Human Heart Simulation Software

Submitted in partial fulfillment of the requirements for the award of the degree of Bachelor of Science (Honours) in Computer Science

Author: Iolanda - Francesca Popescu
Supervisor: Prof. Robert Stevens

May 2, 2016
Abstract

The study of the human heart dates back to antiquity and now the knowledge of its workings as the engine of the human body have become well understood. This project explores the ability to transpose the knowledge of how the human heart functions to a user-friendly virtual environment where it can be studied and monitored.

The main goal of the Human Heart Simulation Software was to provide a functional heart simulation software which uses manual activity input, as well as a database that stores patients’ medical history in order to graphically output an electrocardiogram showing heart behaviour and activity over a period of time. While the main goal has been reached, integrating the Fitbit Charge HR fitness wristband in the project was an objective which has been only partially achieved.

Nevertheless, this software and its evaluation have shown that it is possible to obtain a representation of the heart activity, which can be improved in terms of accuracy and could possibly be used in preventive medicine and real-time monitoring.
Acknowledgements

I would like to take this opportunity to thank my supervisor, Prof. Robert Stevens, who offered me his guidance and advice from the moment I proposed the project idea until the very end. My thanks also go to Alan Davies for agreeing to offer his advice and expertise and help evaluate my project, as well as giving me the permission to use his name and views in the project report. I would like to thank my family who have been morally and financially supporting me throughout not only this crucial year of my education, but throughout the past three years while I have been pursuing this degree. I would also like to thank all my friends, especially my boyfriend, for encouraging me and helping me stay motivated.

Last, but not least, I would like to thank The University of Manchester School of Computer Science for accepting me and providing me with the education, resources and support I needed for an amazing start in the professional world.

I have been and still am surrounded by wonderful people, some of whom I even had the chance to work with and, even though this little section would not allow me to thank everyone one by one, I would like them to know how much they matter to me and that I owe a significant part of my achievements to them.
# Contents

1 Introduction 8
   1.1 Project Description .................................................. 8
   1.2 Motivation Behind Project Choice .................................... 10

2 Background 12
   2.1 Context of Study: The Human Heart ................................ 12
       2.1.1 The Anatomy of the Human Heart .............................. 12
       2.1.2 Electrocardiography ........................................... 15
   2.2 Related Work ............................................................ 17
   2.3 Chapter Summary ....................................................... 18

3 Design and Implementation 19
   3.1 Requirements and Objectives ......................................... 19
   3.2 Choice of Software and Technologies ............................... 20
       3.2.1 Development Tools ............................................. 20
       3.2.2 Web Development ............................................... 20
       3.2.3 Data Generation, Storage and Exchange ....................... 21
   3.3 Choice of Hardware: The Fitbit Charge HR ......................... 21
   3.4 The Architecture of the System ...................................... 23
       3.4.1 The Database .................................................. 24
       3.4.2 The Medical Dashboard ......................................... 25
       3.4.3 The Simulator .................................................. 28
   3.5 The Simulation Algorithm ............................................ 32
       3.5.1 Original Approach: Fourier Series ........................... 32
       3.5.2 Signal Generation .............................................. 32
   3.6 Chapter Summary ....................................................... 36

4 Evaluation 38
   4.1 Behaviour Analysis and Testing ...................................... 38
   4.2 Visual and Metric Juxtaposition ..................................... 39
   4.3 Chapter Summary ....................................................... 42

5 Reflection and Conclusion 43
   5.1 Challenges ............................................................... 43
   5.2 Dropped Features and Future Work .................................. 44
   5.3 Conclusion ............................................................... 44
Appendices 51

A The Electrical System of the Heart 52

B Fitbit Subscription API 47 55

C System Architecture Diagram 56
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Screen capture of the Human Heart Simulator mid-simulation</td>
<td>9</td>
</tr>
<tr>
<td>1.2</td>
<td>Fitbit Charge HR wristbands</td>
<td>10</td>
</tr>
<tr>
<td>1.3</td>
<td>Prevalence of obesity among adults aged 16+ years in the UK</td>
<td>11</td>
</tr>
<tr>
<td>2.1</td>
<td>Diagram of the internal human heart with connecting veins and arteries</td>
<td>13</td>
</tr>
<tr>
<td>2.2</td>
<td>Diagram of the electrical conduction system of the heart</td>
<td>15</td>
</tr>
<tr>
<td>2.3</td>
<td>ECG representation of the sinus rhythm with labels</td>
<td>16</td>
</tr>
<tr>
<td>2.4</td>
<td>Example of normal sinus rhythm on recorded ECG</td>
<td>16</td>
</tr>
<tr>
<td>2.5</td>
<td>Synthetic ECG signal developed in MATLAB at the Technical University of Ostrava</td>
<td>17</td>
</tr>
<tr>
<td>3.1</td>
<td>The range of available Fitbit trackers as of April 2016</td>
<td>22</td>
</tr>
<tr>
<td>3.2</td>
<td>Section of the simulation frame showing choice of Fitbit tracker connection</td>
<td>22</td>
</tr>
<tr>
<td>3.3</td>
<td>Fitbit authentication page the user is redirected to upon clicking the Fitbit connect button</td>
<td>23</td>
</tr>
<tr>
<td>3.4</td>
<td>The structure of the database</td>
<td>25</td>
</tr>
<tr>
<td>3.5</td>
<td>The navigation bar in the Medical Dashboard</td>
<td>26</td>
</tr>
<tr>
<td>3.6</td>
<td>Second step of the Human Heart Simulator’s New Patient form</td>
<td>26</td>
</tr>
<tr>
<td>3.7</td>
<td>Pseudo-CMS showing individual patient files</td>
<td>27</td>
</tr>
<tr>
<td>3.8</td>
<td>Screen capture of the Human Heart Simulator mid-simulation</td>
<td>28</td>
</tr>
<tr>
<td>3.9</td>
<td>20-tap Daubechies wavelet</td>
<td>33</td>
</tr>
<tr>
<td>3.10</td>
<td>Daubechies wavelets sampled at different rates</td>
<td>34</td>
</tr>
<tr>
<td>3.11</td>
<td>Ventricular fibrillation as generated by the simulator</td>
<td>36</td>
</tr>
<tr>
<td>4.1</td>
<td>Comparison between a real ECG and a synthetic one simulated by the Human Heart Simulator</td>
<td>40</td>
</tr>
<tr>
<td>4.2</td>
<td>Comparison between a real ECG and a synthetic one simulated by the Human Heart Simulator</td>
<td>41</td>
</tr>
</tbody>
</table>
List of Tables

4.1 Testing scheme for the Human Heart Simulator modules and corresponding functions ........................................... 39
List of source codes

1. Fitbit heart rate request ........................................... 24
2. Rendering and updating of the ECG graph ...................... 29
3. Application receiving coordinates from the Python server .... 30
4. Generation of synthetic ECG wavelets exhibiting normal sinus rhythm ........................................... 33
5. Use and application of the arrhythmia flag in the simulator .... 35
6. Ventricular fibrillation signal generation function .............. 36
Chapter 1

Introduction

An appropriate adage for the project would be “Prevention is better than cure”\(^1\) as this idea was born from the need of raising awareness about heart disease and helping prevent the consequences of misinformation on this issue.

1.1 Project Description

This project is concerned with designing and programming a human heart simulation program which could be used in preventive medicine and which also has prospective use in real-time heart monitoring if its development is continued.

The main objectives of this project are:

- Develop an algorithm which creates synthetic electrocardiograms using only heart rate as input
- Simulate normal heart function
- Simulate the effects of various activities on the heart
- Calculate a risk percentage for each patient based on their medical history
- Simulate a cardiac arrest
- Integrate the Fitbit Charge HR wristband in the application

The program focuses on both external factors that are likely to influence the normal heart function, such as physical activity, and internal ones which are related to the patient’s medical history in this case. Even though there are plenty of factors which may have a

\(^1\)English proverb, with the American alternative being “An ounce of prevention is worth a pound of cure”
negative impact on a person’s heart and overall health, the primary focus of the application is Coronary Heart Disease (CHD) and common abnormalities in the electrical activity of the heart, such as arrhythmia, heart attack and sudden cardiac arrest (SCA), as recorded on an electrocardiogram (ECG).

On starting the simulation, a risk percentage used for determining arrhythmia occurrence throughout the simulation is calculated for the selected patient. The simulation data consists of multiple sets of points output on an ECG-like graph. As the simulation begins with the heart in normal sinus rhythm, any obvious changes can easily be detected on the rolling real-time ECG sheet. There is also a set of external parameters which can be tweaked from within the user interface, that can influence the virtual heart activity (see Fig. 1.1).

As well as simulations, the software can show real-time data. Fitbit’s Charge HR fitness wristband (see Fig. 1.2) is used to record heart rate in real time. As the application subscribes to an intra-day data feed on Fitbit’s server, if the wristband is connected to either a mobile device or personal computer via Bluetooth, any change in the heart rate that the wristband detects will be sent over to the server, then passed along to the Human Heart Simulation Software.

The Human Heart Simulation Software also provides other types of functionality. It was designed to be an application which can be accessed by both medical staff within a medical institution or system and patients. Only featuring the medical side at the moment, the application allows medical staff to record, view and update patient data, as well as choose person from the list of patients in order to begin a simulation process.
1.2 Motivation Behind Project Choice

“Meta-analysis of prospective data on over one million adults has shown that for adults aged 40 to 69 years, each 20mmHg increase in usual systolic blood pressure, or 10mmHg increase in usual diastolic blood pressure, doubles the risk of death from CHD” [2]. While the UK national statistics office aims to raise awareness and give all the necessary information regarding cardiovascular disease prevention and treatment, not many people read their annual papers. With the obesity levels in the UK steadily increasing (see Fig. 1.3) and more and more people adopting an unhealthy diet and little exercise, cardiovascular disease, especially CHD is also becoming more common.

Moreover, “young people might think that any damage to their health caused by drinking lies so far in the future that it’s not worth worrying about” [3] and a study carried out on American adolescents reveals that more than half of the young population has unhealthy eating behaviours and also does not exercise as much as they should in order to lead a healthy lifestyle [4]. As over 50% of the young people have a profile that presents cardiovascular disease risks so early in their lives, it is likely that their behaviour will not change once they reach adulthood. High blood pressure is only one of the consequences of an unhealthy lifestyle and, when not diagnosed in time, it is also one of the main causes of heart disease and heart failure [5]. Providing information about heart function is one possible route to address an apparent problem of misinformation and misunderstanding.
Figure 1.3: Prevalence of obesity among adults aged 16+ years in the UK [6]
Chapter 2

Background

In the United Kingdom, after cancer and other unrelated causes, coronary heart disease (CHD) accounts for the majority of deaths in both men and women; it is closely followed by stroke and other cardiovascular diseases in this order [7].

2.1 Context of Study: The Human Heart

2.1.1 The Anatomy of the Human Heart

The Structure of the Heart

The human heart is a “hollow, muscular, cone-shaped organ” [8] which lies between the lungs, in the central area of the thoracic cavity, called the mediastinum. The heart (see Fig. 2.1) works as a blood pump in order to maintain a continuous blood flow through the veins and arteries. Its main function is to ensure the other organs and live tissue receive enough oxygen and nutrients at all times. It is divided into two halves, between which there is no communication in a healthy individual. Each half consists of two chambers – an atrium and a ventricle – and each of the two halves performs a different circulatory function, which is why the two subdivisions of the heart are also called the left heart and right heart. Oxygen-filled blood only flows through the left heart, while CO2-filled blood only flows through the right heart [9].
For the blood to be able to pass to and from the heart, as well as within the heart’s chambers, it has to flow through a set of valves which ensure that it only flows in the right direction by closing and opening several times during a cardiac cycle. Each atrium communicates with its corresponding ventricle through an atrioventricular (AV) valve. There is also a set of two semi-lunar (aortic and mitral) valves located at the base of the aorta and the pulmonary artery which separate the ventricles from the arteries and prevent the blood that has been pushed out of the heart from reentering it.

The Cardiac Cycle

The movement of blood within the heart is controlled by electrical stimulation through the innervations in the walls of the heart, coordinating the movement of the heart chambers. The electrical activity of the heart can be recorded on an electrocardiogram (ECG or EKG) and this is what this project attempts to simulate.

The cardiac cycle is defined as “the sequence of events between one heartbeat and the next, normally occupying less than a second” [11], as a complete heartbeat at a 75 beats-per-minute (bpm) rate usually lasts around 0.8 seconds [12]. The cardiac cycle is split into two phases: the contraction phase called systole and the relaxation phase called diastole.
A cardiac cycle begins with the atrial systole, during which the atria contract and blood is pumped into the ventricles until full. The semi-lunar valves and the atrioventricular (AV) valves close so that the blood does not flow back into the atria. The ventricles then enter ventricular systole, during which they also contract, the pressure generated by their contraction forcing the semi-lunar valves to open. Subsequently, the ventricles empty themselves into the arteries through these valves. The ventricular diastole involves the relaxation of the ventricles and the closing of the semi-lunar valves. The ventricular systole takes place at the same time as the atrial diastole, during which the atria fill with blood. When the pressure in the now blood-filled atria exceeds the pressure in the relaxed ventricles, the AV valves open as a new cardiac cycle begins. [13]

The heart sounds or vibrations we can either feel while taking the pulse, hear through a stethoscope or while undergoing an ECG procedure are generated by the closing of the heart valves. These sounds are often referred to as “lub-dub”. The “lub” sound is generated by the closure of the AV valves, while the “dub” can be heard when the semi-lunar valves close at the end of the cardiac cycle [14].

The blood pressure is “the force that the blood exerts on the walls of the blood vessels” [15]. In the large arteries leaving the heart, it varies with the heartbeat and it is higher than that in any other type of blood vessel in the body. The blood pressure in the arteries is highest upon ventricle contraction, thus when the ventricles are in systole, which is why this type of pressure is called systolic blood pressure. When it reaches its lowest level, the ventricles are in diastole, hence the name of diastolic blood pressure. We will later find out what the causes of variation in blood pressure and their implications are.

The Electrical Conduction System of the Heart

Now that we know what a cardiac cycle is and how the blood cells move through the heart, it is time to proceed to learn about what causes change of state in the components of the heart, such as opening and closure of the valves and contraction and relaxation of the cardiac muscle.

The heart’s electrical conduction system has three key components [16]:

- The sinoatrial (SA) node
- The atrioventricular (AV) node
- The His-Purkinje system

The electrical stimuli propagating around the heart co-ordinate the contraction and relaxation of the muscles of the four heart chambers. In this way, in a healthy heart, the electrical signal is generated when blood from the vena cava enters the right atria and it propagates through the left atria and further on through the ventricles. The electrical stimuli force the atria and ventricles to contract in turn – with a small delay, so that they do not pump blood towards each other. If the electrical stimuli do not follow the
correct path through the heart walls, irregularities can be noticed in the beating of one’s heart – these irregularities are commonly known as **arrhythmia**. Depending on how the electrical impulse (incorrectly) travels through the heart, the arrhythmia can be lighter or more serious. When an individual suffers from severe arrhythmia, they are exposed to multiple health risks, including a higher risk of suffering a heart attack. For more detailed information, see Appendix A.

### 2.1.2 Electrocardiography

“The standard 12-lead **electrocardiogram** is a representation of the heart’s electrical activity recorded from electrodes on the body surface” \[19\]. The placement guidelines for the 12 leads (electrodes) are universal throughout the world, so that when two electrocardiograms of the same person’s heart are compared, we know that any dissimilarities must have been caused by a cardiac injury \[20\].

The normal rhythm of the heart is medically known as **sinus rhythm**. Five different deflections labelled **PQRST** can be identified on the curve depicting a single heartbeat in sinus rhythm (see Fig. 2.3). However, waves Q and S are not always visible on an ECG, which is why this segment is often referred to as the **QRS Complex**.

The **P wave** represents atria depolarisation. It is followed by a brief flat line, also called the **PR segment**, which exhibits the signal delay enforced in the Bundle of His \[22\]. Then the **QRS Complex** as a whole depicts ventricular depolarisation, while the subsequent S
Two notable characteristics of an ECG curve are **amplitude** and **duration**. Both the highest and the lowest point of a heartbeat are reached during the QRS Complex, where the R peak denotes the highest amplitude and the S wave marks its lowest point, usually having a negative value. With regard to duration, we have already discussed that a single complete heartbeat lasts approximately 0.8 seconds at a 75 bpm rate. While the duration of the QRS Complex does not display much variation from a lower heart rate to a higher one, the duration of the other wave components and isoelectric lines[^1] representing the depolarisation signal dispersal between two heartbeats is inversely proportional to the heart rate.

Although clearly distinguishable in Figure 2.3, real ECGs rarely exhibit all the mathematical properties of this curve and come in various shapes and sizes, depending on the analysed subject (see Fig. 2.4).

[^1]: the flat parts of an ECG where no electrical activity is recorded in the heart; not to be confused with a flatline (also called *asystole*), which is a prolonged recorded time sequence during which the heart shows no electrical activity whatsoever
How to read an ECG

Electrocardiograms are recorded in black ink on red-grid paper. Amplitude is recorded on the Y (vertical) axis and time is measured along the X (horizontal) axis. As the rate of the paper is 25mV/s (millivolts per second) \[24\], the width of each large block represents 0.2 seconds. The height of two large blocks is the equivalent of 1 mV, thus the height of a small block is equal to 0.1 mV \[25\]. Therefore, in Figure 2.4 we notice that the RR distance corresponding to the first two beats is approximately 4 large blocks, that is 0.8 seconds.

The heart rate (75 bpm in this case) can thus be found using the following simple formula:

\[
HR = \frac{60}{RR}
\]

\[RR = \text{seconds between two successive R peaks}\]

Moreover, the height of the first R peak is roughly 1.8 large blocks or 0.9 mV, with the subsequent beats’ peaks varying by at most 0.1 mV.

Whenever these measurements significantly exceed or lie under the normal limits described above, it is a sign of possible abnormal heart function.

2.2 Related Work

![Figure 2.5: Synthetic ECG signal developed in MATLAB at the Technical University of Ostrava](image)

Although numerous papers having the human heart as object of study have been written, one particular paper is of major interest, as it analyses the generation of synthetic ECG waves to resemble the graphical representation of the human heart’s activity.
Research conducted at the Technical University of Ostrava shows how, by transposing the periodic functions and their Fourier series\(^2\) into MATLAB code, we can obtain synthetic ECG signals \([27]\) (see Fig. 2.5).

As a heartbeat repeats regularly, we can say it is periodic. Therefore, if we can express the different waves of a heartbeat as mathematical functions, we can represent them as Fourier series. As described in *Math of ECGs: Fourier series* \([29]\), we can find functions, of which the Fourier series’ graphical representations resemble the PQRST complex.

### 2.3 Chapter Summary

This chapter introduces the reader to the anatomical structure of the human heart, the roles of its components and how the innervations in its walls conduct electrical impulse in order to ensure its functioning. The science of electrocardiography, as well as other work in this field is discussed so that the main aim of the project – simulating electrical impulse – and the possible means of accomplishing it are clarified.

\(^2\)A Fourier series is an expansion of a periodic function \(f(x)\) in terms of an infinite sum of sines and cosines \([28]\).
Chapter 3

Design and Implementation

As discussed in the Project Description section under the first chapter, the system – originally designed to have 3 large components (see Appendix C) – now provides two functional units: the Medical Dashboard, which is the action-panel page a successful sign-in with a medical account redirects to and the Simulator, which is the core module of the application.

3.1 Requirements and Objectives

The requirements of the system were formulated in the form of user stories and because the development was more or less agile – considering the fact that the “team” contains one person only – the requirements changed throughout the development stage and several bits of functionality were dropped, while others were added. By compromising a set of features, we managed to achieve a fully working product. Below we have the user stories representing the final requirements of the project, all of which have been satisfied:

1. As a medic, I want to be able to record data about the patient’s heart and any other relevant medical information, so that I can simulate their heart.

2. As a medic, I want to have access to patients’ current heart rate during simulation, so that the simulation is more accurate.

3. As a medic, I want to be able to record my patients’ medical history and have access to it, so that I can check it and use it in simulations.

4. As a medic, I want a risk level to be established for each of my patients, so I can prioritise them.

5. As a medic, I want to be able to simulate the heart, so that the patient and I can predict its behaviour.

We will also discuss dropped requirements in section 5.2
3.2 Choice of Software and Technologies

The Human Heart Simulation Software comes in the form of a web application in order to be universally available and not have many dependencies on the platform it is accessed from. Using the application does not require any special tools or technical knowledge, which demonstrates its simplicity. It is quite important to note that this application was created exclusively using open source elements, such as libraries, icons, etc.

3.2.1 Development Tools

In order to be able to explain how several different components sustaining the application (e.g. local servers) work, we need to explain the choice of development tools and their capabilities first.

**XAMPP** is used to host the local **SQL database** and **Apache HTTP server**. As the application is not hosted online at the moment, the local servers and database served the purpose of simulating an online environment. However, since the requests made to the Fitbit API require the application to be accessed from outside the local domain, I decided to use **ngrok** to securely tunnel the traffic on my application through to a public URL which can be used to exchange requests with the Fitbit server. There were no issues identified when the application was exposed through a public tunnel, which demonstrates its fitness for deployment.

All development tasks were carried out using JetBrains’ **IntelliJ IDEA** IDE (Integrated Development Environment). Although initially designed for Java development and formerly known as IntelliJ only, the IDE has grown to support a wide range of frameworks and plugins, being well-known and used in the web development industry.

3.2.2 Web Development

The web application itself is designed using **HTML5**, **CSS3** and Twitter’s Bootstrap Framework for providing better user experience and time management. By reducing the time spent on design, more attention could be paid to implementing the desired functionality of the system and fixing the issues that would come up as a result. While HTML5 and CSS3 serve as foundation for the Human Heart Simulator, the larger part of it makes use of both open source and bespoke **JavaScript** code snippets.

While it was originally intended for the project to be written in Java, quite a bit of research helped steer it down the right path. Upon proposing this project idea, the fact that I had more experience with Java at the time seemed to weigh more than the amount of work required to implement certain features, as time was a transparent issue. As a result, little experience with JavaScript obviously hindered the development in the earliest of stages. However, since JavaScript is natively built for web development, some of the elements required in designing and building separate components of the system were already provided
in open source libraries. As the syntax is fairly similar to Java, it was fairly easy to pick up the pace and get accustomed to writing JavaScript. A major difference between Java and JavaScript is that the latter benefits from just-in-time compilation provided by most browsers nowadays – JavaScript essentially does not require a separate compiler, making writing and editing code quite straightforward.

### 3.2.3 Data Generation, Storage and Exchange

As the greatest part of this project deals with data generation, data output and information access, jQuery and AJAX were a natural fit for setting up the environment for effective data exchange.

The database consists of two separate tables – one which stores encrypted user login information and one which holds patient data, such as contact information and medical history. Patient data is retrieved from the database through AJAX requests. Several PHP scripts are used to carry out jobs such as database querying and template building, which will be discussed later in this chapter.

The simulation data is generated in a Python script, as even though this adds up to the stack of technologies used and JavaScript would have helped with the simplicity, it provides the Digital Signal Processing tools necessary for generating the data to be displayed on the ECG graph. Moreover, the language being high-level helps with faster development, considering the fact that resource management does not present a problem in this project. However, in order for the data to be transferred from the data generation script across to the application, WebSockets had to be used as a communication channel. A Python server deals with sending and receiving messages through the WebSocket. The data transported from the back-end script to the front-end (and the other way round) has the form of a “stringified” JSON object, as the WebSockets library used only allows strings and bytes to be sent. This JSON string usually contains information about changes in parameter values or flags for signalling change of state (e.g. the heart changes state from normal sinus rhythm to arrhythmia of a specified intensity).

### 3.3 Choice of Hardware: The Fitbit Charge HR

Fitbit is a wearable activity-trackers company headquartered in the United States. It produces a wide range of trackers which monitor activity like steps, distance, calories, heart rate, etc., depending on the device’s capabilities (see Fig. 3.1). The Fitbit Charge HR falls in their mid-range category.

Even though the application can be used with any Fitbit device with heart rate detection due to the API’s flexibility, I chose this particular wristband due to its good quality-to-price ratio. Other options would have been the Fitbit Surge and the new Fitbit Blaze. However, all of these integrate the innovative PurePulse™ wrist-based heart rate detection technology. Heart rate detection through the wrist only is possible by applying green LED
light beams onto the skin in order to detect the amount of blood flowing through the wrist veins at any given time.

Although the device records a large amount of data, for the purpose of this project only heart rate was used. As seen in Figure 3.2 connection to a particular person’s tracker is not mandatory upon starting a simulation process. An option to do so does, however, show in the top right corner of the simulation frame. Unfortunately, for being able to connect multiple trackers to one application, Fitbit requires a formal application before granting permission. As this application is a Third Year Project only at the moment, the full set of permissions could not be granted. Therefore, the only tracker I could use in the application is the one I own.

Authentication on the Fitbit server is carried out using OAuth 2.0. Therefore, in order for a user to log in with their Fitbit account in the application, a few steps have to be taken. First, the user is redirected to the authentication page provided by Fitbit (see Fig. 3.3), where they enter their credentials. A successful redirect to the application only happens when the authentication is successful and a response code is returned from the Fitbit server to the application. Using this code, a Client ID and a Client Secret unique to the client, a token is requested from Fitbit. Using this token, a subscription which binds the client application to the logged in user’s data feed is created. This subscription is only created the first time a user connects with their Fitbit account. Once the client is subscribed and new data is sent from the wristband to the server, the application will

Figure 3.1: The range of available Fitbit trackers as of April 2016

Figure 3.2: Section of the simulation frame showing choice of Fitbit tracker connection
be notified. Therefore the heart rate recordings on the wristband will be reflected in the simulator with a few seconds' delay, as long as the wristband is connected to either a mobile device or a computer via Bluetooth and the simulation is running. For more information about the use of the Fitbit Subscription API, check Appendix B.

The following block of code presents an AJAX request which retrieves heart rate intra-day time-series, provided that a subscription is already in place:

The function described in Listing 1 is executed periodically – every minute – and, using the access token retrieved upon log in and the time of the required recording, user-specific data is requested. All data stored on the Fitbit servers and data exchange through the Fitbit API is made using JSON objects or JSON-like strings. When received, the heart rate value is sent through the WebSocket to the simulator Python script and data is generated with the new value.

3.4 The Architecture of the System

In this section we will discuss what the different components of the Human Heart Simulation Software, the roles they have and how they are interconnected.
function getHeartRate() {
    var subtractedMin = subtractMin();
    var currTime = getTime();

    $.ajax({
        type: 'GET',
        url: 'https://api.fitbit.com/1/user/-/activities/heart/date/today/1d/1sec/time/'
            + subtractedMin + '/' + currTime + '.json',
        headers: {
            'Authorization': 'Bearer ' + accessToken
        },
        crossDomain: true,
        success: function(response) {
            if (ws && ws.readyState == 1)
                for (var e in response['activities-heart-intraday']['dataset'])
                    ws.send({'hr': e['value']});
        },
        error: function(error) {
            console.log(error);
        }
    });
}

Listing 1: Fitbit heart rate request

### 3.4.1 The Database

As stated in Section 3.2.3, the database consists of two separate tables – patient file and user (see Fig. 3.4). The user table solely supports access to the application by storing users’ credentials, including encrypted passwords. Although security is important as confidential patient data is stored and accessed, it is carried out upon sign-up using a random salt string currently stored on the machine the project was developed on. The same salt is used to verify the details supplied by the user against the information stored in the database upon log-in.

At the moment, the database does not hold any information regarding the medical staff who is to use this application apart from log-in information in the user table.

The user fields shown in Figure 3.4 can be divided into three main categories:

- **Personal details** – patient’s personal details, such as name, address, etc.
- **Basic Information** – information such as background, sex or measurements
- **Medical History** – information regarding factors that could add to the risk of heart
disease

While the fields in the first category are purely informative and are required for record keeping only, the information in the second and third categories is required for the simulator to be able to run properly.

3.4.2 The Medical Dashboard

**Medical Dashboard** is a general name for the panel of features a certified medic using the application has access to. As pictured in Figure 3.5, there are currently limited features of the application. The main elements the Medical Dashboard consists of are the pseudo-CMS (Content Management System) found under Patient Files and the Heart Simulator. The bottom two features (Stats and Notes) are not functional at the moment and will be discussed in section 5.2.

The Patients menu provides two sections: New Patient and Patient Files. We will further explain how each of the two was developed and what purpose they serve.

Registering a New Patient

All the details in the patient_file table are entered through the four-step form under the New Patient section (see Fig. 3.6) which is accessible to any user who owns a medical
account. Any patient a medic adds using this form is automatically assigned to them, showing up in their list of patients (the CMS) as soon as they access it. The form is designed using Bootstrap elements, such as input fields, as well as separate JavaScript elements reflected in the step organisation of the form. External open-source form validation code was also used, so that information required for starting a simulation procedure is not omitted. A neat layout was obtained by tweaking several CSS properties of these elements in order to scale and position them properly.

Figure 3.6: Second step of the Human Heart Simulator’s New Patient form
Behind the nicely laid out form, an AJAX POST request is made to a PHP script which takes all the information input in the form and inserts it into the database. As all pages available in the application are essentially PHP files, the active session and database connection are kept global so that they can be accessed when needed in side scripts.

**Patient Files**

In order for a medic to easily access and update their patients’ information, this pseudo-CMS which resides under *Patient Files* has been created. Figure 3.7 shows three registered patients. The folder icons are clickable and once the user clicks on one of them, they will be taken to a form containing the same fields as the New Patient form. However, this form is dynamically generated using the selected patient’s data from the database. A *template engine* written in PHP uses a template file which it fills with patient information from the database, then displays its contents on the screen. The template engine is a PHP class called *Template* which takes an HTML file upon instantiation and replaces several keywords in the file. This template engine looks for keywords of the form [@keyword] and replaces each of them when an instruction such as the following is executed:

```php
$profile->set('fname', $data->first_name);
```

The line of code above will replace all occurrences of [@fname] in the Template variable $profile with the value stored in $data->first_name, with $data being an associative array which stores all patient information retrieved from the database. Essentially, the frame is dynamically populated with patient data upon page load and the registered patient profiles are displayed.

Moreover, if when selecting a patient for the simulation it is discovered that mandatory information is missing, a warning message which requires updating the patient details pops up along with a link which points at the patient’s profile within the CMS. The patient profile and update form are essentially one and the same, so that it is easier for the user to make changes faster and without the hassle of having to take multiple steps in order to accomplish the required task.

![Figure 3.7: Pseudo-CMS showing individual patient files](image-url)
3.4.3 The Simulator

The simulator can be accessed from the navigation bar on the left hand side of the application (see Figure 3.5) and, as the name suggests, it is the core element of the Human Heart Simulation Software.

As Figure 3.8 shows, the simulation consists of outputting an ECG-like dynamic graph on what resembles the red grid paper used by an electrocardiograph to print the signals recorded by the twelve electrodes placed on the patient’s body. When a patient is selected and the simulation starts, there are a few things that happen before the ECG graph is rendered. We will shortly discuss each of these steps in turn in order to see how the simulator works and what its capabilities are at the moment.

Output

As stated in Chapter 2, the speed of a real electrocardiograph grid paper is 25mV/s. While it was intended to make the graph rendering as close to a real electrocardiograph as possible, the rendering and “grid paper” speed, as well as the correlation of these two with the stated heart rate during simulation may differ.

As soon as the simulation begins, normal sinus rhythm is reflected in the ECG graph when it starts rendering. As we saw in Figure 3.2 earlier, a loading wheel appears on the screen where the ECG graph is to be rendered before it does so. Once the minimum number of
Listing 2: Rendering and updating of the ECG graph

points – currently set to 200 – is received from the Python server through the WebSocket, the graph starts rendering. The dynamic live chart structure in the CanvasJS library – with a few alterations – is used for the graph.

Listing 2 shows how `dataX` and `dataY`, two separate arrays, hold the coordinates for 200 points on the graph at any time. `dps`, the array used by the CanvasJS chart, consists of coordinates pairs represented by a two-entry JSON object, which is why corresponding X-to-Y coordinates are collected in a JSON object and pushed into `dps`. In order for `dps` not to overflow, it is periodically spliced when the minimum of 200 elements is reached. The code snippet in Listing 3 shows how the data received through the WebSocket ends up in `dataX` and `dataY`.

We can see there are two read-only risk fields on the right hand side of the simulator frame. This cardiovascular disease risk is calculated for each patient when a simulation is started and it takes into account the following information:

- background – sex, race and age
While the base life-time risk can be calculated for any patient regardless of their age, the ten-year risk can only be calculated for patients aged 49 or older, according to the algorithm suggested in 2013 ACC/AHA Guideline on the Assessment of Cardiovascular Risk: A Report of the American College of Cardiology/American Heart Association Task Force on Practice Guidelines [34]. This algorithm uses natural logarithms which have been found to mathematically describe the relationship between the different factors mentioned above and the risk of cardiovascular disease. It was implemented in JavaScript and it currently only affects arrhythmia occurrence – the higher the risk, the higher the probability of random arrhythmia to occur at a higher level. Although it does not affect the behaviour of the simulator in any other way at the moment, it has the potential to provide more accurate ECG simulations which would significantly differ from one patient to another.

```
if (data.hasOwnProperty('x'))
    dataX.push(parseFloat(data.x));
if (data.hasOwnProperty('y'))
    dataY.push(parseFloat(data.y));
```

Listing 3: Application receiving coordinates from the Python server

### Altering Parameters

Once the simulation is running, there is a set of quite straightforward parameters which can be altered, as shown in Figure 3.8.

The **Heart rate** can be manually changed from the labeled slider on the left hand side, right below the ECG grid. It does not affect the other parameters in any way. However, as the maximum heart rate is calculated even before rendering the ECG graph, the user will not be allowed to set a value which is greater than the maximum heart rate on this slider. The initial value of the heart rate is retrieved from the `resting_hr` field corresponding to the patient entry in the database.

Apart from the naturally occurring risk-based arrhythmia discussed earlier, **arrhythmia** can also be manually triggered from the slider below the heart rate slider. Its value is initially set to 0 (none), as no sort of arrhythmia is assumed upon starting a simulation. The type of arrhythmia this slider generates is **atrial fibrillation**. This slider has four possible values:
• **0 (none)** – the patient does not experience arrhythmia
• **1 (low)** – minor arrhythmia occurs with no obvious signs on the ECG graph
• **2 (moderate)** – a fair amount of arrhythmia is experienced and there are obvious differences between the aspect of an ECG depicting normal sinus rhythm and a level 2 arrhythmia ECG graph
• **3 (severe)** – the arrhythmia is so severe that there are clear discrepancies in the spacing between heartbeats and the waves in the PQRST complex become deformed

In the centre of the frame below the ECG grid there is a set of actions which can be undertaken by the person whose heart is being simulated. The speed slider only becomes enabled when either *Walk* or *Run* is selected. The heart rate increase triggered by a change in activity or speed is based on the heart rate training zones [35] and, even though the variation in heart rate depends on the fitness level of each patient, this mapping is currently a generalised one in order to suit the average patient. Evidently, it can be further improved to adapt to each patient profile.

The sleep function comes in the form of a check box, such that the patient can both “fall asleep” and “wake up”. In real life, the heart rate slows by approximately 8 percent during sleep [36], this behaviour being reflected in the simulation as well. Evidently, all other functions are disabled while the patient is asleep and become enabled as soon as they wake up. Apart from manually waking up the patient, if severe (level 3) arrhythmia occurs during sleep, the patient will change state from *asleep to awake*.

### The Cardiac Arrest

Probably the most complex feature of this system, the **cardiac arrest** can be triggered at the push of a button at the bottom of the simulator frame. “If someone has suddenly collapsed, is not breathing normally and is unresponsive, they are in cardiac arrest” [37]. Usually, CPR (Cardiopulmonary Resuscitation) is given or an AED (Automated External Defibrillator) is used to make the person’s heart function normally again. However, this option is not incorporated in the simulator and, once a cardiac arrest is triggered it cannot be stopped and the patient eventually dies. When the heart attack flag is sent to the simulation script, all actions are disabled and a series of steps are taken in the following order:

1. Elevation of the arrhythmia level up to the highest one
2. Sudden heart rate increase up to maximum heart rate
3. Ventricular fibrillation
4. Asystole (flatline)
5. Death
At the end of a cardiac arrest, the time of death is provided in a modal in the centre of the simulation frame while the simulation-related actions remain disabled.

It would be appropriate to say that this is a naive heart simulator. Alan Davies, former cardiology nurse and current PhD student at The University of Manchester, states that research on digital automated interpretation of ECGs proves a diagnosis correctness rate of around 50 percent. Even human diagnosis based on ECG recordings has proven 75 percent correct. Therefore, the accuracy of this simulator is probably as good as it can be, considering the fact that I do not have any medical background.

3.5 The Simulation Algorithm

We will talk about how the original approach to building synthetic ECG waves using Fourier series would have been a better fit for the project, the reasons it was exchanged for the current solution and what the latter consists of.

3.5.1 Original Approach: Fourier Series

As discussed in Section 2.2, we could have used the Fourier series solution – developed in MATLAB – for the signal generation issue. Although we can say that MATLAB and Python can be interchanged easily and are widely used in academia and scientific research, Python has a significant advantage over MATLAB by simply being an open-source language. However, the lack of experience with MATLAB, as well as the late stage the solution to this requirement was identified in made taking this approach rather extreme.

3.5.2 Signal Generation

The ECG data is generated in a back-end Python script using two types of waves provided in the signal module of the scipy library [38]. We will discuss what each of them is used for and in what way.

Daubechies and Ricker Wavelets

The Daubechies Wavelets are based on Ingrid Daubechies’ [39] work and represent mathematical transforms which produce signals of the form depicted in Figure 3.9. While there is a great selection of waveforms available both in the scipy library and outside of it, the choice to use Daubechies waveletes is supported by their appearance which is the closest to recorded ECG signals. Although a resemblance between the waveform in Figure 3.9 and a real ECG wave or even the waves shown in the screen captures of the Human Heart Simulator may not be visible at first sight, we will explain what the factors which incur major changes in the behaviour of the signal are.
In Listing 4 we can see an adapted code snippet which shows the fundamental operations required to generate Daubechies wavelets to represent heartbeats.

```python
import scipy.signal as signal
import numpy

def generate_ecg(**kwargs):
    pqrst = signal.wavelets.daub(8)
    rest = 10
    zeros = numpy.zeros(rest, dtype=float)
    pqrst_full = numpy.concatenate([pqrst, zeros])
    capture_length = 3
    noise = 0.008
    ecg += numpy.tile(pqrst_full, capture_length)
    ecg += numpy.random.normal(0, noise, len(ecg))
    sampling_rate = 40.0
    samples_number = sampling_rate * capture_length
    ecg = signal.resample(ecg, samples_number)
    adc_bit_res = 1024
    ecg *= adc_bit_res
    return list(range(int(samples_number))), ecg
```

Listing 4: Generation of synthetic ECG wavelets exhibiting normal sinus rhythm

After generating a simple Daubechies signal, the length of the resting section between two consecutive heartbeats is set to 10 points in this case and a corresponding array of zeros of the same length is created. The previously generated signal is concatenated with the zeros
array to create a wavelet containing the full PQRST complex with its subsequent flat line. **capture_length** represents the number of heartbeats to be generated in this case and the specified amount of **noise** will be added to the final set of data points in order to create variance between individual heartbeats. Therefore, a sequence of the same repeating signal generated with the use of the `numpy.tile` function will be stored in the **ecg** array, to which the specified amount of noise is added.

The **sampling_rate** defines the number of points a single wavelet (along with its resting segment) should incorporate and is measured in Hertz (Hz). The sampling rate, similar to the noise, is a delicate matter, as any slight change in its value can possibly generate immense discrepancies in the shape of the signal. While for two-digit and three-digit values it only shows minor differences when plotted, for values less than or equal to 10 the signal becomes deformed causing its resulting aspect to distance itself from the original one. Figure 3.10 shows a head-to-head comparison between four signals. The top left signal is the one used by the simulator, sampled at 40Hz, while the top right one is sampled at 20Hz and we can already see change in the curvature of the waves, especially in the sharper peaks. The bottom two signals are sampled at 10Hz (left) and 5Hz (right). We can clearly see the signal changing shape and losing some of the attributes and individual wave definition the original signal had.
In order to generate three separate beats in this case, we need 120 points on the graph, that is the sampling rate multiplied by the capture length – this expression gives the samples number. After the data set is resampled through a Fourier transform along the given axis, we obtain an almost final version of the desired ECG recording. The last step is to set the ADC[1] bit resolution of the signal to 1024. At this point, the desired signal has been created and all Y coordinates of the points to be plotted reside in the ecg array, while the corresponding X coordinates are consecutive points with values from 0 to samples_number – 1. When these two arrays are sent to the front-end and the points described by them plotted on the dynamic chart, the desired ECG signal is represented.

As opposed to the basic version of an ECG signal described above, the one used by the simulator also takes flags received through the WebSocket in consideration, such as arrhythmia and heart rate. With incoming arrhythmia flags, the noise and rest variables can hold various values, depending on the values these flags have. Listing[5] shows how the simulator adapts to incoming changes.

```
1 bps = hr / 60
2
3 if arrhythmia > 0:
4     beats = 1
5     capture_length = 60 / hr
6 else:
7     capture_length = 3
8     beats = kwargs.get('beats', int(capture_length * bps))
9
10 if 'beats' in kwargs:
11     capture_length = beats / bps
```

Listing 5: Use and application of the arrhythmia flag in the simulator

The stages of a cardiac arrest are triggered by the cardiac arrest flag and the simulator enters the heart rate increase and maximum arrhythmia induction stages, which mark the final usage of the Daubechies wavelets.

When the third stage – ventricular fibrillation – is entered, the simulator switches to the second type of wavelets, called Ricker wavelets. These wavelets are used in a similar manner to the Daubechies wavelets (see Listing[6]) and their purpose is to represent a synthetic signal specific to ventricular fibrillation (see Figure 3.11). By looking closely at the v_fib function, we can see that it mainly executes the same operations as the generate_ecg function, apart from the use of the Daubechies waves and the former being more compact.

---

1 "The resolution of a n-bit analog-to-digital Converter (ADC) is a function of how many parts the maximum signal can be divided into. The formula to calculate resolution is $2^n$." [11]
def v_fib(self):
    fs = 6
    f = 3
    x = numpy.arange(fs * f / 2 * 5)
    y = signal.ricker(fs * f / 2, 8)
    y = numpy.tile(y, 5)
    y += numpy.random.normal(0, 0.01, len(y))
    y = signal.resample(y, fs * f / 2 * 5) * 1024
    return list(x), [i - 150 for i in y], [], None

Listing 6: Ventricular fibrillation signal generation function

While it normally takes longer for a real-life person to go from ventricular fibrillation to no heart activity whatsoever (asystole), the simulator is currently set to do so after only 30 seconds for ease of demonstration. As opposed to the previous heart electrical activity representations using different waveforms, the asystole is simulated by generating a number of points with coordinates \((x, 0)\) for yet another 30 seconds, until the time of death is shown on the screen and the simulation stops.

Figure 3.11: Ventricular fibrillation as generated by the simulator

3.6 Chapter Summary

This chapter discusses the technical details of the project. The Human Heart Simulation Software is a web application, the front-end of which is written in HTML5, CSS3 and
JavaScript. While HTML5 and CSS3 only fulfil visual functions, the simulation algorithms are written in JavaScript and Python. Signal generation using Daubechies and Ricker waves and data post-processing are done in Python and transmitted to the front-end through a WebSocket, where the data is displayed on an ECG-like grid paper. The front-end bears the name of Medical Dashboard and supports the simulator in an user-friendly manner.
Chapter 4

Evaluation

In this chapter we analyse the correctness of the algorithm and the overall simulation from the perspective of both the developer and a specialist in this domain.

4.1 Behaviour Analysis and Testing

The two main testing methodologies used are functional and regression testing. Table 4.1 shows the modules and related functions tested as part of the testing process. It is important to note that all tests were carried out manually.

Functional Testing

Functional testing “is primarily used to verify that a piece of software is providing the same output as required by the end-user or business” [12]. All system functions were tested and expected behaviour was compared against the results. Approximately 80 percent of the tests were carried out close to the end of the development stage in order to ensure all subsystems function correctly and a working product can be delivered by the deadline. The remaining 20 percent was spread over the development stage and was usually performed at the end of different milestones (i.e. the end of a component’s implementation).

A few of the major bugs encountered during development were poor ECG graph rendering due to misuse of X coordinates or distortion of the ECG signal upon heart rate increase. Fortunately, these and many others were discovered through functional testing and fixed. The application now passes all functional tests, thus satisfies the project requirements.
<table>
<thead>
<tr>
<th>Module</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Home Page</td>
<td>Visual elements display</td>
</tr>
<tr>
<td></td>
<td>Database connection</td>
</tr>
<tr>
<td></td>
<td>Sign-up and Sign-in features</td>
</tr>
<tr>
<td>Medical Dashboard</td>
<td>Visual elements display</td>
</tr>
<tr>
<td></td>
<td>Database connection</td>
</tr>
<tr>
<td></td>
<td>New Patient form functionality</td>
</tr>
<tr>
<td></td>
<td>Content Management System contents</td>
</tr>
<tr>
<td></td>
<td>Template engine (patient profile pages)</td>
</tr>
<tr>
<td>Simulator</td>
<td>Visual elements display</td>
</tr>
<tr>
<td></td>
<td>Database connection</td>
</tr>
<tr>
<td></td>
<td>ECG graph rendering</td>
</tr>
<tr>
<td></td>
<td>Output of the simulation function</td>
</tr>
<tr>
<td></td>
<td>Cardiovascular disease risk correctness</td>
</tr>
<tr>
<td></td>
<td>Parameter-altering elements functionality</td>
</tr>
<tr>
<td></td>
<td>Python server functionality</td>
</tr>
<tr>
<td></td>
<td>WebSocket data exchange</td>
</tr>
<tr>
<td>Fitbit</td>
<td>User connection</td>
</tr>
<tr>
<td></td>
<td>Data retrieval</td>
</tr>
<tr>
<td></td>
<td>Heart rate setting</td>
</tr>
</tbody>
</table>

Table 4.1: Testing scheme for the Human Heart Simulator modules and corresponding functions

**Regression Testing**

Regression testing was used throughout the entire development stage, especially after integrating different components. The simulation algorithm was developed gradually, starting with a basic version, similar to the one presented in Listing 4. As this function only generates ECG signals representing normal sinus rhythm, regression testing helped when its functionality was expanded through the addition of arrhythmia, activities and cardiac arrest. Extensive testing had to be carried out upon code expansion and refactoring in order to ensure any bugs which arose were fixed.

**4.2 Visual and Metric Juxtaposition**

Through both functional and regression testing, the simulation results were visually measured against real ECG recordings. There are a few elements to consider when measuring synthetic ECGs against real ones, such as amplitude, PQRST complex individual waves’ shapes, synchronisation with given heart rate and behaviour and aspect in different situations.
Figure 4.1 aims to compare a real ECG signal to a synthetic one. On the synthetic ECG, each square is equivalent to the smallest squares on the real ECG grid paper, therefore we can conclude that the amplitude is correct. While there is no such thing as two identical ECGs of the same person – let alone two different people – we can safely assume the synthetic amplitude is within the variability range of a real ECG.

As pointed out by Alan Davies and visible in this example as well, the T wave (the last wave of the PQRST complex), as well as the S wave (the sharpest negative wave), is more prominent than it should be. This is due to the nature of the Daubechies wavelet but both cases can potentially be altered by creating a function which takes the coordinates generated by the simulator and post-processes them. There currently are several peak detection algorithms which could be used to determine the negative peak describing the S wave. Furthermore, one such algorithm was designed during development for R-peak detection but was never used in the end. It can easily be adapted for negative peak detection, which would help find the S wave negative peaks and average out the wave in order to bring its amplitude closer to zero, therefore obtain an evened-out, rounder wave.

Although when people think of a cardiac arrest and no electrical activity in the heart,
they think of an actual flat line and it is often pictured so on television. However, the actual asystole exhibits a slightly undulating signal, despite its nickname – “flatline”. This is therefore not as much a functionality issue, but rather a requirements flaw and it can be fixed by replacing the \((x, 0)\) points generation with a sine wave.

**Arrhythmia** – atrial fibrillation in this case – is not very obvious on the graph, unless it exhibits its highest level of severity. Real arrhythmia can easily be spotted as the spacing between individual heartbeats varies significantly from a pair of heartbeats to another. This issue can be fixed by dynamically scaling the `rest` variable which holds the spacing between heartbeats, depending on the arrhythmia level.

![Figure 4.2: Comparison between a real ECG in ventricular fibrillation and a synthetic ECG in ventricular fibrillation simulated by the Human Heart Simulator](image)

Last but not least, Alan Davies claims that the accuracy of the synthetic ventricular fibrillation is quite high, being one of the most accurate features of the simulator. In Figure 4.2 below we can see the similarities between a real recording of ventricular fibrillation and a simulated signal.
4.3 Chapter Summary

Rigorous prolonged testing and evaluations have shown that the accuracy of the simulator is relatively high. While 100 percent accuracy is impossible to achieve with ECGs varying like fingerprints, Alan Davies claims it is fairly high, considering the developer’s lack of proper medical background. While some attributes, such as the S and T wave, are less accurate, ventricular fibrillation is very similar to its real graphical representation. More work could be done to improve the software and it could possibly be used in preventive medicine and, according to Alan Davies, in digital automated ECG interpretation.
Chapter 5

Reflection and Conclusion

By the end of the time spent working on the project, I had been gaining experience with web development tools, as well as Python and Digital Signal Processing overall. While previous experience certainly helped with the development, technologies completely new to the developer had to be learnt and used. One important thing I have learnt, apart from the technology and medicine knowledge, was planning and issue tracking. Using Trello, an online project management application, along with SourceTree, a version control software, I was able to record all requirements, changes and bugs. This made the development infinitely smoother and more organised than it would have been otherwise.

5.1 Challenges

Not only developing this project, but also choosing it was a challenge. The lack of sufficient medical knowledge implied a need for extensive research and study which represent nearly half of the time spent on project-related work.

Regarding development only, the main challenge was finding a solution to the graphical representation of heart activity. While upon choosing the project 3D graphics were part of the requirements, lack of experience and resources, as well as the slow development pace resulted in the decision to abandon this feature after a few weeks only. Apart from 3D objects rendering, the objective could have been achieved through the use of Fourier series. Although this solution would have yielded more accurate results, the time left, as well as lack of experience, would not allow for its implementation, considering the fact that calculus study only was spread over two entire weeks. Therefore, finding a solution to suit both the requirements and the available resources was the biggest challenge of this project.

Evidently, there were times when bugs slowed down the development and quite some time had to be devoted to fixing them. In spite of numerous frustrations and tough decisions, all bugs were eventually fixed. Each and every one of them represented a challenge, as most of them generally generated a chain reaction.
Finally, working with the Fitbit Charge HR tracker was partly difficult due to never having worked with similar hardware. However, Fitbit has quite a big development community, as well as API documentation, which helped with every single issue I came across.

I managed to deliver a product which does not only serve as a third year project, but also has prospective uses in preventive medicine, digital ECG interpretation and real-time monitoring if further improved. Therefore, by combining two different fields of study – medicine/biology and computer science – I am proud of having delivered a useful application, which was the first thought I had even before choosing a project.

5.2 Dropped Features and Future Work

As mentioned in the previous section, the first feature which was dropped was the 3D representation of the human heart. However, there was no real purpose for this 3D object other than attractive visuals. This is the reason behind not including it in the future work planned for this project.

Naturally, as the Human Heart Simulation Software was only focused on a few of the many existing heart conditions, further development would include extending it to cover as many cases and diseases as possible, including side effects of diseases mainly affecting other organs.

One last-minute idea was a reporting tool which would collect simulation information and create individual reports. Based on this information, statistics regarding each patient simulations have been run on could be provided to the medic. Examples of such statistics could be prevalence of arrhythmia and other abnormalities, duration of the simulation and individual wave analysis.

Among the dropped features we could also mention a more elaborate Medical Dashboard. In the initial plans an appointment and notifications system was also included. However, it was dropped as part of a prioritisation scheme which aimed to select the features which significantly contributed to the functionality of the simulator.

5.3 Conclusion

Overall, the Human Heart Simulator has achieved its main objectives of mimicking the electrical activity in the heart and simulating both normal and a limited range of abnormal behaviour, such as cardiac arrest and arrhythmia, were accomplished. While other specialist-conducted work was done in this field before the development of this project began, a comparison between this application and the formerly mentioned work is not appropriate due to the knowledge gap. The biggest achievement of this project, however, is the cardiac arrest simulation with all the individual stages it consists of, from a sharp increase in heart rate through to severe arrhythmia, ventricular fibrillation and death. The
key issues of the project were signal generation and data manipulation which, once solved, lead to the quality simulation data. Both the Daubechies and Ricker waves used for signal generation had to be resampled and post-processed in order to cover all key properties of a real ECG. While some of them are more accurate than others, there still is room for improvement and, as Alan Davies suggested, further work on this issue could possibly produce a good PhD research subject.

The system components – the Simulator and Medical Dashboard – interconnected by the cutting-edge technology used provide a specialised, user-friendly application which is intended to be used by medical staff.

Therefore, we can admit that an efficient algorithm which produces synthetic ECG data from a given heart rate value only has been successfully developed for this project and all objectives mentioned in Section 1.1 have been met, apart from the Fitbit integration. The latter was partially met, as it only supports heart rate retrieval at the moment.
Bibliography


Appendices
Appendix A

The Electrical System of the Heart

The topic of one of the very first biology lessons taught in school was the cell. Thus, we should be aware of the term polarity with regard to cells. Cell polarity, as described in Encyclopedic Reference of Genomics and Proteomics in Molecular Medicine, “is a fundamental property of eukaryotic cells and refers to the polarized organization of the cell membrane with associated proteins, as well as to the polarized arrangement of the cytoskeleton and organelles within the cytoplasm and to the vectorial transport of secretory vesicles” [17]. We know that human body cells normally depolarise – while being negatively charged, they become more positively charged – when a neighbouring cell does so. However, there is an area in the heart’s right atrium called the SA node, which contains unique cells that have the ability of depolarising by themselves – that is, without the aid of a depolarising cell nearby.

All this being said, a depolarisation signal is generated in the SA node when blood coming from the vena cava enters the right atrium and this signal is then propagated through gap junctions towards the other cells in the right atrium outside the SA node, as well as towards the left atrium and the AV node. The signal travels significantly slower towards the right atrium cells than it does towards the left atrium and the AV node. As depicted in Figure 2.2, this is due to the existence of Bachmann’s Bundle, a band of tissue which facilitates rapid signal propagation. Therefore, atria depolarisation occurs in a coordinated manner.

In the same fashion, the SA node is connected to the AV node by inter-nodal tracts, which also allow the signal to reach the AV node almost instantly. Once the signal has reached the AV node, its propagation is delayed by approximately 0.1 seconds, so that the atria and ventricles do not contract at the same time, pumping blood towards each other. Next, the AV node allows the depolarisation signal to propagate through the Bundle of His. The Bundle of His splits into the left and right bundle, while the left bundle splits in turn into the left posterior fascicle and the left anterior fascicle surrounding the left ventricle. Therefore, the absence of this ramification in the right heart explains why the left ventricle contracts a moment before the right ventricle. Both the left anterior fascicle and the right bundle fork into multiple small Purkinje fibres within the ventricle walls which depolarise the two ventricles. [18]
As the signal passes, the ventricles depolarise and therefore relax. The process described above occurs every cardiac cycle, that is every time the vena cava releases blood into the right atria.
Appendix B

Fitbit Subscription API

Using Fitbit Subscription API

1. Configure your server to respond to a subscription notification with HTTP 204
2. Make sure your subscriber endpoint is accessible from fitbit.com servers
3. Provide Fitbit with one or more subscription endpoints. A subscription endpoint is a callback URL where notifications are sent to
4. Authenticate Fitbit users on your website by using OAuth 2.0 authentication API
5. Add a subscription for the user to get notifications and return a response in the format requested
Appendix C

System Architecture Diagram