet5);

// Generate XML for ResultSet
convertXML.main(gusername);

// Load Reportmill file template
RMDocument template = new RMDocument("C:/Users/Colm/Documents/Final Year
Project/reports/template3.rpt");

// Generate report from result set and get PDF
RMDocument report = template.generateReport(map);
String file_1 = "";// report file names
String file_2 = "";
String filename1 = "";
String filename2= "";
filename1 = filename +"1.PNG";// image file names
filename2 = filename +"2.PNG";
int temp = 2;
if(sess2.equals("Final")){
    file_1 = "C:/Users/Colm/Documents/Final Year
    Project/reports/"+gusername+"Report"+ sess2 + ".pdf";
```

```
    file_2 = "C:/Users/Colm/Documents/Final Year
    Project/reports/"+gusername+"Report"+ sess + ".pdf";
    temp=3;
}
else {
    file_1 = "C:/Users/Colm/Documents/Final Year
    Project/reports/"+gusername+"Report"+ sess + ".pdf";
    file_2 = "C:/Users/Colm/Documents/Final Year
    Project/reports/"+gusername+"Report.pdf";
    temp = 2;
}

report.write(file_2); // generate report
JavaWritePDF.main(file_1,file_2, filename1,filename2,temp); // adding
charts to reports
```

Figure 9-30 Report Class Improvements

Figure 9-30 displays the code changes implemented in the report class. These changes were made as the generated report required data from multiple result sets. The HashMap class was utilized to associate each result set with a key and to store all result sets in a single object. Following this, an XML file containing an XML dataset similar in structure to the HashMap object was required. The previously used RMXMLWriter() class for converting the result set to XML did not work when passed a HashMap object. As a result, the convertXML class was developed to generate the required XML file. Once the XML file has been successfully created, the report is generated like before and the JavaWritePDF class is class. The JavaWritePDF class was developed to insert the images generated during the respiratory test into the report.

```
// creating new documents
DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
DocumentBuilder builder = factory.newDocumentBuilder();
Document doc = builder.newDocument();
Element results = doc.createElement("Results");// adding parents element
doc.appendChild(results);

ResultSetMetaData rsmd = resultSet.getMetaData();// obtaining metadata
ResultSetMetaData rsmd1 = resultSet1.getMetaData();
ResultSetMetaData rsmd2 = resultSet2.getMetaData();
ResultSetMetaData rsmd3 = resultSet3.getMetaData();
ResultSetMetaData rsmd4 = resultSet4.getMetaData();
ResultSetMetaData rsmd5 = resultSet5.getMetaData();
int colCount = rsmd.getColumnCount();// counting columns
int colCount1 = rsmd1.getColumnCount();
int colCount2 = rsmd2.getColumnCount();
int colCount3 = rsmd3.getColumnCount();
int colCount4 = rsmd4.getColumnCount();
int colCount5 = rsmd5.getColumnCount();

while (resultSet.next()) {// iterating through first resultset
  Element row1= doc.createElement("Row1");
  results.appendChild(row1); // adding element to parent element
  for (int i = 1; i <= colCount; i++) {
    String columnName = rsmd.getColumnName(i);
    Object value = resultSet.getObject(i);
    Element node = doc.createElement(columnName);
```

```
    node.appendChild(doc.createTextNode(value.toString()));
    row1.appendChild(node); // adding data to Row1 element
  }
}
```

Figure 9-31 convertXML Class

Figure 9-31 presents a segment of code from the convertXML class which was developed to generate the required XML file for the generation of the report. This class makes use of the DocumentBuilderFactory and TranformerFactory objects in the XML file creation process. Firstly, a document is created using the DocumentBuilder object and the parent element "Results" is added to the document. Following this, the metadata is obtained from each result set and in turn the number of columns in each result set. Finally, a loop creates a child element called "Row1", adds this element to the "Results" element and then inserts the columns and values for each row into the "Row1" element. This process is then repeated for each of the remaining result sets. Once each result set has been processed and added to an element in the document, the TranformerFactory object is used to create the XML file.

```
public void manipulatePdf(String src, String dest) throws IOException,
DocumentException {
PdfReader reader = new PdfReader(src); // reading template file
PdfStamper stamper = new PdfStamper(reader, new FileOutputStream(dest)); //
writing to destination file
stamper.insertPage(reader.getNumberOfPages() +
1,reader.getPageSizeWithRotation(1)); // adding new page
Image img = Image.getInstance(IMG);
img.setAbsolutePosition(0,350);
stamper.getOverContent(2).addImage(img); // inserting image
stamper.close();
File delfile = new File(SRC);
delfile.delete();// deleting template file
respiratorytest.success();
}
```

Figure 9-32 JavaWritePDF Class

Figure 9-32 displays a section of code from the JavaWritePDF class which was developed to insert the graphs generated during the respiratory test into the PDF report. This class utilizes the PdfReader and PdfStamples classes of the itextpdf library to generate a new report containing the respiratory line graphs. Firstly, the report generated in the report class is read using the PdfReader. Following this, the PdfStamper is used to add a new page into the document and insert the appropriate graph into the newly added page. Finally, the previously generated report is deleted. In the event of the user's second session, two pages are added to the documents and an image is added to each of the pages.

Finally, a help menu item was added to each page. This option when clicked, displays informative text giving a step by step guide through the actions of the associated page. Documentation test was performed to ensure the text provided was free from errors, reference section 10.10. The results of this test were positive and conclude no errors existed in the text provided in the help documentation.

9.4 Graphing Prototypes

9.4.1 Initial Prototype

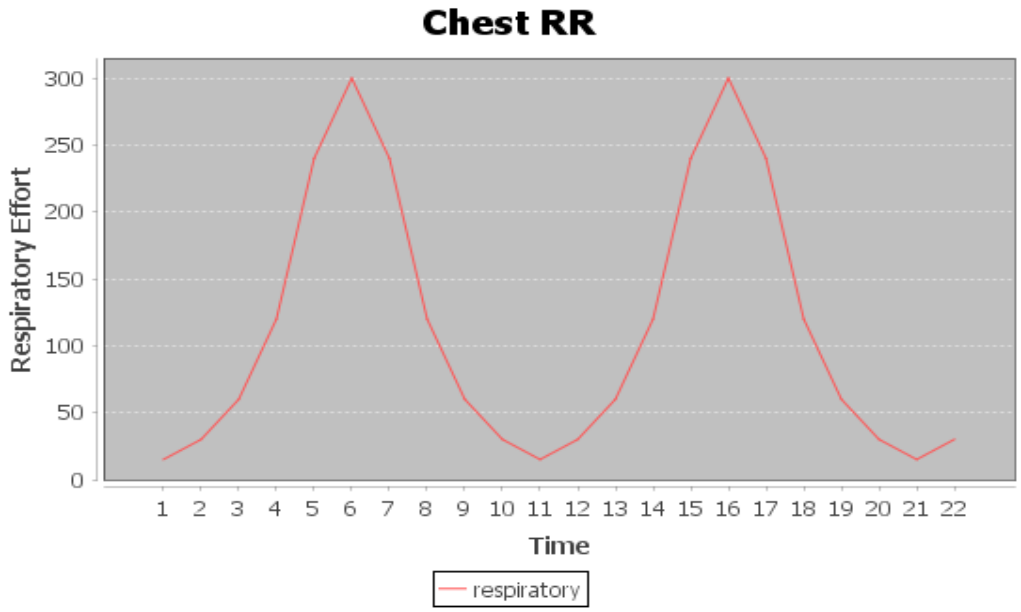


Figure 9-33 JFreeChart Prototype

Figure 9-33 shows the RR line graph prototype developed using JFreeChart. A potentiometer was used to replicate the data generated by a respiratory belt as no respiratory belts were currently available. The x-axis is time and the y-axis is respiratory effort. Readings for respiratory effort are displayed every second. The patient's inhalation is displayed in the graph as an increase in respiratory effort. The opposite can be said for exhalation.

9.4.2 Chest Respiratory Belt Integration

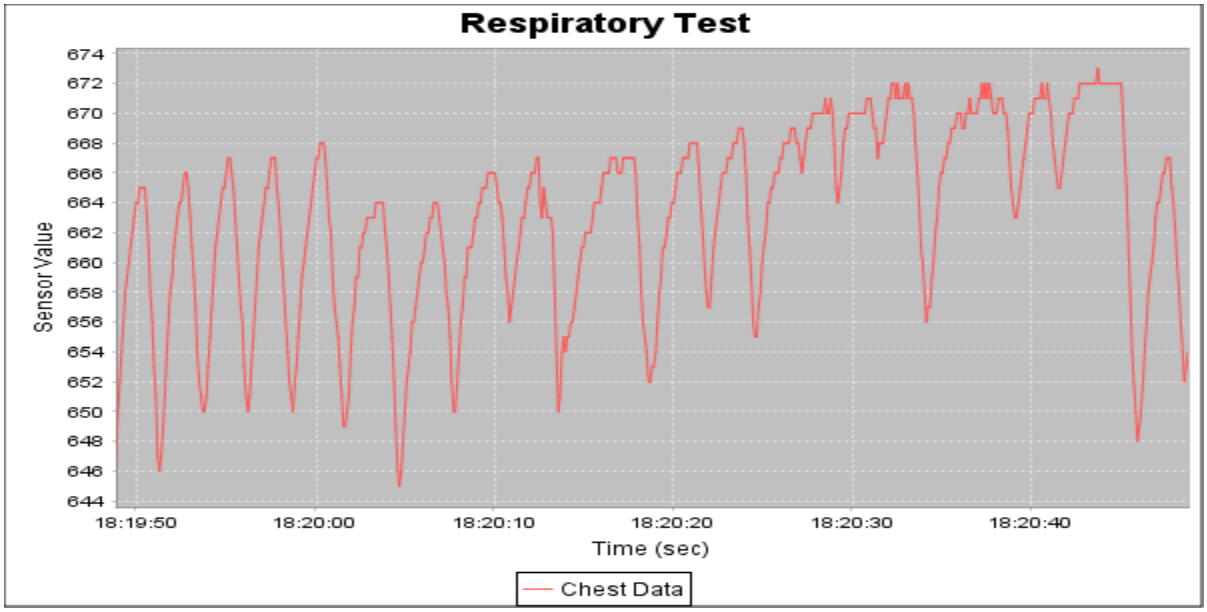


Figure 9-34 Chest Respiratory Belt Data Graph

Figure 9-34 presents the second prototype for the respiratory test generated graph. This graph was created with data generated by the newly constructed and integrated chest respiratory belt.

9.4.3 Abdominal Respiratory Belt Integration

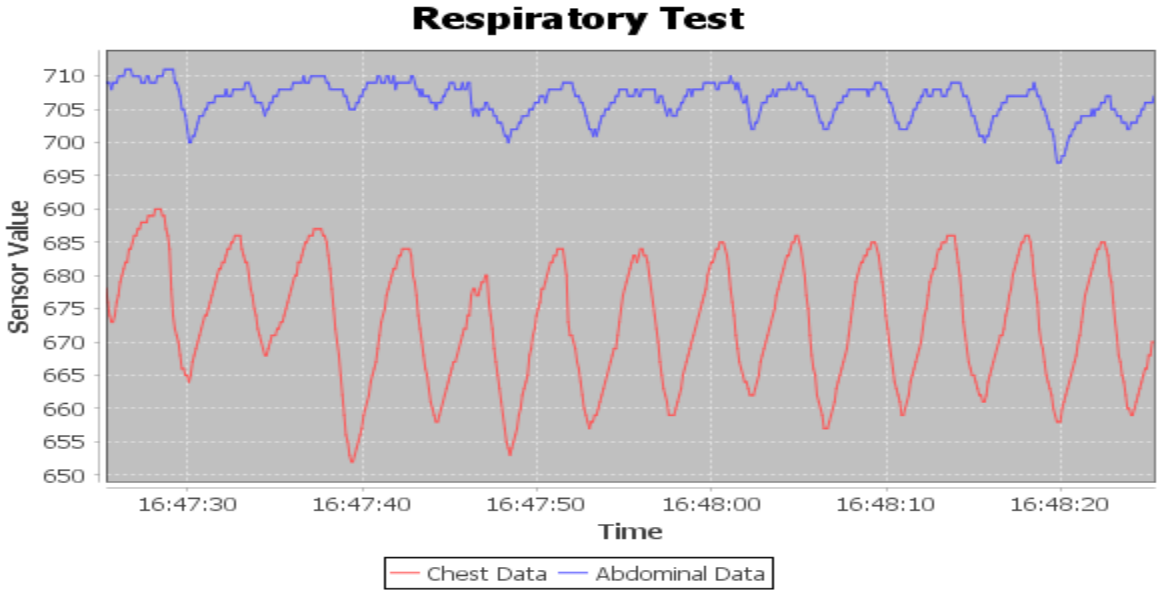


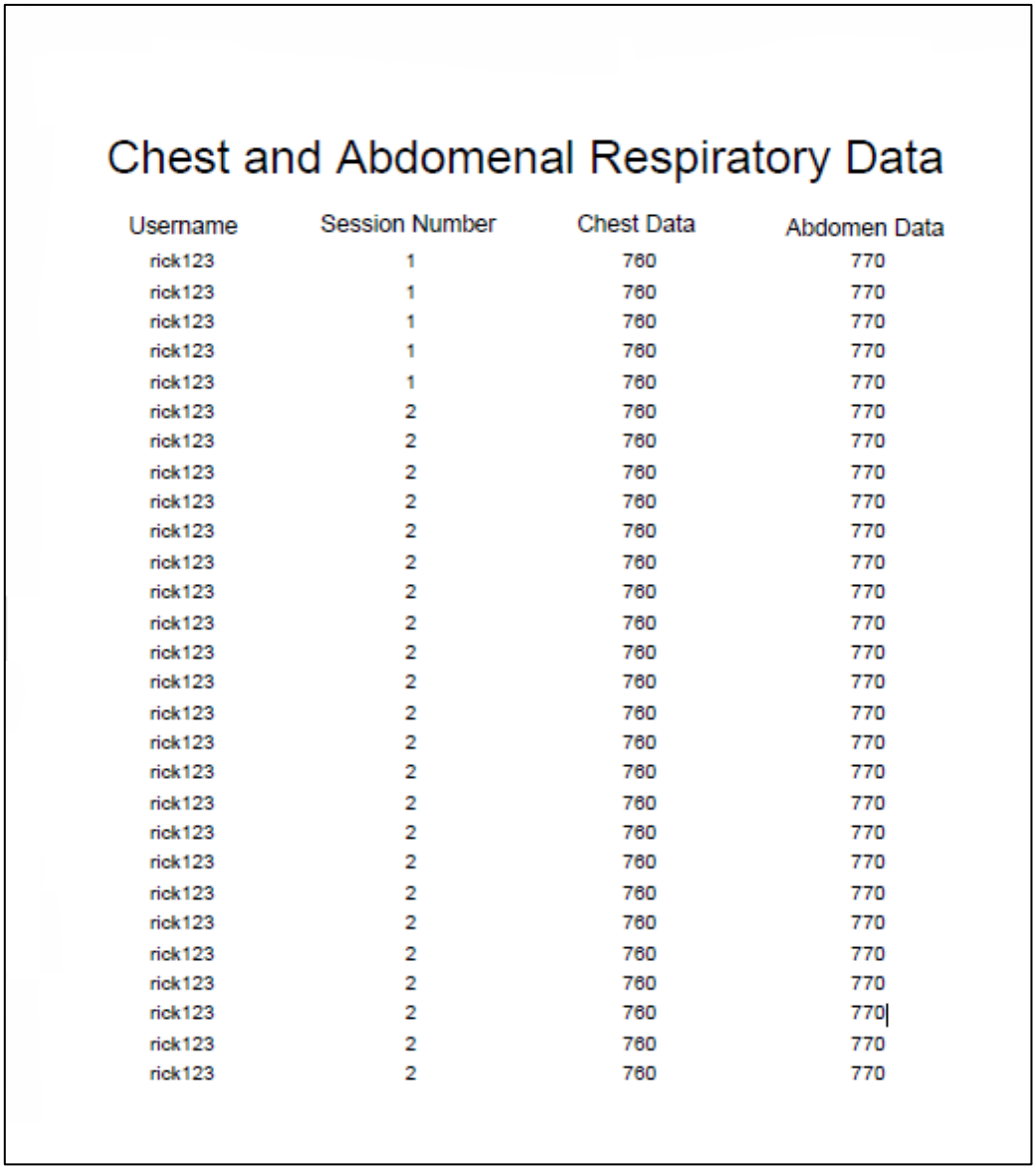
Figure 9-35 Chest and Abdominal Respiratory Graph

Figure 9-35 displays the graph generated during the respiratory test process while making use of both the chest and abdominal respiratory belts. During a meeting with Aisling McGowan, reference section 4.10, the above graph was presented to her for examination. Aisling was able to understand the graph and explain the breathing pattern displayed. Aisling concluded the meeting by stating that the graph was sufficiently accurate and informative as the total lung capacity and residual volume, meaning lungs are empty, were synchronized.

9.5 Report Generation

This section will show the prototypes and changes associated with stages in the design of the generated report.

9.5.1 Initial Report



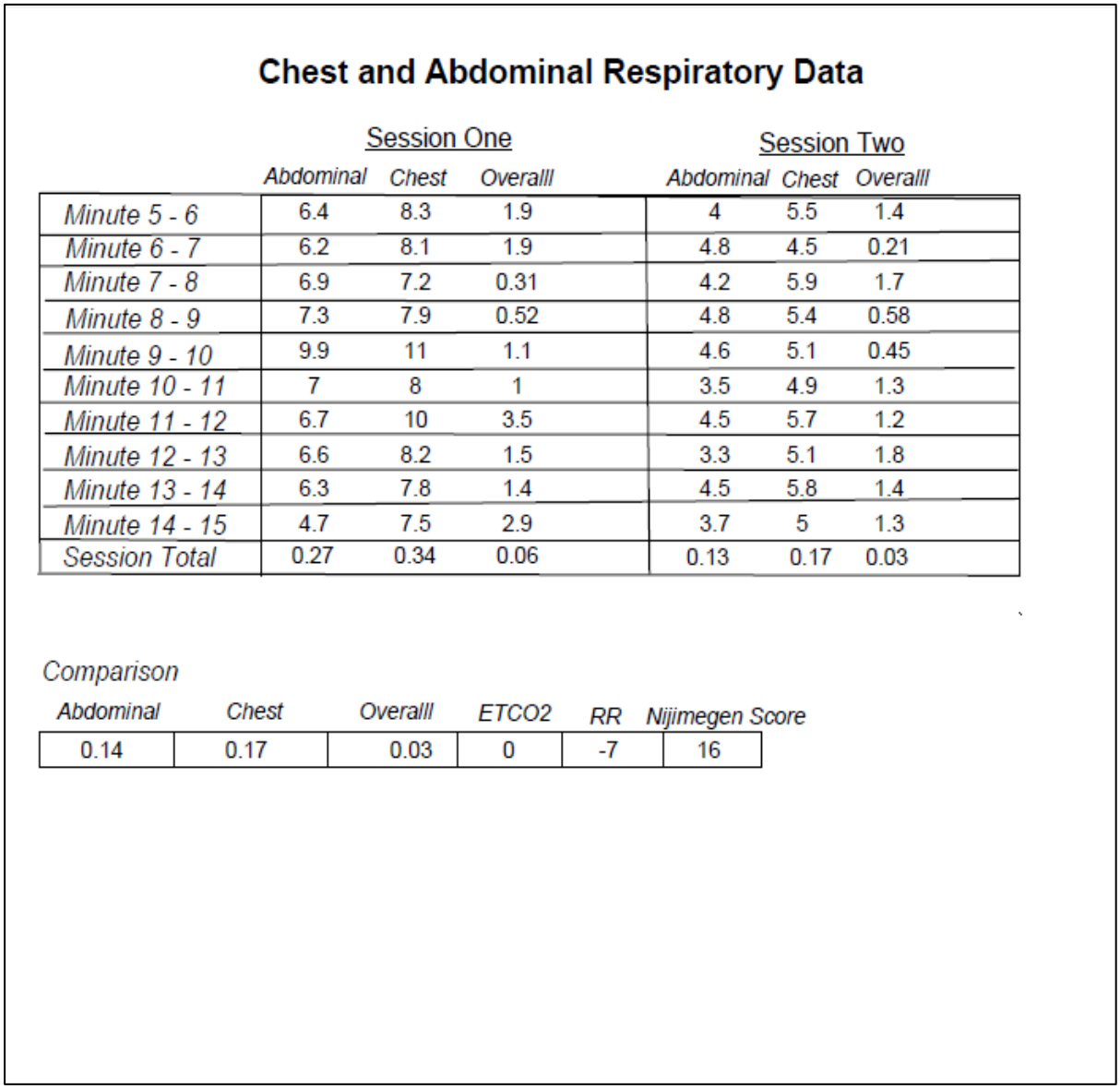
Chest and Abdomenal Respiratory Data

Username	Session Number	Chest Data	Abdomen Data
rick123	1	760	770
rick123	1	760	770
rick123	1	760	770
rick123	1	760	770
rick123	1	760	770
rick123	2	760	770
rick123	2	760	770
rick123	2	760	770
rick123	2	760	770
rick123	2	760	770
rick123	2	760	770
rick123	2	760	770
rick123	2	760	770
rick123	2	760	770
rick123	2	760	770
rick123	2	760	770
rick123	2	760	770
rick123	2	760	770
rick123	2	760	770
rick123	2	760	770
rick123	2	760	770
rick123	2	760	770
rick123	2	760	770
rick123	2	760	770
rick123	2	760	770
rick123	2	760	770
rick123	2	760	770
rick123	2	760	770
rick123	2	760	770

Figure 9-36 Report Prototype One

Figure 9-36 displays the initial design of the generated report. The report is filled with the data produced by the chest and abdominal respiratory belts. The content of the report dynamically increases as the user performs more respiratory tests.

9.5.2 Report Improvement



Chest and Abdominal Respiratory Data

	Session One			Session Two		
	Abdominal	*Chest*	*Overalll*	*Abdominal*	*Chest*	*Overalll*
Minute 5 - 6	6.4	8.3	1.9	4	5.5	1.4
Minute 6 - 7	6.2	8.1	1.9	4.8	4.5	0.21
Minute 7 - 8	6.9	7.2	0.31	4.2	5.9	1.7
Minute 8 - 9	7.3	7.9	0.52	4.8	5.4	0.58
Minute 9 - 10	9.9	11	1.1	4.6	5.1	0.45
Minute 10 - 11	7	8	1	3.5	4.9	1.3
Minute 11 - 12	6.7	10	3.5	4.5	5.7	1.2
Minute 12 - 13	6.6	8.2	1.5	3.3	5.1	1.8
Minute 13 - 14	6.3	7.8	1.4	4.5	5.8	1.4
Minute 14 - 15	4.7	7.5	2.9	3.7	5	1.3
Session Total	0.27	0.34	0.06	0.13	0.17	0.03

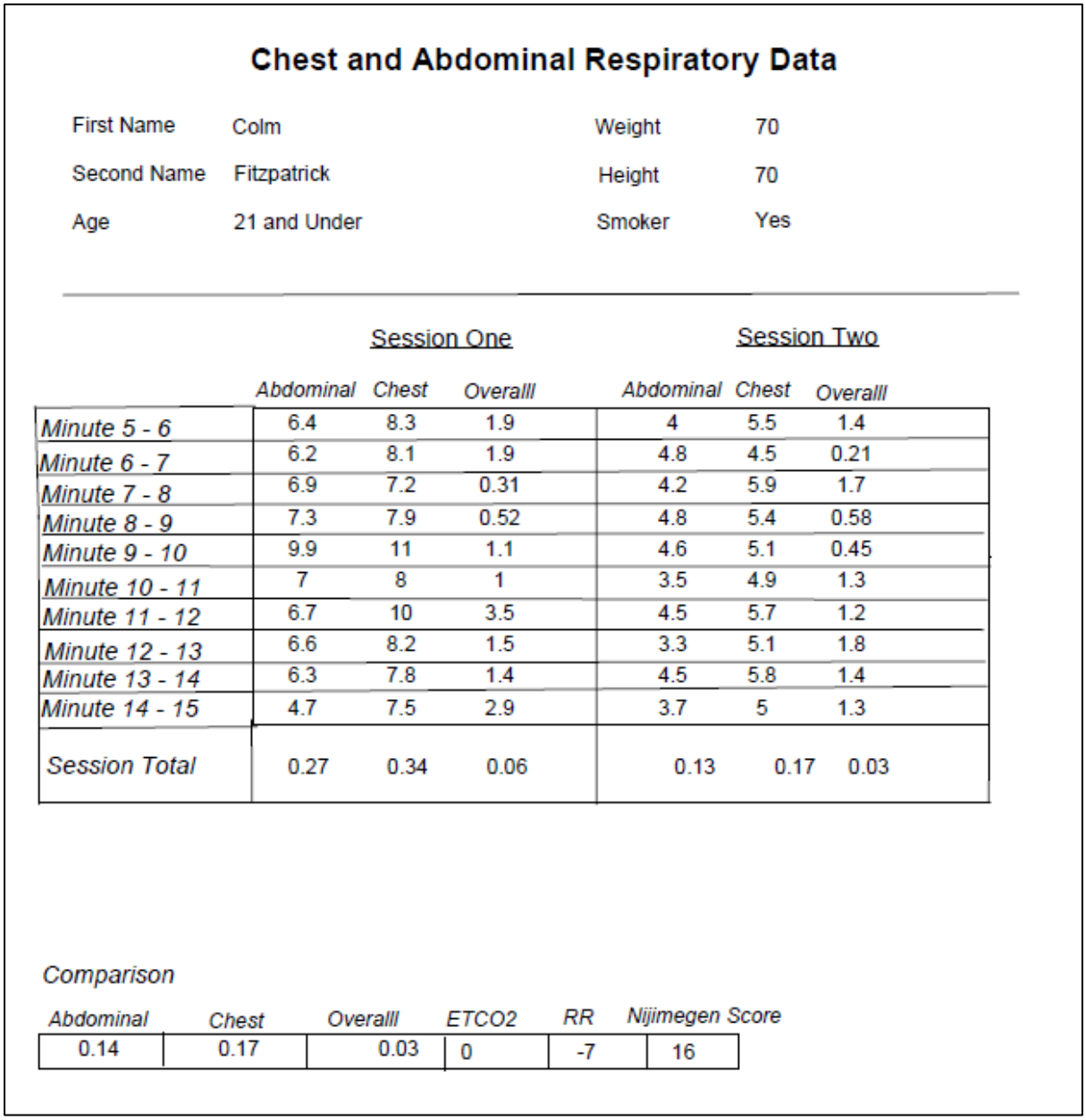
Comparison

Abdominal	*Chest*	*Overalll*	*ETCO2*	*RR*	*Nijimegen Score*
0.14	0.17	0.03	0	-7	16

Figure 9-37 Report Prototype Two

Figure 9-37 presents the second prototype for the design of the generated report. As a result of usability testing, multiple changes have been made to the design of the report. Please reference section 10.4.6 for details and results of the usability test. The previous raw data structure has been replaced with session averages which are split into ten one minute intervals. Additionally, a session comparison section has been added aimed at providing clear data to the respiratory professional. A usability test case was developed to test these newly added features, reference section 10.4.7. The results of this test case conclude that the actions taken successfully corrected the problems highlighted.

9.5.3 Further Report Improvement



Chest and Abdominal Respiratory Data

First Name	Colm	Weight	70
Second Name	Fitzpatrick	Height	70
Age	21 and Under	Smoker	Yes

	Session One			Session Two		
	Abdominal	*Chest*	*OverallI*	*Abdominal*	*Chest*	*OverallI*
Minute 5 - 6	6.4	8.3	1.9	4	5.5	1.4
Minute 6 - 7	6.2	8.1	1.9	4.8	4.5	0.21
Minute 7 - 8	6.9	7.2	0.31	4.2	5.9	1.7
Minute 8 - 9	7.3	7.9	0.52	4.8	5.4	0.58
Minute 9 - 10	9.9	11	1.1	4.6	5.1	0.45
Minute 10 - 11	7	8	1	3.5	4.9	1.3
Minute 11 - 12	6.7	10	3.5	4.5	5.7	1.2
Minute 12 - 13	6.6	8.2	1.5	3.3	5.1	1.8
Minute 13 - 14	6.3	7.8	1.4	4.5	5.8	1.4
Minute 14 - 15	4.7	7.5	2.9	3.7	5	1.3
Session Total	0.27	0.34	0.06	0.13	0.17	0.03

Comparison

Abdominal	*Chest*	*OverallI*	*ETCO2*	*RR*	*Nijimegen Score*
0.14	0.17	0.03	0	-7	16

Figure 9-38 Report Prototype Three Page One

Figure 9-38 displays the third prototype for the first page of the generated report. These changes were agreed upon during a meeting with Aisling McGowan, reference section 4.10 for meeting summary. This meeting concluded that the patient's personal information and graphs for each session were to be included in the report. The first page of the report now includes all the required patient information in a structured and easy to read format.

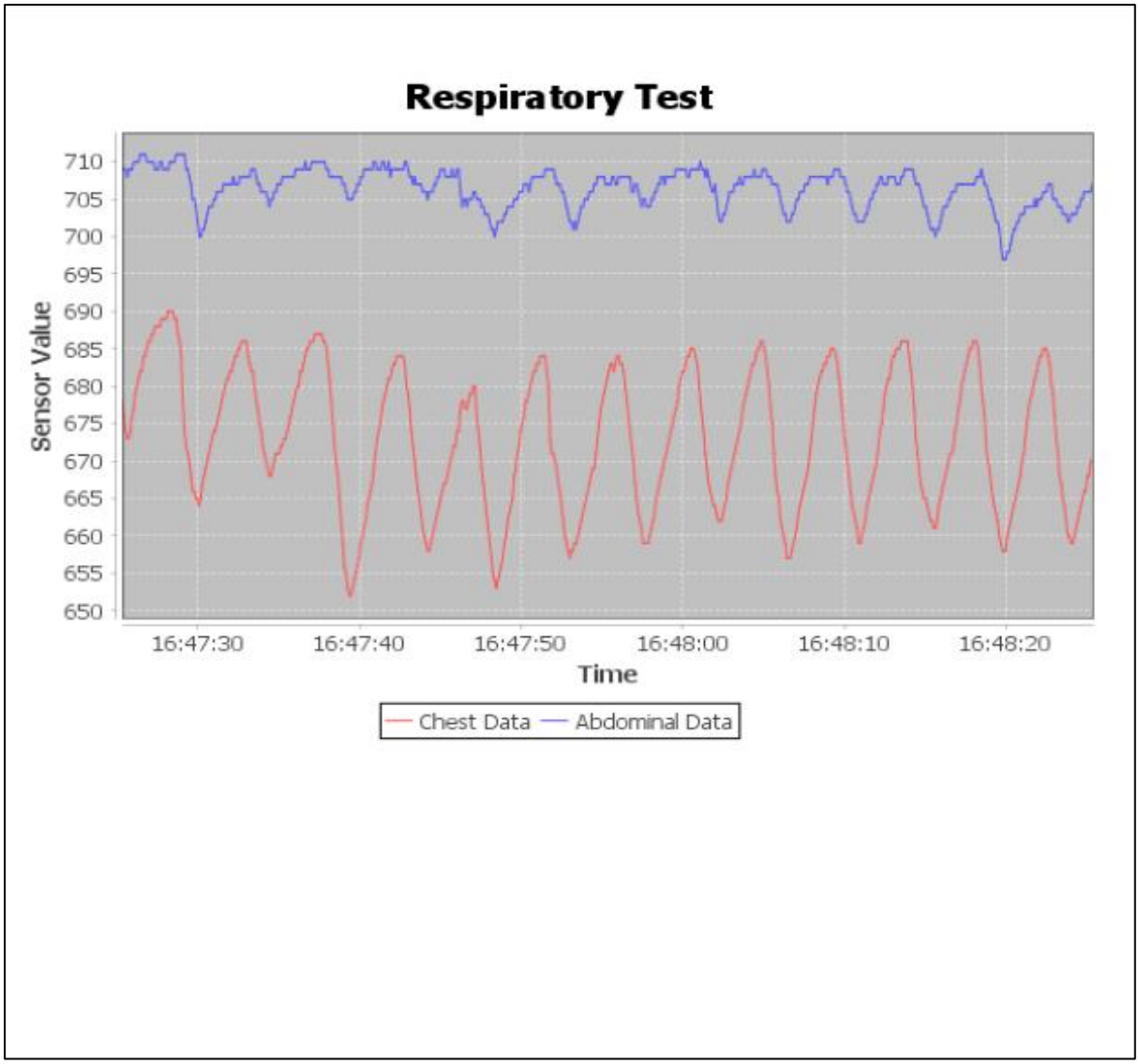


Figure 9-39 Report Prototype Three Page Two

Figure 9-39 presents the second page of the generated report. This page displays the graph generated during the first session of the respiratory test. Once the second respiratory test is complete, a third page is added to the report containing the graph generated in the second session. The generated report now satisfies the requirements provided by respiratory scientist Aisling McGowan.

9.5.4 Change to Percent Difference

Following a meeting with Damian Bourke, reference section 4.11, a change to how data was presented on the generated report was made. This change involved presenting the overall differences between the chest and abdominal expansions as a percent change value. This change was made with the hope of making the data presented on the report easier to understand.

9.6 Respiratory Belt Construction

This section will describe the steps which were taken in the building of the chest and abdominal respiratory belts.

In order to measure chest and abdominal respiratory expansion, two respiratory belts were required. A method[34] for building respiratory belts through the use of carbon-black impregnated rubber cord[35] was adapted for this project.

9.6.1 Preparing Conductive Rubber Cord

The first stage in the construction of the respiratory belts was to cut the cord and attach conductive terminals to both ends of the cord. In order to determine the appropriate length of cord, ten participants were tested. Please reference section 4.9 for a full description and results. From this test, it was concluded that a 4" length of cord would be sufficient. Two 4" pieces, one for each belt, of the cord were precisely cut. Next conductive terminals were crimped to both ends of each piece of cord using a vice.

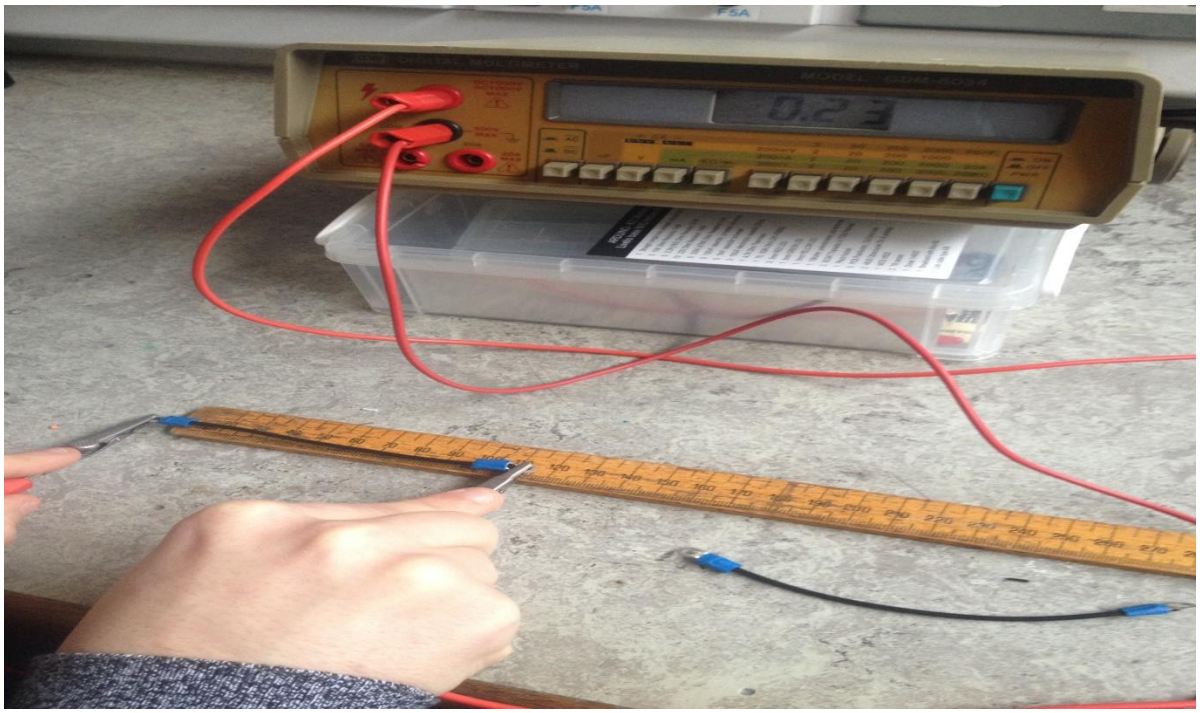


Figure 9-40 Conductive Rubber Cord Calibration

Next for calibration purposes, the resistance of both of the conductive rubber cords was measured at rest and at a full stretch, which is 70% of its length totalling at 6.8". Figure 9-40 displays that the resistance of the rubber cord at rest was 0.23 of 20,000 Ω, equalling 4600 Ω. The resistance of the cord when stretched to a length of 6.8" was then found to be 12000 Ω. These results were noted for later use in the building and configuration of the Arduino circuit. The effect of temperature on the resistance of the conductive cord was also examined. The first set of resistance data was obtained in a lab at around room temperature. The second set of resistance data was obtained in a significantly warmer environment. It was found that the resistance of the conductive cord increased as the temperature increased. However, the difference between the resistance value at rest and the resistance value fully stretched

remained virtually the same. As a result, temperature can be eliminated as a limitation for the respiratory belts.

9.6.2 Preparing the Belts

In order to measure chest and abdominal respiratory expansion, two adjustable fabric respiratory belts were assembled. Two Velcro belts and elastic were obtained to enable the building of the belts. The Velcro belt was cut in the middle of the area which didn't contain any of the Velcro material. A 4" piece of the elastic was then cut and sewed to each end of the Velcro belt. The purpose of the elastic is to support the conductive rubber cord and to prevent it from getting damaged. Following this, an electrical drill was used to drill two holes in the Velcro belt so that the conductive terminals at the end of each conductive rubber cord could be attached to the belt.

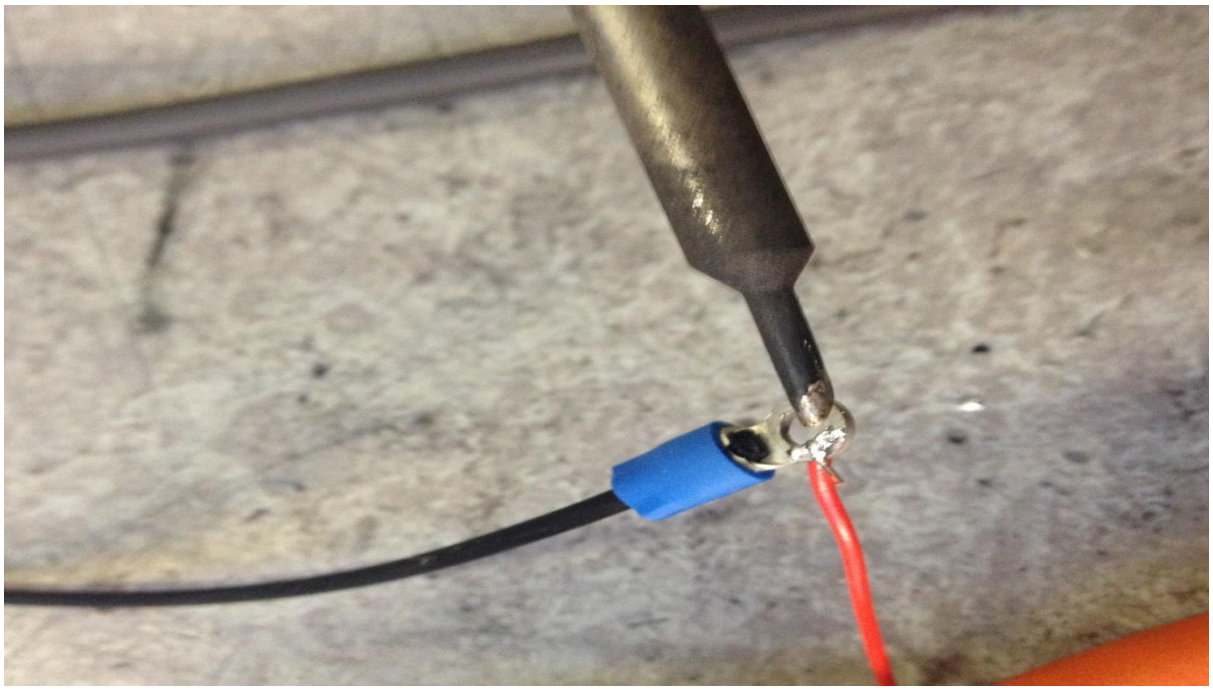


Figure 9-41 Soldering Wire to Conductive Terminal

Next two sufficiently long pieces of wire were cut and each soldered to the conductive terminals at the opposite ends of the conductive cord for durability. Figure 9-41 displays one of the wires being soldered to one of the conductive terminals. These wires act as the connection between the sensors and the Arduino. Finally, a bolt was put through each hole in the belt and each conductive terminal was secured to the belt by tightening a nut on the opposite side of the material. Once complete, the same procedure was repeated to construct the second belt. On the completion of the second belt, both belts were fitted and tested using the serial monitor of the Arduino IDE to confirm data was been received from the sensors. Once confirmed, the data was read by the Java program using the RXTX library and graphed using the JFreeChart library. While examining the graphed data, it was noticed that lines in the line graphs were fluctuating due to mechanical noise.

9.6.3 Reducing Mechanical Noise and Increasing Durability

Following a discussion with Damon Berry, reference section 4.8, it was discovered that twisting together the wires, which connect the conductive terminals of each conductive cord to the Arduino, reduces mechanical noise. Following this, the wires associated with each respiratory belt were twisted together and tested. The respiratory belts were tested again and an improvement in the quality of the graph was confirmed. Next, the focus turned to increasing the durability of the respiratory belts. To prevent the twisted wires from untangling heat shrink tubing was placed around the wires. The tubing was then heated which caused it to shrink around the wires, securing them in place. Although soldering the wires to the conductive terminals secures them sufficiently in terms of receiving data through the wires it does not make them durable. In order to reduce stress on the soldered connections and increase durability, a lower section of the wire was sewed to the belt. Sewing a lower part of the wire to the belt reduces the strain on the connection by increasing the surface area that is affected by the stress. Figure 9-42 demonstrations the result of the efforts made to increase durability.

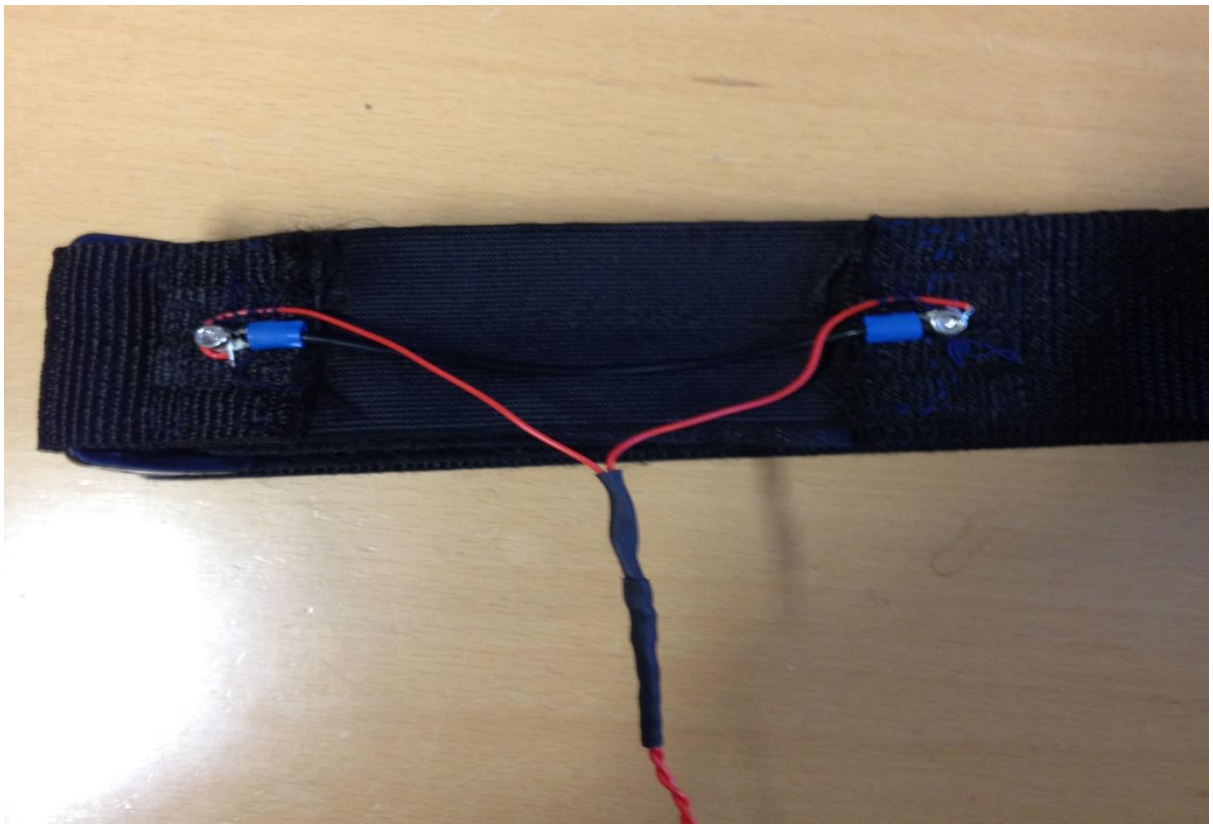


Figure 9-42 Complete Belt

9.7 Arduino Circuit and Configuration

This section will describe the various stages in the development and configuration of the Arduino circuit while stating the purpose and result of each prototype.

An Arduino Uno Kit was acquired containing an Arduino board, breadboard, 50Kohm potentiometer, USB cable, selection of resistors and a vast array of other components. The Arduino 1.6.7 IDE was used to upload programs to the Arduino.

9.7.1 Reading Variable Resistor Data through Analog Pin

```
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // print out the value you read:
  Serial.println(sensorValue);
  delay(1000);        // 1 second delay in between reads for stability
}
```

Figure 9-43 Analog Read Serial Code

Figure 9-43 presents code which was taken from the "AnalogReadSerial" example in the Arduino IDE. This example was used to read data from analog pin 0 and print it on the serial monitor.

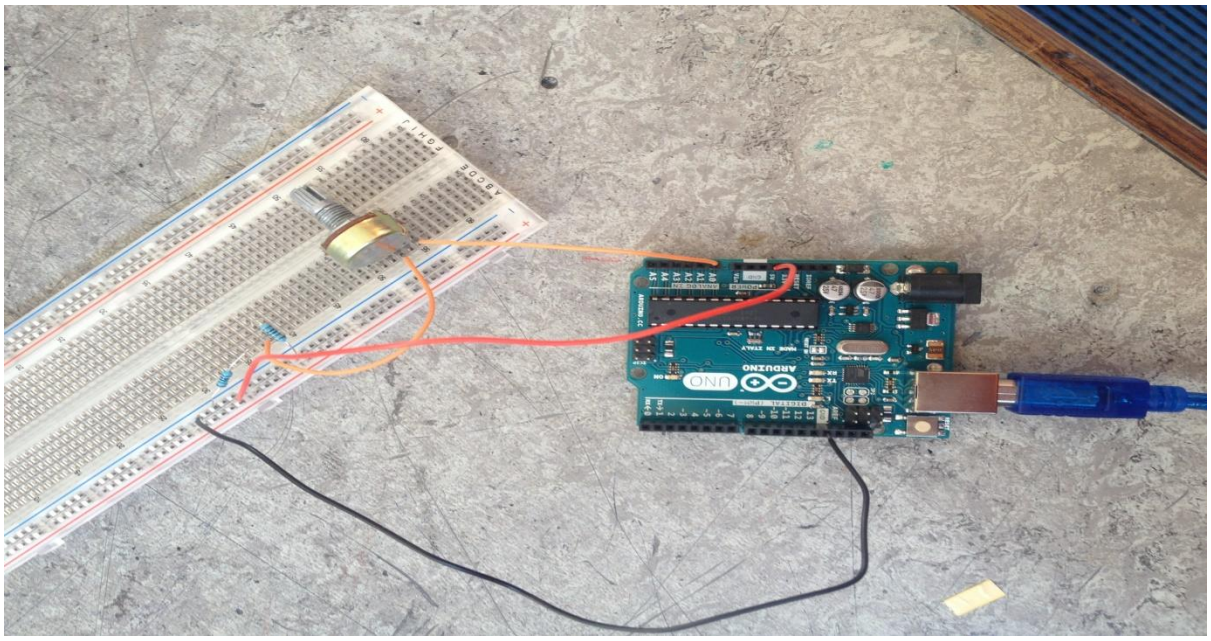


Figure 9-44 Arduino Analog Read Circuit

Figure 9-44 displays the first Arduino circuit prototype which was adapted from the example sketch "AnalogReadSerial"[48] provided by the Arduino IDE. The purpose of this prototype was to simulate a respiratory belt through the use of a potentiometer. Firstly, the circuits design was tested by verifying that data was being printed to the serial monitor. The data generated by the variable resistor was read to the Java program using the RXTX library and graphed using the JFreeChart library. The resistance of the variable resistor was adjusted to simulate respiratory data received from a respiratory belt.

9.7.2 Reading Data from Both Respiratory Belts

```
/*
  AnalogReadSerial
  Reads an analog input on pin 0 and 1.
*/

int count=0;
78nti =0;
// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0 and 1:
  int sensorValue = analogRead(A0);
  int sensorValue2 = analogRead(A1);
  String datav;
  char buffer [50];
  i=sprintf (buffer, "%d,%d", sensorValue, sensorValue2); // constructing
string with comma separated values
  for(int l= 0; l<i; l++)
  datav = datav + buffer[l];

  Serial.println(datav);
  delay(100);        // 0.1 ms delay provides 10 Hz sampling frequency.
}
```

Figure 9-45 Adapted Analog Serial Read Code

Figure 9-45 shows code which was adapted from the example sketch "AnalogReadSerial" [48] provided by the Arduino IDE. The Arduino 1.6.7 IDE was used to upload the above code which reads analog input on pin 0 and 1 onto the Arduino device. Pin 0 will be used to read analog input generated by the chest respiratory effort sensor and pin 1 will be used to read analog input generated by the abdominal respiratory effort sensor. However, this program also determines the sampling frequency, through the delay function, which is typically calculated using Nyquist's Theorem. The Nyquist's Theorem states that the sampling rate must be at least twice the frequency of the highest frequency component contained within the signal[49]. A recent paper [50] presented a respiratory frequency of 0.2-0.33 Hz or 3-5 breaths per second in a patient at rest. According to Nyquist's Theorem, the sampling rate should be 0.66 Hz as it is double the highest frequency component. However after meeting with Damon Berry an engineering expert, a sample rate of 10 Hz was decided as it would provide clear and more accurate graphs which are essential to the success of this project. This sampling rate was applied to the Arduino by setting a delay of 0.1 milliseconds in between reads.

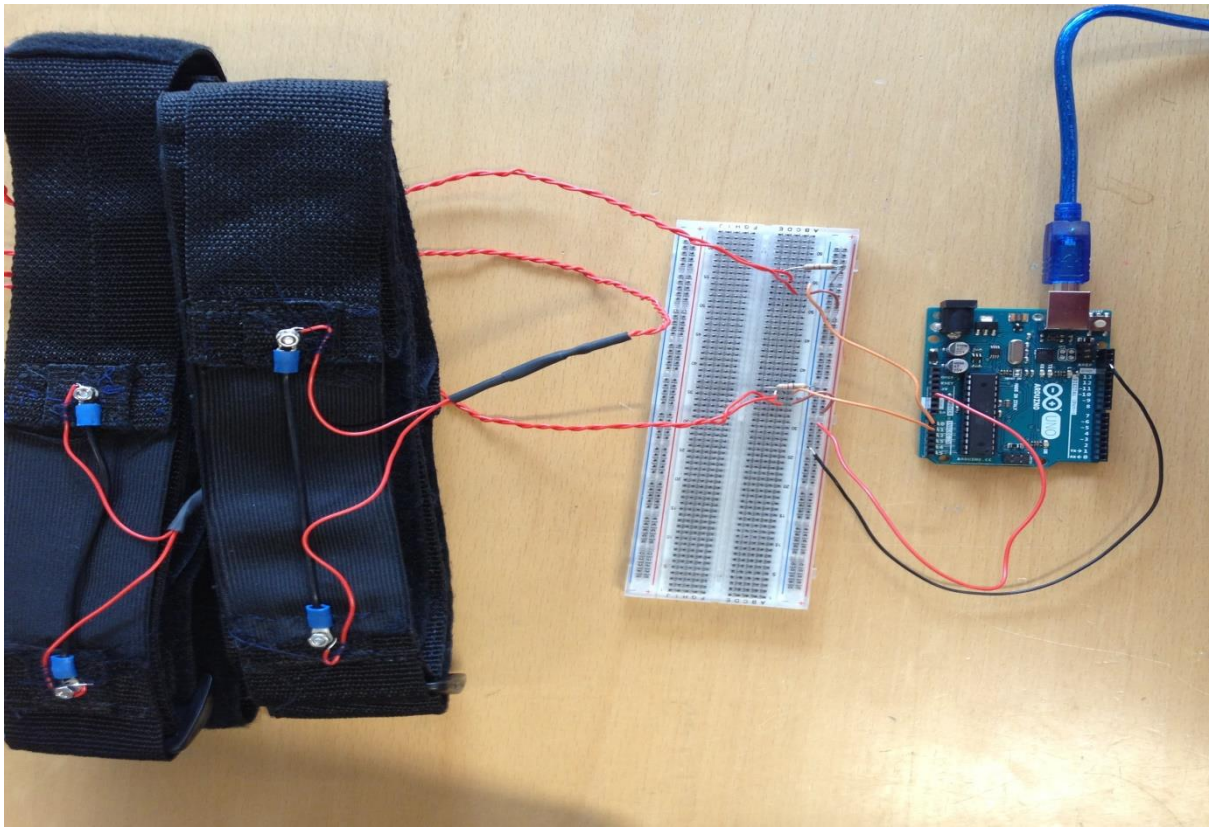


Figure 9-46 Complete Circuit

Figure 9-46 displays the circuit which was configured to facilitate the measurement of chest and abdominal respiratory movement. This circuit was made up of two variable resistors/respiratory belts and two pull-down resistors in parallel. As advised by Damon Berry, see section 4.6, Ohm's law was considered during the construction of the circuit. Ohm's Law is the relationship between the current (I) flowing through a resistance(R) and the potential drop across it (V), where current is measured in amps, resistance is measured in ohms (Ω) and potential drop/difference in volts. As a result of Ohm's law, pull-down resistors were required to both protect the Arduino from damage and to keep the analog voltage values within a range. In order to determine the resistance required by the pull-down resistors, the Axel Benz formula was used.

$$Rref = \sqrt{(Rmin * Rmax)}$$

This formula states that the value of the pull-down resistor is equal to the square root of minimum resistance multiplied by the maximum resistance of the variable resistor[51]. During the construction of the respiratory belts, the resistance of the 4" conductive rubber cord at rest and at a 70% stretch was recorded using a multimeter. The resistance of the conductive rubber cord at rest was 4600 Ω, and the resistance of the cord fully stretched was then found to be 12000 Ω. According to the Axel Benz formula, the value of the pull-down resistor required is 7430 Ω. Since the respiratory belts in the circuit are in parallel two pull-down resistors of this value were used.

This Arduino circuit configuration was then tested by using the serial monitor to verify that data was being read and changed sufficiently as the conductive rubber cord in the respiratory belts stretched and retract. Once confirmed correct, the data was collected by the Java program using the RXTX library and graphed using the JFreeChart library.

9.8 Problems Encountered

Throughout the implementation phase several problems and obstacles were encountered.

9.8.1 Respiratory Belt Data Inconsistencies

During the testing of the first respiratory belt prototypes, fluctuating and zero value data was being received from the respiratory belts. This was a problem as it made analysing the data difficult and the results inaccurate. This problem was fixed by only excepting values larger than zero and twisting the wires on the respiratory belts to reduce mechanical noise, reference section 9.6.3.

9.8.2 Entering ETCO2 Values During Respiratory Test

In the event of no ETCO2 value being supplied in the registration form, ETCO2 values needed to be collected during the respiratory test. The preferred method of collection was to prompt the user at each one minute interval for the current ETCO2 value. However, prompting for user input paused the respiratory test and affected the real-time graph. An attempt to fix the problem using multi-threading was performed. This attempt was unsuccessful as it caused multiple errors and caused the program to stop working for a period of time. Instead, a method of prompting for the ETCO2 values once the test had completed was implemented.

9.9 Conclusion

The Prototyping and Development chapter provided an overview of how the system was developed while following the agile software methodology. Several methods of testing were stated and undertaken at the end of each sprint. The results of these test determined the actions to be taken in the following sprint. However, not all of the system testing takes place at the end of sprints. The Testing chapter is the next chapter and it will describe the tests already taken and the remaining tests to be done in detail.

10 Testing

10.1 Introduction

The aim of the testing chapter is to present and explain the testing performed. The different types of testing will be split up into different sections based on when they occur and their purpose.

10.2 Agile Testing

Each of the testing methods mentioned below will be carried out at the end of each sprint. This agile testing method ensures functionalities are working correctly and satisfies user requirements before progressing to the next stage of the project.

10.2.1 Black Box Testing

Testing that ignores the internal mechanism of a system or component and focuses solely on the outputs generated in response to selected inputs and execution conditions[52]. It focuses on functional and non-functional requirements from a user's point of view.

Usability Testing

Description of 2 Tasks Completed by Each User		
Tast Number	**Description Of Task**	**User Score**
1	**Create a new account**	.
	Make obvious what is clickable	
	Easy to target clicks	
	Provide clear feedback	
	Easy to read	
	Text and background color contrast	
	Easy to find things	
	Use of user's language	
2	**Log into account**	.
	Make obvious what is clickable	
	Easy to target clicks	
	Provide clear feedback	
	Easy to read	
	Text and background color contrast	
	Easy to find things	
	Use of user's language	

Table 4 Usability Test Case Example

In this project, the usability testing method was adopted from online COPD respiratory therapy and education video resource centre[24]. Each user is provided with a set of common tasks. These tasks represent common exercises in the program. The user is then required to perform these tasks. Once complete the user is required to score the functionalities against a set of predefined heuristics with a score between zero and ten, with ten being good and zero being bad. Table 4 is an example of a set of tasks and heuristics which would be presented to

the user. After all users have successfully completed the tasks, the heuristic scores from each user are added to a report. This report will calculate the average score for each heuristic. Before starting the next sprint, heuristics corresponding to low values are brought to attention. Each heuristic in focus will be examined and changed within the functionalities. This new approach is also adopted in the next sprint and the feedback from this sprint will determine the success of the previously made changes. This process continues until all functionalities have been developed and users score high in each heuristic evaluation.

10.2.2 White Box Testing

Testing that takes into account the internal mechanism of a system or component[52]. It is often carried out by the developers as they have the greatest understanding of the source code.

Unit Testing

Testing of individual hardware or software units or groups of related units[52]. This form of testing will be performed to validate each of the different sections of code which make up individual functionalities. The units of code which will be testing can vary in size but generally are no larger than a class[53]. Only units of code which are of significant importance or can take variable inputs will be tested using this method due to time restrictions. The JUNIT Java library will be used to perform unit testing. JUNIT is a free tool used to perform unit testing more efficiently than a developer manually doing it. This will be used to verify that each aspect of the unit of code is correct. Such verifications include confirming all data structures are correct, verifying independent paths produce expected output and ensuring all error handling is sufficiently used. At the end of each sprint, a unit test plan will be written that will outline which sections of the recently developed functionalities are to be tested. Each unit of code to be tested will then be entered into the test case.

Test Case ID	Description	Input Data	Expected Result	Actual Result	Pass/Fail	Remarks
1	Enter numbers into name field	1234	The name field can only contain letters	The name field can only contain letters	Pass	.

Table 5 Unit Testing Test Case

Table 5 is an example of a common test case used in unit testing. It clearly states the description of the use case, the inputted data, expected results, actual results and if the unit of code passed of failed the test. The remarks section will only be used in the case of a fail. It will contain information on possible solutions to fix the problem.

Performance Testing

Testing conducted to evaluate the compliance of a system or component with specified performance requirements[52]. Performance testing such as load, stress, and testing are not required for this project as only a single user will be interacting with the system at a single time. However, performance testing is still an essential aspect of the development. Medical

professional are very busy and require the software they used to be fast and efficient. At the end of each sprint, the execution time for functionalities developed will be tested. This is achieved by using the Java built-in function System.currentTime.

Test Case ID	Description	Range (seconds)	Result	Pass/Fail	Remarks
1	Make new user account	0 - 15	10	Pass	.

Table 6 Performance Testing Test Case

Table 6 is an example of a test case used for this project. It clearly states the functionality description, range in which the execution time must fall, the result and if it passed or failed. Based on the functionality being tested an appropriate execution time range will be determined.

10.3 Testing Phase

Each of the following testing methods will be utilized in the testing phase at the end of the software development cycle.

Reliability Testing

Reliability testing is defined as testing the system's ability to perform its required functions under stated conditions for a specified period of time[52]. The system will be tested by using it and each of its functions repeatedly for a certain amount of time. In this space of time, the number of crashes or failures will be counted. The reliability of the system can be determined based on the number of crashes per hour.

Precision and Accuracy Testing

Precision is defined as the degree of exactness or discrimination with which a quantity is stated[52]. Precision testing will be used in this system to confirm that differences in chest and abdominal breathing are being measured correctly. To do this a patient with proven differences in chest and abdominal breathing and a patient with no form of DB will be sourced and tested.

Accuracy is defined as a qualitative assessment of correctness, or freedom from error[52]. This testing method will be used to confirm the correctness of data used in the system. This will be achieved by validating the data collected using the chest and abdominal respiratory frequency belts against the professional equipment used in Connelly Hospital.

Documentation Testing

This form of testing will be used to ensure that the program's corresponding documentation is up to the expected standard. In order to verify all spelling and grammar are correct, spell check will be activated while writing it and once completed a spelling and grammar checking

tools will be used. Finally, a manual inspection of the document will be completed with the aim of correcting inconsistencies and eliminating vagueness.

Portability Testing

Portability is defined as the ease with which a system or component can be transferred from one hardware or software environment to another[52]. The portability testing method that will be used in this case will involve setting up the system in a different environment while observing the time taken. If the time taken is too large then action will be taken to make the system more robust.

10.4 Usability Testing Results

This section will present the usability test cases and describe the results obtained throughout the project. Each usability test will be made up of a set of tasks and a set of predefined heuristics associated with each task. The user is then required to perform the specified tasks and give a score for each of the corresponding heuristics. Once complete, a brief interview will take place with the goal of identifying the reasoning behind the low scores and agreeing on action to be taken to correct the faults mentioned.

10.4.1 Usability Test #1

Description of 2 Tasks Completed by Each Use				
Tast Number	**Description Of Task**	**Novice User**	**Intermediate User**	**Average Score**
1	**Create a new account**	**Score**	**Score**	
	Make obvious what is clickable	5	5	5
	Easy to target clicks	4	6	5
	Provide clear feedback	0	0	0
	Easy to read	4	5	4.5
	Text and background color contrast	5	8	6.5
	Easy to find things	8	7	7.5
	Use of user's language	8	10	9
	Informative	5	4	4.5
2	**Log into account**	**Score**	**Score**	
	Make obvious what is clickable	7	6	6.5
	Easy to target clicks	4	5	4.5
	Provide clear feedback	0	0	0
	Easy to read	4	6	5
	Text and background color contrast	5	8	6.5
	Easy to find things	7	9	8
	Use of user's language	6	8	7
	Informative	7	9	8

Table 7 Usability Test Case One

Table 7 displays the first usability test which includes the tasks of creating a new account and logging into the new account and each user's scores which correspond to the predefined usability heuristics. The results of this test highlight areas which need to be improved in each task. The make obvious what is clickable, easy to target clicks, provide clear feedback, easy to read, text and background colour contrast and informative heuristics of the "create a new account" task need to be brought to focus and improved. Additionally, the easy to target

clicks, provide clear feedback, easy to read and text and background colour contrast heuristics of the "log into account" task also need to be brought to focus and improved.

Description of 2 Tasks Completed by Each User		
Tast Number	**Description Of Task**	**Action Taken**
1	**Create a new account**	
	Make obvious what is clickable	Increase button size, border size and font size
	Easy to target clicks	Increase button size
	Provide clear feedback	Create input testing functionality to provide precise feedback
	Easy to read	Change font to Sans Scipt and increase font size
	Text and background color contrast	Distinguish form boxes. Add background
	Informative	Specify measurement units were required. Use combo boxes
2	**Log into account**	
	Easy to target clicks	Increase button size
	Provide clear feedback	Create input testing functionality to provide precise feedback
	Easy to read	Change font to Sans Scipt and increase font size
	Text and background color contrast	Distinguish form boxes. Add background

Table 8 Usability Test Actions One

Table 8 demonstrations the actions taken against the highlighted heuristics for each task. The success or failure of these actions will be determined by the results of the next usability test.

10.4.2 Usability Test#2

Description of 3 Tasks Completed by Each User				
Tast Number	**Description Of Task**	**Novice User**	**Intermediate User**	**Average Score**
1	**Create a new account**	**Score**	**Score**	
	Make obvious what is clickable	7	8	7.5
	Easy to target clicks	8	9	8.5
	Provide clear feedback	9	8	8.5
	Easy to read	7	8	7.5
	Text and background color contrast	5	8	6.5
	Informative	7	7	7
2	**Log into account**	**Score**	**Score**	
	Easy to target clicks	8	7	7.5
	Provide clear feedback	9	9	9
	Easy to read	7	8	7.5
	Text and background color contrast	5	8	6.5
3	**Complete the Nijmegan Questionnaire**	**Score**	**Score**	
	Make obvious what is clickable	8	7	7.5
	Easy to target clicks	5	6	5.5
	Provide clear feedback	9	10	9.5
	Easy to read	7	9	8
	Text and background color contrast	7	8	7.5
	Easy to find things	9	10	9.5
	Use of user's language	6	10	8
	Informative	8	9	8.5

Table 9 Usability Test Case Two

Table 9 displays the usability test for the tasks of creating a new account, logging into the new account and completing the Nijmegen Questionnaire. The users scored each task against a set of predefined heuristics. The previously tested "create a new account" and "log into account" tasks were tested again and confirmation that the changes made have been a success is evident. Additionally, the results of the "Complete the Nijmegen Questionnaire" task are

present. Users scored highly for the heuristics associated with the "Complete the Nijmegen Questionnaire" task as information obtained in the prior usability test was utilised in the development of the "Nijmegen Questionnaire" page. However, the easy to target clicks heuristic obtained a low enough average score to highlight it for attention.

Description of 1 Tasks Completed by Each User		
Tast Number	Description Of Task	Action Taken
1	Complete Nijmegen Questionnaire	
	Easy to target clicks	Cursor change when hovering over target item

Table 10 Usability Test Actions Two

Table 10 presents the action that will be taken to try to increase the usability of the "Nijmegen Questionnaire" page. The effectiveness of this change will be determined by the results of the next usability test.

10.4.3 Usability Test #3

Description of 2 Tasks Completed by Each User					
Tast Number	Description Of Task	Novice User	Intermediate User	Respiratory Professional	Average Score
1	Complete the Nijmegan Questionnaire	Score	Score	Score	
	Easy to target clicks	7	9		8
2	Start Respiratory Test and View Real-Time Graph				
	Make obvious what is clickable			5	5
	Easy to target clicks			9	9
	Provide clear feedback			7	7
	Easy to read			8	8
	Text and background color contrast			9	9
	Easy to find things			8	8
	Use of user's language			9	9
	Informative			4	4

Table 11 Usability Test Case Three

Table 11 shows the usability test for the tasks of completing the Nijmegen questionnaire, starting the respiratory test and viewing the real-time graph. The low scoring heuristic for the "Complete the Nijmegen Questionnaire" task in the previous usability test was tested again. As an increase in the average heuristic score for this task is shown it can be concluded that the actions taken were a success. The "Start Respiratory Test and View Real-Time Graph" task will only ever be undertaken by the respiratory professional, meaning the novice and intermediate users do not need to test the task. In this task, the respiratory professional highlighted the "make obvious what is clickable" and "informative" heuristics as areas which need improvement.

Description of 1 Tasks Completed by Each User		
Tast Number	Description Of Task	Action Taken
1	Start Respiratory Test and View Real-Time Graph	
	Make obvious what is clickable	Only display show graph button when graph is ready
	Informative	provide respiratory belt application demo and instruction

Table 12 Usability Test Actions Three

Table 12 presents the actions which are to be taken in order to increase the average score for the specified heuristics in the next round of usability testing. The effectiveness of these changes will be displayed in the next usability test.

10.4.4 Usability Test#4

Description of 2 Tasks Completed by Each User					
Tast Number	Description Of Task	Novice User	Intermediate User	Respiratory Professional	Average Score
1	Start Respiratory Test and View Real-Time Graph	Score	Score	Score	
	Make obvious what is clickable			8	8
	Informative			9	9
2	Complete Breathing Retraining Session and Watch Videos	Score	Score	Score	
	Make obvious what is clickable	5	6		5.5
	Easy to target clicks	8	8		8
	Provide clear feedback	0	0		0
	Easy to read	7	8		7.5
	Text and background color contrast	8	6		7
	Easy to find things	5	6		5.5
	Use of user's language	6	9		7.5
	Informative	7	9		8

Table 13 Usability Test Case Four

Table 13 displays the results of the usability test associated with the tasks of starting the respiratory test, viewing the real-time graph and completing the breathing retraining session while remembering to watch the videos included. The “Start Respiratory Test and View Real-Time Graph” task’s two low scoring heuristics of the previous usability test were tested again. The resulting high average scores for both the heuristics concludes that the actions taken to amend the faults were a success. Additionally, the “make obvious what is clickable”, “provide clear feedback” and “easy to find things” heuristics of the “Complete Breathing Retraining Session and Watch Videos” tasks were brought to attention.

Description of 1 Tasks Completed by Each User		
Tast Number	Description Of Task	Action Taken
1	Complete Breathing Retraining Session and Watch Videos	
	Make obvious what is clickable	Inform user of play button for videos and removew arrow
	Provide clear feedback	Inform user when test is complete
	Easy to find things	Bring focus to the play video button

Table 14 Usability Test Actions Four

Table 14 demonstrates the actions that are to be taken in order to resolve the issues associated with the low scoring heuristics. The outcome of these actions will be determined through the results of the subsequent usability test.

10.4.5 Usability Test#5

Description of 1 Tasks Completed by Each User				
Tast Number	Description Of Task	Novice User	Intermediate User	Average Score
1	**Complete Breathing Retraining Session and Watch Videos**	**Score**	**Score**	
	Make obvious what is clickable	7	9	8
	Provide clear feedback	9	9	9
	Easy to find things	7	8	7.5

Table 15 Usability Test Case Five

Table 15 presents the results of the usability test for the task of completing the breathing retraining session and watching the associated videos throughout the duration of the session. In this usability test the low scoring heuristics associated with the "Complete Breathing Retraining Session and Watch Videos" task from the previous test are tested again. With significant improvement clearly visible it can be concluded that the actions taken to amend the low scoring heuristics have been a success.

10.4.6 Usability Test#6

Description of 1 Tasks Completed by Each User		
Tast Number	Description Of Task	Respiratory Professional
1	**View Report**	**Score**
	Informative	3
	Clear structure	4
	Easy to read	5
	Easy to find things	5
	Use of user's language	8

Table 16 Usability Test Case Six

Table 16 displays the usability test results for the respiratory professional's task of analysis. The findings from this test are significant as four out of five heuristics obtained a score of five or lower.

Description of 1 Tasks Completed by Each User		
Tast Number	Description Of Task	Action Taken
1	**View Report**	
	Informative	Only show session totals and comparison data
	Clear structure	Replace raw data with session totals and session comparison
	Easy to read	Structure data in table format
	Easy to find things	Label important values with bolt text

Table 17 Usability Test Actions Six

Table 17 shows the proposed actions to be taken in order to correct the low scoring heuristics. The success or failure of these actions will be outlined in the next usability test.

10.4.7 Usability Test#7

Description of 1 Tasks Completed by Each User		
Tast Number	Description Of Task	Respiratory Professional
1	View Report	Score
	Informative	7
	Clear structure	8
	Easy to read	8
	Easy to find things	7

Table 18 Usability Test Case Seven

Table 18 presents the results of the follow up usability test for the "View Report" task. Each low scoring heuristic was scored again following the changes made as a result of the previous test. It can be concluded that the actions taken have been a success as significant improvements can be seen in each heuristic.

10.5 Unit Testing Results

This section will describe and present the unit tests undertaken. The findings and resulting actions taken for each unit test will be discussed. Additionally, the effectiveness of the actions taken will be tested in order to increase the robustness of the system. Each selected unit of code is tested using the JUNT unit testing framework. An example of a unit test which was developed using the JUNIT framework and used within the project is as follows.

```
@Test
public void testSignin() {
    System.out.println("signin");
    String username = "cf123";// input values
    String password = "pass";
    agileTest instance = new agileTest();
    String expResult = "You Have Successfully Signed In";// expected
    result
    String result = instance.signin(username, password);
    assertEquals(expResult, result); // checking if true or false
}
```

Figure 10-1 Unit Test Code Example

Figure 10-1 presents a code example of a unit test which tests the login "signin" function. The value "cf123" is entered as the username and the value "pass" is entered as the password. These values are then passed to the "signin" function. If the username and password values are correct the "signin" function will return "You Have Successfully Signed In" and the test will pass. If either the username or password values are incorrect the "signin" function will return a different result and the test will fail.

10.5.1 Login Function Unit Test

The unit test case for the login function, reference section 13.1, was a success as all unit tests passed. The initial code developed for checking the username and password against data

stored in the database was sufficient to pass the tests. Limited data validation is required as data entered either returns true or false.

10.5.2 Register Function Unit Test #1

The first unit test case register function, reference section 13.2, brought some data validation requirements to attention. Four tests relating to preventing empty strings being submitted failed. It was identified that these failures were due to checking if the text field's value was equal to null. The method of checking if the length of the text field's value is smaller than one was identified as a potential fix.

10.5.3 Register Function Unit Test #2

The second unit test case for the register function, reference section 13.3, was successful as all tests passed. This unit test case confirmed that the method for checking blank fields suggested previously was a success. Further attempts to break the data validation were made with no success. Tested register function now has a considerably high level of protection against unclean data entering the database and malicious attacks.

10.5.4 Respiratorytest4 Class Unit Test #1

The first unit test case for the respiratorytest4 class, reference section 13.4, produced mostly positive results. However, the condition of an if statement caused a failure to occur. This if statements correctly verified that two string were being received but failed to check the length of the received strings. As a result, empty strings were allowed to pass and cause an error in the code. The method of checking the length of each string as well as the array holding the values was decided.

10.5.5 Respiratorytest4 Class Unit Test #2

Prior to this unit test, reference section 13.5, the previously decided method of action was implemented. This changed was determined by the results of the previous unit test and involved adding functionality to prevent empty strings from being passed into the respiratorytest4 class. The result of this unit test confirms that the action taken has successfully corrected the flaw.

10.6 Performance Testing Results

This section will present and explain the results of the performance tests undertaken throughout the implementation phase. Each performance test is performed at the end of a sprint in order to determine if the performance time of the newly created functionalities is sufficient. In the case of a failure, the corresponding function will be edited to improvement performance.

10.6.1 Performance Test #1

Performance Testing					
Test Case ID	Description	Range (seconds)	Result (seconds)	Pass/Fail	Remarks
1	Make new user account	3	1.36	Pass	.
2	Sign in	3	2	Pass	.

Table 19 Performance Test Case One

Table 19 presents the results of the performance test for the account creation and login functionalities. Both functionalities passed the test by at least a third of the allowed time.

10.6.2 Performance Test #2

Performance Testing					
Test Case ID	Description	Range (seconds)	Result (seconds)	Pass/Fail	Remarks
1	Submit NQ	3	1.1	Pass	.

Table 20 Performance Test Case Two

Table 20 displays the performance testing results for the functionality of submitting the Nijmegen questionnaire answers. This test case was a success with the functionality passing with a time one-third of the allowed time.

10.6.3 Performance Test #3

Performance Testing					
Test Case ID	Description	Range (seconds)	Result (seconds)	Pass/Fail	Remarks
1	Load belt application tutorial	2	1.8	Pass	.
2	Load pursed lip video	2	1.6	Pass	.
3	Load diaphramic breathing video	2	1.7	Pass	.
4	Analyse respiratory test data to db	3	2.1	Pass	.

Table 21 Performance Test Case Three

Table 21 shows the findings of the third performance test undertaken. Each of the functionalities tested executed fully within the time range allocation and passed the test.

10.6.4 Performance Test #4

Performance Testing					
Test Case ID	Description	Range (seconds)	Result (seconds)	Pass/Fail	Remarks
1	Insert each breathe data to db	10	7	Pass	.
2	insert interval data db	3	2.5	Pass	.
3	insert session data to db	2	1.3	Pass	.
4	generate report	2	1.5	Pass	.

Table 22 Performance Test Case Four

Table 22 displays the results of the final performance test. The results of this performance test show that all the tested functionalities passed as they executed fully within the allowed time.

10.7 Precision Testing

Precision testing was performed to test if the system has the ability to correctly measure differences between chest and abdominal breathing. Each test used the respiratory test functionality within the program and lasted the duration of fifteen minutes. The report generated was used to hold the results of each of the tests.

Precision testing was to be performed on a patient with a known case of predominant chest breathing. Unfortunately, this patient cancelled the appointment with little notice. As a result, a patient with predominant chest breathing was sourced due to time restrictions. Instead, precision testing was performed by mimicking the breathing pattern associated with predominant chest breathing.

	Session One			Session Two		
	Abdominal	*Chest*	*Overalll*	*Abdominal*	*Chest*	*Overalll*
Minute 5 - 6	1.7	9.6	7.8	5.2	10	4.8
Minute 6 - 7	2.4	11	8.6	5.1	12	6.5
Minute 7 - 8	2.8	11	8.6	5.5	11	5.3
Minute 8 - 9	3	10	7.5	5.1	11	6
Minute 9 - 10	4.5	15	10	5.6	14	8.4
Minute 10 - 11	4	12	8.1	4.9	9.6	4.6
Minute 11 - 12	2.5	12	9.5	5.2	8.8	3.5
Minute 12 - 13	3.3	11	8.1	5.5	11	5.1
Minute 13 - 14	3.9	10	6.4	5.1	8.9	3.8
Minute 14 - 15	1.7	6.4	4.7	3.2	6.7	3.5
Session Total	0.12	0.45	0.33	0.19	0.37	0.19

Table 23 Predominant Chest Breathing Precision Test Results

Table 23 displays the results for the predominant chest breathing precision test. During the first session, a breathing pattern of a person with a case of predominant chest breathing was mimicked. The results of session one conclude that a significant difference exists between the average expansion of the chest and abdomen. During the second session, a breathing pattern considered to be normal was practiced. The results of this session show that the average difference between the chest and abdominal respiratory expansion is almost half the size of the difference in the first session.

This test was considered a success as the system was able to correctly measure and present when a predominant chest breathing pattern existed. Additionally, the system also clearly showed the improvement in the breathing pattern associated with the second session.

10.8 Accuracy Testing

Accuracy testing was performed to test the accuracy of the respiratory rate values being generated during the respiratory test. The accuracy testing was to be performed in Connelly Hospital using a BCI 9004-000 Capnocheck Plus Capnograph. However, due to complications this visit was no possible. A new approach for testing the accuracy of the respiratory rate calculation was developed. During the respiratory test, each breath is counted manually and this value acts as the expected result. Using the difference between the expected and actual results a level of accuracy can be determined.

Accuracy Testing				
Test Case ID	Description	Expected RR	Result RR	Difference
1	Breathing normally	15	16	1
2	Breathing normally	13	13	0
3	Breathing normally	14	14	0

Table 24 Accuracy Test Case One

Table 24 presents the accuracy test case for the calculation of respiratory rate while breathing normally. The results of this test case conclude that the program has the ability to accurately measure normal respiratory rates with only a small margin of error.

Accuracy Testing				
Test Case ID	Description	Expected RR	Result RR	Difference
1	Breathing fast	25	23	2
2	Breathing fast	22	22	0
3	Breathing fast	24	21	3

Table 25 Accuracy Test Case Two

Table 25 displays the accuracy test case which tests the accuracy of the calculated respiratory rate while a fast respiratory rate is practiced. The results show that as the respiratory rate increases the accuracy of the calculated respiratory rate decreases. With 22-25 breaths being counted in each test, this respiratory rate is at the extreme end of over breathing. In the second test, the expected and actual respiratory rates matched. This shows that even at the extreme end of over breathing the program has the ability to calculate the correct respiratory rate.

Accuracy Testing				
Test Case ID	Description	Expected RR	Result RR	Difference
1	Breathing slow	6	10	4
2	Breathing slow	9	14	5
3	Breathing slow	8	15	7

Table 26 Accuracy Test Case Three

Table 26 shows the third accuracy test case which presents the results of the accuracy test while a slow breathing pattern is being practiced. The results conclude that the programs accuracy decreases significantly as the respiratory rate decreases. The respiratory rates tested are very low and would rarely, if ever, presented during the respiratory test. The results of this test conclude the code developed is no suited best for lower respiratory rates.

10.9 Portability Testing

Portability testing was performed to test if the program developed was in fact platform independent. Measures were taken to ensure that the system developed was platform independent. Such measures include using the Java programming language and providing the ports used by Linux and Mac OS in the RXTX serial port connection method.

Portability Testing		
Test Case ID	Description	Pass/Fail
1	Run program on Windows OS	Pass
2	Run program on Mac OS	Pass
3	Run program on Linux Os	Pass

Table 27 Portability Test Case

Table 27 presents the results of the portability testing. The system was successfully able to run on each of the operating systems tested. The results of this test case conclude that the actions taken to ensure that the system was platform independent were successful.

10.10 Documentation Testing

Documentation testing was performed on the text for each of the help options. These sections of text were tested using Microsoft Word's spellcheck and Grammarly. Grammarly is a tool

used to check spelling and grammar. The results of the test were positive as no spelling or grammar mistakes were discovered in any of the testing documentation.

10.11 Conclusion

The Testing chapter presented and explained the results for each of the different types of testing performed during the implementation and testing phases. The testing performed in the agile testing section was performed at the end of each sprint and helped develop the system through user interaction and code validation. Finally, the testing performed in the testing phase section was performed once the implementation phase was complete. This phase of testing was used to determine the robustness, flexibility, accuracy and precision of the software developed. The next chapter will outline the project plan and describe the tasks performed in each of the sprints.

11 Project Plan

11.1 Introduction

In this section, the project plan will be presented in the form of a Gantt chart. This Gantt chart will consist of a number of sprints as this project as opted to use the agile software development methodology. The work that is expected to be completed in each sprint will be stated.

11.2 Project Plan

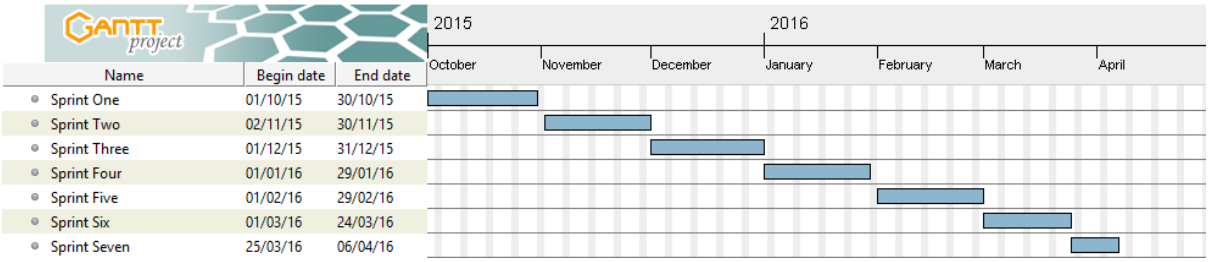


Figure 11-1 Project Gantt Chart

Sprint One

- Research DB causes, diagnosis and treatment.
- Compare similar systems as to identify users and usability requirements.
- Research and choose technologies to be used.
- Develop and deploy NQ.

Sprint Two

- Source medical professional in the area and conduct meetings.
- Identify problem and solution.
- Define functional and non-functional requirements.
- Choose software development methodology.

Sprint Three

- Create system architecture and prototypes.
- Produce test plan.
- Outline risks and issues and plan for the future.
- Finalize report.

Sprint Four

- Create Oracle relational database.
- Construct chest and abdominal respiratory frequency monitor using Arduino.

Sprint Five

- Develop client side application including GUI, NQ, database and Arduino communications; and other low-level functionalities.
- Develop testing process and interface which includes real-time graphics.

Sprint Six

- Create breathing retraining session.
- Write data processing algorithm and generate report.

Sprint Seven

- Testing phase.
- Project finalization.

11.3 Conclusion

A total of seven sprints will be performed throughout the entire timeline of the project. Each of these sprints has specific tasks which need to be complete within the allocated time.

12 Conclusions

12.1 Introduction

This section aims to present the project's main findings, provide a goal and objectives review, provide a personal statement and describe the work which will be undertaken in the future with the project.

12.2 Main Findings

Throughout the several stages in the completion of the project much has been discovered. It has become evident that DB is a vague condition which requires more attention and research from medical and scientific communities. This can be proven by the limited resources available relating to the condition. Such limited resources results in little understanding of the condition and has made progress towards finding a standard diagnosis method slow. The NQ has shown its worth by acting as a measure for the benefit of breathing retraining in DB patients in recent studies. It has also proved the importance of DB awareness as 22% of people, who participated in the NQ distributed, obtained a score of over 23 which indicates a positive reading for hyperventilation syndrome. Through meetings with Aisling McGowan, the hyperventilation characteristic of dominant chest breathing over abdominal breathing was acquired. ETCO2 levels have also been identified as an important measurement but it must be in conjunction with other positive readings as it commonly presents normal readings in DB patients. This is due to the body's ability to regulate gas concentrations.

Identifying functional and non-functional requirements has proven difficult as no similar system currently exists. A unique method of diagnosis was developed through a meeting with Aisling McGowan and subsequently defined the functional requirements. Upon discovering DB is more prevalent in older patients and is somewhat related to COPD, non-functional requirements could be derived from systems used by older users and medical professional as well as utilizing results from a heuristic evaluation of a COPD educational website.

The use of the Java programming language proved best suited for this project as it is flexible, offers database and Arduino communications, enables the use of real-time charting and provides report generation capabilities. Communication between the Java application and the database will be achieved through the use of the JDBC library. An Oracle relational database will be developed using Oracle SQL. Communication between the Java application and the Arduino will be made possible using the RXTX library. Arduino is the best option, in terms of minicomputers, as it is best able to provide sensor reading functionality. The constructed chest and abdominal respiratory belts will be used to take all necessary reading throughout the respiratory test processes. The JFreeChart library will be used to generate and display real-time graphs using the data generated during the respiratory test process. Once all data has been gathered the Report Mill library will generate a structured report of all findings.

The agile development method was determined the best approach in this project because no similar systems exist. With no DB systems to reference it is unavoidable and required that

frequent feedback and changes are made. It has also become apparent that there are many risks associated with this project. Factors such as data confidentiality agreements and ethical agreements will need to be organised due to the nature of working with patients and their sensitive information.

With no similar systems or standard diagnosis method available, a creative approach was needed. This involved the development of a unique method. This method consists of recording chest and abdominal RR; and CO2 levels before and after breathing retraining and comparing results. I believe this method has the ability to present evidence of DB in patients through recording the benefits of breathing retraining and will encourage and help others research further in this area.

The Analysis chapter defined the environment in which the system is to be used. Factors such as the patient's required condition and the medical professional's influence on the test were identified. Additionally, the flow of the system and tasks performed by each user were specified.

The Design chapter presented the simple to read and understand system designs developed. Each design created specified how a different part of the system was going to be implemented. The designs created simplified the implementation process as they were reference regularly.

The Implementation chapter displayed the different stages involved in the implementation process while utilising the agile software development methodology. Each of the functionalities implemented during a sprint were tested before beginning the next sprint. This ensured the functionalities met the required standard before progressing further in the implementation process.

The Testing chapter presented the test cases and test results for the testing performed at the end of sprints and during the testing phase. The usability testing performed helped to shape the Java application's GUI and ensured the GUI's was user focused. Unit testing and performance ensured that each of the functionalities performed correctly and quickly before progression further into the implementation. Accuracy and precision testing confirmed that the system was able to identify predominant chest breathing and that the data being calculated was accurate. Portability testing results proved that the system is indeed flexible and platform independent. Finally, documentation testing provided confidence that the text provided in the systems help documentation included correct spelling and grammar.

The Project Plan chapter presented the timeline which was followed through the project. Each of the required tasks were split up into realistic sprints. This project plan was followed strictly throughout the project and resulted in the goal and each of the objectives being achieved within the allocated time.

12.3 Goal and Objectives Review

At the beginning of this project, a realistic goal and subsequent objectives were set. Each of these objectives were worked towards and achieved throughout the project. As a result, the

goal of developing an objective method of measuring and displaying differences between chest and abdominal expansion was achieved.

12.4 Personal Reflection

At the beginning of this project, I was excited to further my knowledge in the area of DB and software development. I felt eager but nervous at the same time as I decided to take on a project which involved not only computer science but also engineering and respiratory science components. Respiratory science was my biggest weakness when starting this project. I had little knowledge in the area but found it very interesting and wanted to learn more about it while combining it with my passion for computer science. Through hard work and help from Aisling McGowan, I greatly improved my knowledge in the area. Throughout the different stages involved in the completion of this project, I feel I have been challenged and have learned a lot from overcoming each obstacle presented. I have improved my project and time management skills greatly throughout the project. Additionally, I have bettered my engineering knowledge and gained an extensive understanding and passion for respiratory science. On completion of this project, I now believe I have the skills required to begin my career in computer science and have confidence in my abilities. With this new confidence, I know I will be an asset to future software development teams.

12.5 Future Work

12.5.1 Improve Registration Form

During a meeting with Aisling McGowan, reference section 4.9, Aisling stated the registration form needed to include the patient's address and date of birth in addition to their full name. These three pieces of information act as a three-point identification method. Aisling also suggested generating a body mass index using the inputted weight and height values. Additionally, the patient's sex, current diagnosis, reason for test and medications also need to be added to the form to provide all the necessary information. Due to time limitations, these features did not make it into the project but will be added into the application in the future.

12.5.2 Inductance Plethysmography Belts

Unfortunately, inductance plethysmography belts could not be obtained for this project due to financial and time restrictions. For future work, these belts will be acquired and used as they provide more accurate results with the added dimension of providing respiratory volume.

12.5.3 Test Patients

During the project, a patient with a known case of predominant chest breathing was found and asked to come to Connelly hospital to be tested using the system developed. However, this patient cancelled on the day of the appointment. It is planned to source multiple patients who display symptoms of DB and test them using the software. The results of these tests will be examined and displayed in a scientific report.

12.5.4 Improve Accuracy for Slower RR

During the testing phase, results from accuracy testing showed that the system's accuracy decreased as extremely low RR were presented. In order to improve accuracy in this area, the

algorithm for analysing that data will be modified. Further testing will take place until a satisfactory level of accuracy is achieved.

12.5.5 Addition of Biofeedback

Biofeedback during the breathing retraining session was a feature that was to going to be implemented in this project, time permitting. However, this feature did not make it into the project. When continuing on with the development of this system, this feature will be added to inform that patient when they are performing the described breathing technique correctly.

Bibliography

[1] C. Hagman, C. Janson, and M. Emtner, "Breathing retraining - A five-year follow-up of patients with dysfunctional breathing," *Respir. Med.*, vol. 105, no. 8, pp. 1153–1159, 2011.

[2] S. D. Aaron, K. L. Vandemheen, L.-P. Boulet, R. A. McIvor, J. M. Fitzgerald, P. Hernandez, C. Lemiere, S. Sharma, S. K. Field, G. G. Alvarez, R. E. Dales, S. Doucette, and D. Fergusson, "Overdiagnosis of asthma in obese and nonobese adults.," *CMAJ*, vol. 179, no. 11, pp. 1121–31, 2008.

[3] Physiopedia, "Breathing Pattern Disorders - Physiopedia, universal access to physiotherapy knowledge." [Online]. Available: http://www.physio-pedia.com/Breathing_Pattern_Disorders. [Accessed: 03-Dec-2015].

[4] M. Thomas, R. K. McKinley, E. Freeman, and C. Foy, "Prevalence of dysfunctional breathing in patients treated for asthma in primary care: cross sectional survey.," *BMJ*, vol. 322, no. 7294, pp. 1098–100, 2001.

[5] M. D. L. Morgan, "Dysfunctional breathing in asthma: is it common, identifiable and correctable?," *Thorax*, vol. 57 Suppl 2, no. Suppl II, pp. II31–II35, 2002.

[6] H. K. Hornsveld, B. Garssen, M. J. C. Fiedeldij Dop, P. I. Van Spiegel, and J. C. J. M. De Haes, "Double-blind placebo-controlled study of the hyperventilation provocation test and the validity of the hyperventilation syndrome," *Lancet*, vol. 348, no. 9021, pp. 154–158, 1996.

[7] M. Thomas, R. K. McKinley, E. Freeman, C. Foy, P. Prodger, and D. Price, "Breathing retraining for dysfunctional breathing in asthma: a randomised controlled trial," *Thorax*, vol. 58, no. 2, pp. 110–115, 2003.

[8] B. F. Palmer, "Evaluation and treatment of respiratory alkalosis.," *Am. J. Kidney Dis.*, vol. 60, no. 5, pp. 834–8, Nov. 2012.

[9] C. Hagman, C. Janson, and M. Emtner, "A comparison between patients with dysfunctional breathing and patients with asthma," *Clin. Respir. J.*, vol. 2, no. 2, pp. 86–91, 2008.

[10] J. C. Hannon, *Recognizing and Treating Breathing Disorders*. 2014.

[11] A. Mc Gowan, "Aisling Mc Gowan Email," vol. 1. p. 2, 2015.

[12] J. Van Dixhoorn and H. Folgering, "The Nijmegen Questionnaire and dysfunctional breathing," pp. 3–6, 2015.

[13] D. H. Van Dixhoorn J, "Hyperventilatieklachten in de fysiotherapiepraktijk.pdf," *Ned T Fysiother.*, no. 95–7/8, pp. 167 – 171, 1985.

[14] M. Thomas, R. K. McKinley, E. Freeman, C. Foy, and D. Price, “The prevalence of dysfunctional breathing in adults in the community with and without asthma,” *Prim. Care Respir. J.*, vol. 14, no. 2, pp. 78–82, 2005.

[15] D. H. Van Dixhoorn J, “Efficacy of Nijmegen Questionnaire in recognition of the hyperventilation syndrome.,” no. 29–2, pp. 199–206, 1985.

[16] J. Van Dixhoorn and H. Folgering, “De Nijmeegse vragenlijst en dysfunctioneel ademen,” pp. 2–5, 2015.

[17] R. Courtney, J. van Dixhoorn, and M. Cohen, “Evaluation of Breathing Pattern: Comparison of a Manual Assessment of Respiratory Motion (MARM) and Respiratory Induction Plethysmography,” *Appl. Psychophysiol. Biofeedback*, vol. 33, no. 2, pp. 91–100, 2008.

[18] T. T. Kean and M. B. Malarvili, “Analysis of capnography for asthmatic patient,” *2009 IEEE Int. Conf. Signal Image Process. Appl.*, pp. 464–467, 2009.

[19] J. Spahija, “Effects of Imposed Pursed-Lips Breathing on Respiratory Mechanics and Dyspnea at Rest and During Exercise in COPD,” *CHEST J.*, vol. 128, no. 2, p. 640, 2005.

[20] “Diaphragmatic breathing | definition of diaphragmatic breathing by Medical dictionary.” [Online]. Available: http://medical-dictionary.thefreedictionary.com/diaphragmatic+breathing. [Accessed: 23-Nov-2015].

[21] S. DeGuire, R. Gevirtz, D. Hawkinson, and K. Dixon, “Breathing retraining: A three-year follow-up study of treatment for hyperventilation syndrome and associated functional cardiac symptoms,” *Biofeedback Self. Regul.*, vol. 21, no. 2, pp. 191–198, Jun. 1996.

[22] M. Thomas and A. Bruton, “Breathing exercises for asthma,” no. June, 2014.

[23] C. a Slader, H. K. Reddel, L. M. Spencer, E. G. Belousova, C. L. Armour, S. Z. Bosnic-Anticevich, F. C. K. Thien, and C. R. Jenkins, “Double blind randomised controlled trial of two different breathing techniques in the management of asthma.,” *Thorax*, vol. 61, pp. 651–656, 2006.

[24] M. Stellefson, B. Chaney, and D. Chaney, “Heuristic Evaluation of Online COPD Respiratory Therapy and Education Video Resource Center,” *Telemed. e-Health*, vol. 20, pp. 972–976, 2014.

[25] J. Gosling and A. Buckley, “The Java ™ Language Specification Java SE 7 Edition,” *Oracle*, 2011.

[26] E. H. Perry, B. Wright, T. Pfaeffle, M. Bastawala, L. Chidambaran, S. Krishnaswamy, R. Irudayaraj, S. Urman, S. Kraft, S. Maring, and A. Long, “Oracle ® Database JDBC Developer ’ s Guide and Reference,” vol. 1, no. December, 2003.

[27] H. a L. Caswell, *The C Programming Language Second Edition*. 1988.

[28] J. M. Redondo and F. Ortin, “A Comprehensive Evaluation of Common Python Implementations,” *Software, IEEE*, vol. 32, no. 4, pp. 76–84, 2015.

[29] D. Lorentz, M. B. Roeser, S. Abraham, A. Amor, G. Arora, V. Arora, L. Ashdown, G. Eadon, A. Ganesh, B. Glover, N. Gopal, M. Hallas, M. Ho, C. Iyer, M. Jaeger, V. Kapoor, P. Knaggs, S. Krishnaswamy, A. Kruglikov, P. Lane, H. Li, Y. Li, V. Liang, B. Llewellyn, R. Long, S. Lynn, V. Marwah, J. Matsuzawa, R. Mcguirk, R. Mir, G. Mulagund, S. Muthulingam, H. Qian, A. Ray, J. Russell, L. Schneider, V. Schupmann, J. Shi, A. Singh, W. Smith, S. Song, V. Srihari, J. Stenoish, S. Subramanian, S. Sundara, M. van de Wiel, B. Varanasi, W. Waddington, P. Wahl, C. Wetherell, S. Wolicki, D. Wong, T. Yu, M. Zait, F. Zemke, W. Zhang, and W. Zhang, “Oracle ® Database SQL Language Reference 11g Release 2 (11.2),” vol. 2, no. July, 2014.

[30] I. MongoDB, “MongoDB documentation,” 2015.

[31] T. Postgresql and G. Development, “PostgreSQL 9.1.18 Documentation.”

[32] E. Upton and Raspberry Pi Foundation, “Programming the Raspberry Pi,” *Www.Element14.Com*, p. 41, 2013.

[33] W. Durfee, “Arduino Microcontroller Guide,” pp. 1–27, 2011.

[34] E. Gee, “Basic Belt Respiration Sensor,” 2015. [Online]. Available: http://www.instructables.com/id/Quick-and-dirty-Respiration-Sensor/. [Accessed: 18-Feb-2016].

[35] RobotShop, “1m Flexible Stretch Sensor Cord - RobotShop.” [Online]. Available: http://www.robotshop.com/uk/1m-flexible-stretch-sensor-cord.html. [Accessed: 18-Feb-2016].

[36] COZIR, “Ultra Low Power Carbon Dioxide Sensor,” pp. 5–7, 2012.

[37] BCI, “BCI 9004-000 Capnocheck Plus Capnograph.” [Online]. Available: https://www.a1medicalsales.com/product/9004-000.html. [Accessed: 16-Nov-2015].

[38] T. O. Connor and C. R. Physician, “Spirometry : Performance and Interpretation A Guide for General Practitioners,” no. November, 2005.

[39] J. Wanger, “Standardisation of the measurement of lung volumes,” *Eur. Respir. J.*, vol. 26, no. 3, pp. 511–522, 2005.

[40] R. Jensen and R. O Crapo, “Diffusing Capacity: How to Get It Right,” *Respir. Care*, vol. 48, no. 8, 2003.

[41] Patient, “Respiratory System History and Examination | Doctor | Patient.co.uk,” pp. 1–5.

[42] P. Lip and B. Technique, “Diaphragmatic Breathing,” no. May, pp. 3–4, 2011.

[43] Bitzesty, “Our Agile Web Development Process - Bit Zesty - London based design and development agency.” [Online]. Available: http://bitzesty-2014.squarespace.com/blog/2014/7/our-agile-web-development-process. [Accessed: 03-Dec-2015].

[44] P. Uml and U. M. L. D. Ch, “Design and UML Class Diagrams.” p. 5.

[45] M. Felici, “Sequence Diagrams What are Sequence Diagrams ?,” 2011.

[46] Microsoft Corporation, “Service Level Agreement for Microsoft Online Services,” pp. 1–42, 2016.

[47] Microsoft Corporation, “Microsoft Azure Network Security,” pp. 1–16, 2014.

[48] Arduino, “Arduino - AnalogReadSerial,” 2015. [Online]. Available: https://www.arduino.cc/en/Tutorial/AnalogReadSerial. [Accessed: 18-Feb-2016].

[49] G. Hecht, “The Nyquist Sampling Theorem.”

[50] K. B. Ps and A. Jatti, “Respiration and Heart Rate Monitoring from Photoplethysmograph Signal,” vol. 3, no. 1, pp. 8–16, 2015.

[51] L. Last and P. M. Edt, “Photocells,” *Adafruit Learn. Syst.*, 2013.

[52] Ieee, *IEEE Standard Glossary of Software Engineering Terminology*, vol. 121990, no. 1. 1990.

[53] S. Nidhra, “Black Box and White Box Testing Techniques - A Literature Review,” *Int. J. Embed. Syst. Appl.*, vol. 2, no. 2, pp. 29–50, 2012.

Appendices

13.1 Login Function Unit Test

Login Function Unit Testing						
Test Case ID	Description	Input Data	Expected Result	Actual Result	Pass/Fail	Remarks
1	Enter valid username and password	username:"cf123", password:"pass"	You Have Successfully Signed In	You Have Successfully Signed In	Pass	.
2	Enter incorrect password	username:"cf123", password:"pass123"	The Password You Have Entered Is Incorrect. Please Try Again	The Password You Have Entered Is Incorrect. Please Try Again	Pass	.
3	Enter incorrect username	username:"cf111", password:"pass"	There Is No User Registered With This Username. Please Try Again Or Create An Account	There Is No User Registered With This Username. Please Try Again Or Create An Account	Pass	.
4	Enter blank password	username:"cf123", password:""	The Password You Have Entered Is Incorrect. Please Try Again	The Password You Have Entered Is Incorrect. Please Try Again	Pass	.
5	Enter blank username	username:"", password:"pass"	There Is No User Registered With This Username. Please Try Again Or Create An Account	There Is No User Registered With This Username. Please Try Again Or Create An Account	Pass	.
6	Enter correct password in all caps	username:"cf123", password:"PASS"	The Password You Have Entered Is Incorrect. Please Try Again	The Password You Have Entered Is Incorrect. Please Try Again	Pass	.
7	Enter correct username in all caps	username:"CF123", password:"pass"	There Is No User Registered With This Username. Please Try Again Or Create An Account	There Is No User Registered With This Username. Please Try Again Or Create An Account	Pass	.
8	Enter correct username with space at end	username:"cf123 ", password:"pass"	There Is No User Registered With This Username. Please Try Again Or Create An Account	There Is No User Registered With This Username. Please Try Again Or Create An Account	Pass	.
9	Enter correct password with space at end	username:"cf123", password:"pass "	The Password You Have Entered Is Incorrect. Please Try Again	The Password You Have Entered Is Incorrect. Please Try Again	Pass	.
10	Enter correct username with space at beginning	username:" cf123", password:"pass"	There Is No User Registered With This Username. Please Try Again Or Create An Account	There Is No User Registered With This Username. Please Try Again Or Create An Account	Pass	.
11	Enter correct password with space at beginning	username:"cf123", password:" pass"	The Password You Have Entered Is Incorrect. Please Try Again	The Password You Have Entered Is Incorrect. Please Try Again	Pass	.

13.2 Register Function Unit Test #1

Register Function Unit Testing						
Test Case ID	Description	Input Data	Expected Result	Actual Result	Pass/Fail	Remarks
1	Enter valid data	firstname:"colm",secondname:"fitzpatrick",username:"cff",password:"pass",verifypassword:"pass",weight:"70",height:"5.7",age:"22",smokestatus:"Yes",etco2:"30"	You Have Successfully Created An Account	You Have Successfully Created An Account	Pass	.
2	Enter symbol into first name field	firstname:"",secondname:"fitzpatrick",username:"cff",password:"pass",verifypassword:"pass",weight:"70",height:"5.7",age:"22",smokestatus:"Yes",etco2:"30"	Please Enter A Valid First Name. It May Only Contain Letters	Please Enter A Valid First Name. It May Only Contain Letters	Pass	.
3	Enter symbol into second name field	firstname:"colm",secondname:"",username:"cff",password:"pass",verifypassword:"pass",weight:"70",height:"5.7",age:"22",smokestatus:"Yes",etco2:"30"	Please Enter A Valid Second Name. It May Only Contain Letters	Please Enter A Valid Second Name. It May Only Contain Letters	Pass	.
4	Enter blank first name	firstname:"",secondname:"fitzpatrick",username:"cff",password:"pass",verifypassword:"pass",weight:"70",height:"5.7",age:"22",smokestatus:"Yes",etco2:"30"	Please Enter A Valid First Name. It May Only Contain Letters	Please Enter A Valid First Name. It May Only Contain Letters	Pass	.
5	Enter blank second name	firstname:"colm",secondname:"",username:"cff",password:"pass",verifypassword:"pass",weight:"70",height:"5.7",age:"22",smokestatus:"Yes",etco2:"30"	Please Enter A Valid Second Name. It May Only Contain Letters	Please Enter A Valid Second Name. It May Only Contain Letters	Pass	.
6	Enter number in first name field	firstname:"colm7",secondname:"fitzpatrick",username:"cff",password:"pass",verifypassword:"pass",weight:"70",height:"5.7",age:"22",smokestatus:"Yes",etco2:"30"	Please Enter A Valid First Name. It May Only Contain Letters	Please Enter A Valid First Name. It May Only Contain Letters	Pass	.
7	Enter number in second name field	firstname:"colm",secondname:"fitzpatrick7",username:"cff",password:"pass",verifypassword:"pass",weight:"70",height:"5.7",age:"22",smokestatus:"Yes",etco2:"30"	Please Enter A Valid Second Name. It May Only Contain Letters	Please Enter A Valid Second Name. It May Only Contain Letters	Pass	.
8	Enter only numbers to first name field	firstname:"7",secondname:"fitzpatrick",username:"cff",password:"pass",verifypassword:"pass",weight:"70",height:"5.7",age:"22",smokestatus:"Yes",etco2:"30"	Please Enter A Valid First Name. It May Only Contain Letters	Please Enter A Valid First Name. It May Only Contain Letters	Pass	.
9	Enter only numbers to second name field	firstname:"colm",secondname:"7",username:"cff",password:"pass",verifypassword:"pass",weight:"70",height:"5.7",age:"22",smokestatus:"Yes",etco2:"30"	Please Enter A Valid Second Name. It May Only Contain Letters	Please Enter A Valid Second Name. It May Only Contain Letters	Pass	.
10	Enter space before name in first name field	firstname:" colm",secondname:"fitzpatrick",username:"cff",password:"pass",verifypassword:"pass",weight:"70",height:"5.7",age:"22",smokestatus:"Yes",etco2:"30"	Please Enter A Valid First Name. It May Only Contain Letters	Please Enter A Valid First Name. It May Only Contain Letters	Pass	.

13.3 Register Function Unit Test #2

11	Enter space before name in second name field	firstname:"colm",secondname:" fitzpatrick",username:"cff",password:"pass",verifypassword:"pass",weight:"70",height:"5.7",age:"22",smokestatus:"Yes",etco2:"30"	Please Enter A Valid Second Name. It May Only Contain Letters	Please Enter A Valid Second Name. It May Only Contain Letters	Pass	.
12	Enter space after name in first name field	firstname:"colm ",secondname:"fitzpatrick",username:"cff",password:"pass",verifypassword:"pass",weight:"70",height:"5.7",age:"22",smokestatus:"Yes",etco2:"30"	Please Enter A Valid First Name. It May Only Contain Letters	Please Enter A Valid First Name. It May Only Contain Letters	Pass	.
13	Enter space after name in second name field	firstname:"colm",secondname:"fitzpatrick",username:"cff",password:"pass",verifypassword:"pass",weight:"70",height:"5.7",age:"22",smokestatus:"Yes",etco2:"30"	Please Enter A Valid Second Name. It May Only Contain Letters	Please Enter A Valid Second Name. It May Only Contain Letters	Pass	.
14	Enter username already in use	firstname:"colm",secondname:"fitzpatrick",username:"cf123",password:"pass",verifypassword:"pass",weight:"70",height:"5.7",age:"22",smokestatus:"Yes",etco2:"30"	The Username You Have Entered Is Already In Use, Please Try Another	The Username You Have Entered Is Already In Use, Please Try Another	Pass	.
15	Enter blank username	firstname:"colm",secondname:"fitzpatrick",username:"",password:"pass",verifypassword:"pass",weight:"70",height:"5.7",age:"22",smokestatus:"Yes",etco2:"30"	Please Enter A Value For The Username Field	You Have Successfully Created An Account	Fail	Check length instead of null
16	Enter blank password	firstname:"colm",secondname:"fitzpatrick",username:"cff",password:"",verifypassword:"pass",weight:"70",height:"5.7",age:"22",smokestatus:"Yes",etco2:"30"	Please Enter A Value For The Password Field	You Have Successfully Created An Account	Fail	Check length instead of null
17	Enter incorrect password in verify password field	firstname:"colm",secondname:"fitzpatrick",username:"cff",password:"pass",verifypassword:"pass123",weight:"70",height:"5.7",age:"22",smokestatus:"Yes",etco2:"30"	The Password And Verify Password Values You Have Entered Do Not Match. Please Try Again	The Password And Verify Password Values You Have Entered Do Not Match. Please Try Again	Pass	.
18	Enter letter in weight field	firstname:"colm",secondname:"fitzpatrick",username:"cff",password:"pass",verifypassword:"pass",weight:"7t",height:"5.7",age:"	The Weight Field May Only Contain Numbers	The Weight Field May Only Contain Numbers	Pass	.
19	Enter letter in height field	firstname:"colm",secondname:"fitzpatrick",username:"cff",password:"pass",verifypassword:"pass",weight:"70",height:"5.seven",	The Height Field May Only Contain Numbers	The Height Field May Only Contain Numbers	Pass	.
20	Enter blank weight	firstname:"colm",secondname:"fitzpatrick",username:"cff",password:"pass",verifypassword:"pass",weight:"",height:"5.7",age:"22",smokestatus:"Yes",etco2:"30"	Please Enter A Value For The Weight Field	You Have Successfully Created An Account	Fail	Check length instead of null
21	Enter blank height	firstname:"colm",secondname:"fitzpatrick",username:"cff",password:"pass",verifypassword:"pass",weight:"70",height:"",age:"22",smokestatus:"Yes",etco2:"30"	Please Enter A Value For The Height Field	You Have Successfully Created An Account	Fail	Check length instead of null

13.4 Register Function Unit Test #3

Register Function Unit Testing						
Test Case ID	Description	Input Data	Expected Result	Actual Result	Pass/Fail	Remarks
1	Enter a blank etco2 value	firstname:"colm",secondname:"fitzpatrick",username:"cff",password:"pass",verifypassword:"pass",weight:"70",height:"5.7",age:"22",smokestatus:"Yes",etco2:""	Please Enter A Positive ETCO2 Value	Please Enter A Positive ETCO2 Value	Pass	.
2	Enter negative value in etco2 field	firstname:"colm",secondname:"fitzpatrick",username:"cff",password:"pass",verifypassword:"pass",weight:"70",height:"5.7",age:"22",smokestatus:"Yes",etco2:"-30"	Please Enter A Positive ETCO2 Value	Please Enter A Positive ETCO2 Value	Pass	.
2	Enter negative value in etco2 field	firstname:"colm",secondname:"fitzpatrick",username:"cff",password:"pass",verifypassword:"pass",weight:"70",height:"5.7",age:"22",smokestatus:"Yes",etco2:"-30"	Please Enter A Positive ETCO2 Value	Please Enter A Positive ETCO2 Value	Pass	.

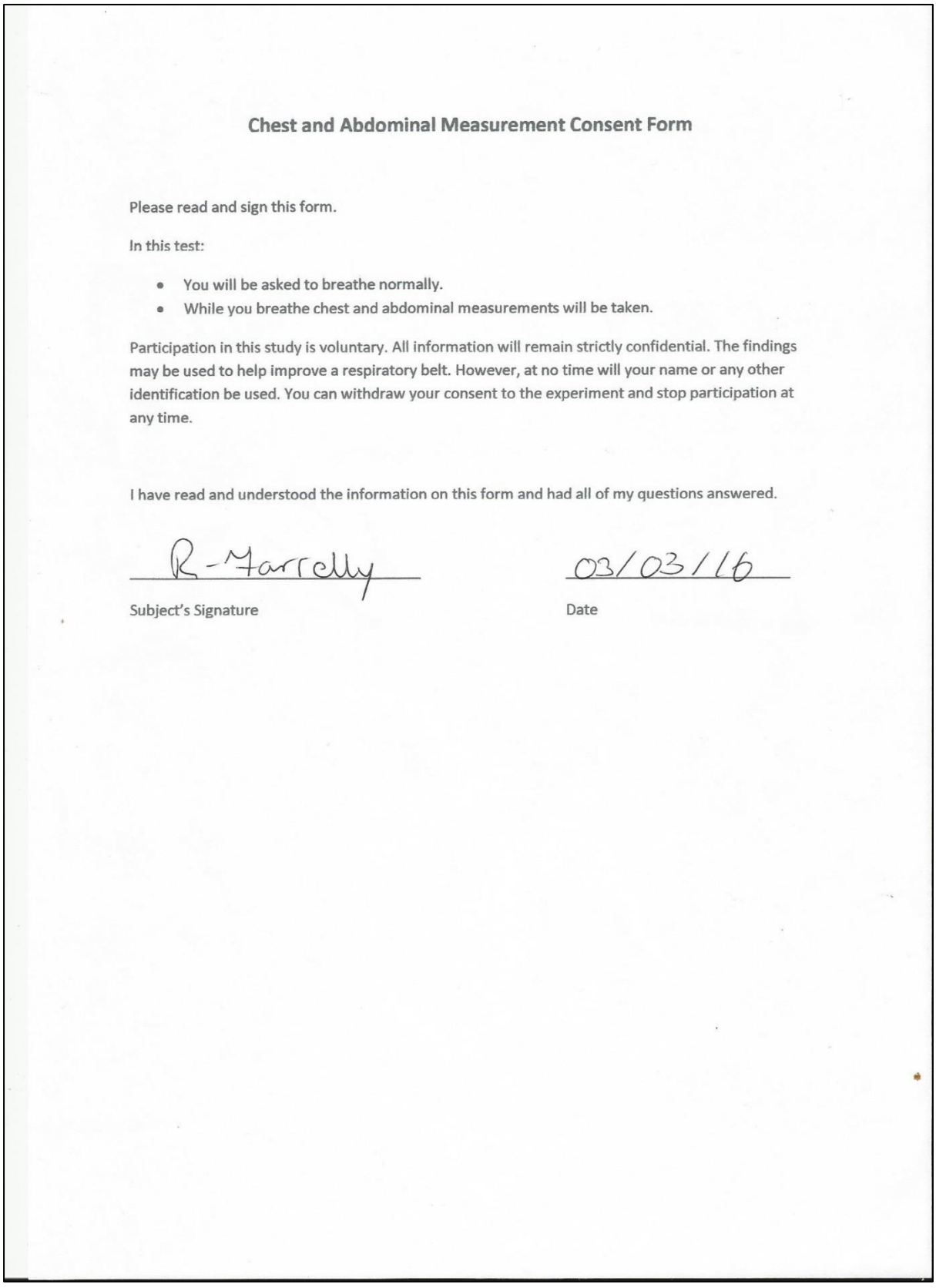
13.5Respiratorytest4 Class Unit Test #1

Respiratorytest4 Serial Event Unit Testing						
Test Case ID	Description	Input Data	Expected Result	Actual Result	Pass/Fail	
1	Enter valid data	"1,1"	data entered is satisfactory	data entered is satisfactory	Pass	.
2	Include letter with first value in string	"1t,1"	belt data is not numeric	belt data is not numeric	Pass	.
3	Include letter with second value in string	"1,1t"	belt data is not numeric	belt data is not numeric	Pass	.
4	First value in string is zero	"0,1"	data smaller than 1	data smaller than 1	Pass	.
5	Second value in string is zero	"1,0"	data smaller than 1	data smaller than 1	Pass	.
6	Blank first value in string	",1"	input string format error	Error empty string	Failed	Check length of each value
7	Blank second value in string	"1,"	input string format error	input string format error	Pass	.
8	Enter only a single comma	","	input string format error	input string format error	Pass	.
9	pass empty string	""	empty string passed to function	empty string passed to function	Pass	.
10	Enter fullstop instead of comma	"1.1"	input string format error	input string format error	Pass	.
11	Replace comma with space	"10 10"	input string format error	input string format error	Pass	.
12	Enter single value only	"10"	input string format error	input string format error	Pass	.

13.6 Respiratorytest4 Class Unit Test #2

Respiratorytest4 Serial Event Unit Testing						
Test Case ID	Description	Input Data	Expected Result	Actual Result	Pass/Fail	
1	Blank first value in string	",1"	input string format error	input string format error	Pass	.

13.7 Chest and Abdominal Respiratory Expansion Test Consent Forms



Chest and Abdominal Measurement Consent Form

Please read and sign this form.

In this test:

- You will be asked to breathe normally.
- While you breathe chest and abdominal measurements will be taken.

Participation in this study is voluntary. All information will remain strictly confidential. The findings may be used to help improve a respiratory belt. However, at no time will your name or any other identification be used. You can withdraw your consent to the experiment and stop participation at any time.

I have read and understood the information on this form and had all of my questions answered.

R-Farrelly — Subject's Signature

03/03/16 — Date

Chest and Abdominal Measurement Consent Form

Please read and sign this form.

In this test:

- You will be asked to breathe normally.
- While you breathe chest and abdominal measurements will be taken.

Participation in this study is voluntary. All information will remain strictly confidential. The findings may be used to help improve a respiratory belt. However, at no time will your name or any other identification be used. You can withdraw your consent to the experiment and stop participation at any time.

I have read and understood the information on this form and had all of my questions answered.

Alice Farrelly — Subject's Signature

3/3/16 — Date

Chest and Abdominal Measurement Consent Form

Please read and sign this form.

In this test:

- You will be asked to breathe normally.
- While you breathe chest and abdominal measurements will be taken.

Participation in this study is voluntary. All information will remain strictly confidential. The findings may be used to help improve a respiratory belt. However, at no time will your name or any other identification be used. You can withdraw your consent to the experiment and stop participation at any time.

I have read and understood the information on this form and had all of my questions answered.

Subject's Signature

Date 3/3/16

Chest and Abdominal Measurement Consent Form

Please read and sign this form.

In this test:

- You will be asked to breathe normally.
- While you breathe chest and abdominal measurements will be taken.

Participation in this study is voluntary. All information will remain strictly confidential. The findings may be used to help improve a respiratory belt. However, at no time will your name or any other identification be used. You can withdraw your consent to the experiment and stop participation at any time.

I have read and understood the information on this form and had all of my questions answered.

Paul Moran — Subject's Signature

03/03/16 — Date

Chest and Abdominal Measurement Consent Form

Please read and sign this form.

In this test:

- You will be asked to breathe normally.
- While you breathe chest and abdominal measurements will be taken.

Participation in this study is voluntary. All information will remain strictly confidential. The findings may be used to help improve a respiratory belt. However, at no time will your name or any other identification be used. You can withdraw your consent to the experiment and stop participation at any time.

I have read and understood the information on this form and had all of my questions answered.

William Fitzpatrick — 3/3/16

Subject's Signature — Date

Chest and Abdominal Measurement Consent Form

Please read and sign this form.

In this test:

- You will be asked to breathe normally.
- While you breathe chest and abdominal measurements will be taken.

Participation in this study is voluntary. All information will remain strictly confidential. The findings may be used to help improve a respiratory belt. However, at no time will your name or any other identification be used. You can withdraw your consent to the experiment and stop participation at any time.

I have read and understood the information on this form and had all of my questions answered.

Brian Briscoe _____ 03/03/16 _____

Subject's Signature Date

Chest and Abdominal Measurement Consent Form

Please read and sign this form.

In this test:

- You will be asked to breathe normally.
- While you breathe chest and abdominal measurements will be taken.

Participation in this study is voluntary. All information will remain strictly confidential. The findings may be used to help improve a respiratory belt. However, at no time will your name or any other identification be used. You can withdraw your consent to the experiment and stop participation at any time.

I have read and understood the information on this form and had all of my questions answered.

Ross Mccann _____ 3/3/16 _____

Subject's Signature Date

Chest and Abdominal Measurement Consent Form

Please read and sign this form.

In this test:

- You will be asked to breathe normally.
- While you breathe chest and abdominal measurements will be taken.

Participation in this study is voluntary. All information will remain strictly confidential. The findings may be used to help improve a respiratory belt. However, at no time will your name or any other identification be used. You can withdraw your consent to the experiment and stop participation at any time.

I have read and understood the information on this form and had all of my questions answered.

Subject's Signature

03/03/16

Date

Chest and Abdominal Measurement Consent Form

Please read and sign this form.

In this test:

- You will be asked to breathe normally.
- While you breathe chest and abdominal measurements will be taken.

Participation in this study is voluntary. All information will remain strictly confidential. The findings may be used to help improve a respiratory belt. However, at no time will your name or any other identification be used. You can withdraw your consent to the experiment and stop participation at any time.

I have read and understood the information on this form and had all of my questions answered.

Colin Lyons — Subject's Signature

03/03/16 — Date

Chest and Abdominal Measurement Consent Form

Please read and sign this form.

In this test:

- You will be asked to breathe normally.
- While you breathe chest and abdominal measurements will be taken.

Participation in this study is voluntary. All information will remain strictly confidential. The findings may be used to help improve a respiratory belt. However, at no time will your name or any other identification be used. You can withdraw your consent to the experiment and stop participation at any time.

I have read and understood the information on this form and had all of my questions answered.

Rory Hayes — Subject's Signature

03/03/16 — Date