

5-2012

A modular framework for home healthcare monitoring

Blake Puryear

University of Arkansas, Fayetteville

Follow this and additional works at: <http://scholarworks.uark.edu/csceuht>

 Part of the [Equipment and Supplies Commons](#), and the [Graphics and Human Computer Interfaces Commons](#)

Recommended Citation

Puryear, Blake, "A modular framework for home healthcare monitoring" (2012). *Computer Science and Computer Engineering Undergraduate Honors Theses*. 10.
<http://scholarworks.uark.edu/csceuht/10>

This Thesis is brought to you for free and open access by the Computer Science and Computer Engineering at ScholarWorks@UARK. It has been accepted for inclusion in Computer Science and Computer Engineering Undergraduate Honors Theses by an authorized administrator of ScholarWorks@UARK. For more information, please contact scholar@uark.edu.

A MODULAR FRAMEWORK FOR HOME HEALTHCARE MONITORING

A MODULAR FRAMEWORK FOR HOME HEALTHCARE MONITORING

A thesis submitted in partial
fulfillment of the requirements for the degree of
Bachelor of Science in Computer Science

By

Blake Puryear

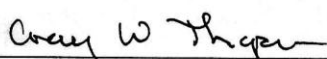
May 2012
University of Arkansas

ABSTRACT

Many patients with chronic health problems have multiple ailments but different patients may have different such ailments. Home monitoring systems for individual ailments exist but a patient may have multiple of these, all designed independently. There are no standard architectures so this leads to unmanageable diversity which causes problems for patients in having to learn to use a variety of monitors and for physicians in trying to monitor many patients. The purpose of this project was to design and prototype a next generation modular remote healthcare monitoring system capable of monitoring multiple ailments and extensible to new ailments in order to explore and evaluate the feasibility of such a one-size-fits-all system and assess a practical way to implement it. The project was designed and programmed as if it were to be deployed in a real world situation using real monitors and a smart phone based monitoring scheme and was also implemented and tested in part using a 3D virtual world, Second Life. Using this virtual world platform provided freedom in exploring some of the alternative designs. Implementing such a system using real world devices and not simply designing it conceptually gave a better view of the future of home health monitoring as well as a better framework for developing a future family of remote monitoring systems. The system was evaluated and was determined to provide a reasonable proof of concept patient monitoring architecture that could potentially influence a next generation of modular home healthcare monitoring systems.

This thesis is approved for recommendation to the Honors College.

Thesis Director:

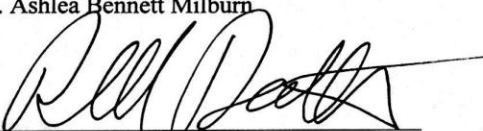


Dr. Craig Thompson

Thesis Committee:



Dr. Ashlea Bennett Milburn



Dr. Russell Deaton

THESIS DUPLICATION RELEASE

I hereby authorize the University of Arkansas Libraries to duplicate this thesis when needed for research and/or scholarship.

Agreed

Blake Puryear

Refused

ACKNOWLEDGEMENTS

I thank Dr. Craig Thompson for his support and advice throughout this challenging project. His encouragement, scrutiny and assistance helped identify the need for this project and his expertise in system architecture influenced the modular design.

I also thank Matt Rothmeyer and John Brady for their dedication to the software prototype portion of this project (as part of our joint Capstone project). Without their implementation efforts, I could not have created the operational prototype we have today. Their innovation and design decisions assisted my work and their input and comments on my design decisions influenced the prototype architecture, design, and implementation.

I thank my parents, Morris and Millie Puryear, for their continued support and commitment to my education over the years. I also thank Natalie Brandon for bearing with me when this project became the focus of my waking hours. Without their help I would never have completed this undertaking.

TABLE OF CONTENTS

1. Introduction.....	1
1.1 Context.....	1
1.2 Problem.....	1
1.3 Towards a Solution	4
1.4 Objective.....	5
1.5 Approach.....	5
1.6 Organization of this Thesis	6
2. Background	7
2.1 Mobile Healthcare.....	7
2.2 Pervasive Computing.....	13
2.3 Device Blind Systems	14
2.4 Application Generation.....	15
2.5 Modular Software Architecture	17
2.6 Trust.....	21
3. Architecture and Design.....	22
3.1 High Level Architecture	22
3.1.1 Sensor Layer	23
3.1.2 Application Layer	26
3.1.3 Data Stream Layer	28
3.1.4 Database and “Business Logic” Layer.....	30

3.2 Mobile Application Design.....	32
3.2.1 Patient Application Mockups.....	32
3.2.2 Doctor Application Mockups.....	36
3.2.3 Devices.....	39
3.3 Application Generator Design	41
3.3.2 Use Cases for Application Generator	44
3.3.3 Application Generator Mockup Screens.....	49
4. Methodology, Results and Analysis.....	54
4.1 Methodology.....	54
4.2 Prototype Development and Results of Prototype Testing using Android.....	55
4.3 Prototype Development and Results of Prototype Testing using a 3D Virtual World.....	60
4.4 Analysis of the Benefits of a Modular Framework for Home Healthcare Monitoring.....	64
5. Conclusions.....	66
5.1 Summary.....	66
5.2 Potential Impact	66
5.3 Future Work.....	69
6. References.....	72
Appendix A. HxM Dev Kit.....	74
Appendix B. FitBit API Information	75
Appendix C. Business Rules.....	76
Appendix D. Prototype Video	80

LIST OF FIGURES

Figure 1: iHealth Blood Pressure Monitor and App	8
Figure 2: Google App Inventor Blocks Editor.....	16
Figure 3: Chrome Web Store Screenshot.....	20
Figure 4: High Level Architecture	22
Figure 5: Sensor Layer Diagram.....	25
Figure 6: Application Architecture Diagram	28
Figure 7: Data stream Diagram.....	29
Figure 8: Data stream Communication Sample.....	30
Figure 9: Alert System	32
Figure 10: Patient Monitor Screen.....	33
Figure 11: Patient History Screen	34
Figure 12: Patient Help Screen	35
Figure 13: Doctor Application Login Screen.....	37
Figure 14: Doctor Application Patient Selection Screen	38
Figure 15: HxM Device on Charging Stand [16].....	40
Figure 16: Class Diagram for Application Generator Model	44
Figure 17: Doctor New Device Plan Screen	49
Figure 18: Doctor Devices Screen	50
Figure 19: Doctor Patients Screen	51
Figure 20: Doctor Device Plans Screen	52
Figure 21: Doctor Settings Screen	53

Figure 22: Patient Monitor Screen (Android emulator has issues with the tab layout)	57
Figure 23: Patient History View	57
Figure 24: Patient Chart View	58
Figure 25: Patient Settings Screen	58
Figure 26: Doctor Application Patient List.....	59
Figure 27: Doctor Application Add Patient Screen	59
Figure 28: Main Generator Screen.....	61
Figure 29: Disease Selection Screen.....	62
Figure 30: Disease Selection Screen.....	63
Figure 31: Inventory Addition Screen	64
Figure 32: Zephyr Bluetooth Packet Structure	74
HR above certain range HR below certain range HR increasing regularly HR decreasing regularly HR up weekly HR down weekly HR major up weekly HR major down weekly NO HR reading for week NO HR reading for day	76
Weight UP Weight DOWN Water LOW Water HIGH Fitness LOW Activity DOWN Activity UP Steps UP Steps DOWN Calorie UP Calorie DOWN Distance above certain range Distance below certain range	76
Weight UP and HR up Weight UP and HR down Weight UP and NO HR Activity DOWN and HR UP Activity UP and HR DOWN Sleep DOWN and HR UP Steps UP and HR DOWN Steps DOWN and HR UP Calorie UP and HR UP Calorie DOWN and HR DOWN.....	77

1. INTRODUCTION

1.1 Context

At the University of Arkansas in the Computer Science and Computer Engineering Department over the past several years, Dr. Craig Thompson's *Everything is Alive* project has been developing pervasive computing demonstrations using technologies like RFID, smart phones, and 3D virtual worlds [1]. We have identified ways to attach RFID tags to real world objects, read the tags with readers (which we will assume will be available in future smart phones) and then use these phones to control the objects and potentially gather readings from those objects. We have also demonstrated how to prototype general-purpose smart objects using 3D virtual world technology. Several of these demonstrations make use of the healthcare domain. Developing some of these prototypes led to identifying the problem considered in this thesis.

1.2 Problem

As the baby boomer generation ages, the need for medical care and hospital visits will skyrocket, as risk for illness increases with age. Insurance companies will seek ways to keep their clients well longer; doctors will look for innovative ways to handle increased patient demand; and patients will look for ways to avoid costly hospitalization. The average cost of a stay in a U.S. hospital is currently around \$9,000 [2]. Many hospital visits could be prevented with proper daily disease management and early intervention on behalf of the patient when health risk indicators are observed. Furthermore, in remote rural locations in the U.S. and in developing nations worldwide, many patients do not have the advantage of being near a local

hospital with current generation medical care. Further complications arise in these areas when patients require frequent hospital visits or continued monitoring of a condition. In these areas, there is a major need for providing quality healthcare at a distance.

A significant step toward service healthcare needs at a distance is home healthcare monitoring, which seems to continuously monitor a patient even between visits to healthcare facilities. A collection of health conditions could benefit from the use of remote monitoring devices. For example, controlling blood pressure can reduce a diabetic patient's risk for heart disease and stroke by 33% to 50% [3]. In a study of congestive heart failure patients conducted by Intel and Aetna, the use of a remote monitoring system enabled over half of the study participants to avoid some hospital stays [4]. The medical community will need a variety of monitoring devices and a variety of ways to connect them to a patient (either all the time or intermittently), a variety of ways to store the data (transmit immediately, or cache locally and broadcast intermittently) and a variety of ways to monitor different types of signal, look for out of bounds health indicators, and then alerting a variety of personnel (the patient, the family, a nurse, a paramedic). A monitoring framework will need to take into account monitoring the same patient for multiple kinds of problems and using rules to identify the correct actions to take.

For a solution to this problem, one looks to the future to redefine what healthcare will become. If, instead of the occasional, inconvenient, expensive doctor's visit or short-term hospitalization, a patient's health could be continuously monitored from home, many expensive health complications could be avoided. Technology exists to enable continuous home health monitoring. However, this technology is not being used pervasively or deployed in forms easy for patients and their doctors to use. First generation home health monitors allow either the

patient to enter data into a desktop or laptop, or allow the monitoring device to automatically input the data from various health sensors. Medical professionals then only sometimes send health data to a monitoring station for review. A new generation of monitoring systems is just beginning to emerge based on smart phones connected to a network of sensors. Currently, these devices are limited mostly to exercise or diet applications and are marketed mainly to the “extreme” health conscious individuals. This thesis presumes that a discontinuity can occur and health monitoring devices can be aimed towards the average healthcare industry patient, to be used by their healthcare provider to better their quality of life. But so far, there are many idiosyncratic ways to build such applications developed for different diseases by different manufacturers - there is no universal framework.

As the need for this type of advanced home healthcare grows and the market rapidly becomes more advanced, continued competition between companies and an inherent fracturing of the market will create a situation in which healthcare professionals will have to make tough decisions regarding device choices for their patients. Will they recommend monitoring “Product X” combined with software “Product Y” or just the more expensive integrated package unit “Product Z”? Packaging these products and combining them correctly will prove troublesome as the number of options grow. As other systems that service multiple devices also develop, complications will increase even further. A standard architectural framework needs to sit in between these devices to allow them to cooperatively interact with one another. This framework also needs to intelligently combine the multiple streams of data from the patient and have some decision making power based on individual streams or combined streams of this information. Without such an integration framework, the number of devices and data streams and software

options will become unmanageable. Doctors will not know what particular devices to use in certain situations and will need a secondary service to determine the best use of the framework for their patients.

1.3 Towards a Solution

One solution to this problem is the use of an application generator that takes in options selected by the professional and uses combinational logic and hard-coded “opinions” of the various monitoring devices, offers a solution and combination of the devices, along with a custom configured monitoring program the patient can utilize.

This application generation portion of the system must be robust and easily upgradeable to keep pace with new devices being introduced as well as radical new ways to monitor patients. This system must also be able to retain knowledge of these systems and have “opinions” on which systems work together well on packages. These “opinions” must have some level of flexibility to provide a rating or recommendation to the doctor. The healthcare professional must not have to spend significant amounts of time to create a device recommendation for a new patient—the ratings must be simple and straightforward. The framework must also provide a modern, well-designed and easy to use graphical user interface (GUI) for patients and healthcare professionals to use. These users may be non-technical so the ease-of-use must be high. One must also take into consideration that users must be able to add new devices and their specifications to the system. The system must be intelligent enough to handle the addition of “custom” products. The system must also be stable and independent enough to have a near perfect up time. The system must be secure and protect patient data.

Clearly, for this new generation of home healthcare monitoring to succeed and undergo widespread consumer adoption, a modular system must exist to connect these systems and allow for easy use for the average doctor and patient. Because of the complexity of this framework, a secondary product should sit alongside it and keep track of sensor-patient interactions. This secondary product, an application generator, should be robust enough to provide recommendations on device use to healthcare professionals and be as modular as the framework itself. Without such a framework, the mobile healthcare market will become fractured and unnavigable for all but trained experts—thus dooming healthcare monitoring to a very expensive “niche market” fate instead of providing an excellent option for the masses.

1.4 Objective

The objective of this project is to provide an architecture, design, prototype, and preliminary evaluation of a modular framework for providing an integrated family of home health monitors.

1.5 Approach

The main prototype of the system exists as a set of smart-phone applications developed in the Android platform. These applications represent a proof-of-concept feature view of the framework system and even interact with multiple current generation home healthcare monitoring devices. The architecture of a future version of this system is the focus of this thesis. While the prototype lacks in some areas due to issues with licensing, cost and lack of current generation devices, the architecture outlined compensates by providing groundwork for the future use of this modular framework. In addition to the architecture for the framework, an

application generator prototype is provided that would use this framework and could provide recommendations on multi-device systems for patients, were the framework ever to increase to an unmanageable number of healthcare monitoring devices.

1.6 Organization of this Thesis

Chapter 2 of this thesis covers background subjects that relate to this project. The background information on these areas is useful for understanding this project. Related work and a literature review are included for each section. Chapter 3 covers the design and architecture of the proposed modular home healthcare system. Chapter 4 covers the methodologies of testing and using this prototype, the results and analysis of this system. Chapter 5 provides conclusions and provides a glimpse towards future work and the impact of this project.

2. BACKGROUND

In this section, the concepts of mobile healthcare, pervasive computing, device blind systems, application generation, modular software, and trust are described to provide the reader with a general background to understand the proposed modular mobile home healthcare framework.

2.1 Mobile Healthcare

The mobile computing marketplace is growing rapidly and includes the healthcare segment. According to a report released in 2011 by Research2Guidance:

“The smartphone application market for mobile healthcare will reach US\$1.3 billion in 2012 – up from US\$718 million in 2011. Despite this substantial growth, the mHealth market is still in an embryonic state – especially in comparison to the US\$6 trillion of the overall global healthcare market. Several factors (esp. smartphone penetration), will continue, however, to drive mHealth market growth over the next couple of years.” [5]

Potential users of mobile health include anyone with a smart phone and anyone who sees a doctor and needs some sort of remote monitoring at any point in their life, which includes close to 100% of the population in countries where healthcare is readily available. A modular mobile healthcare framework, if implemented, would see widespread integration and application across various populations and areas of use. The number of patients using the framework would also increase as the baby boomers age and require more monitoring. Mobile healthcare is a relevant market and an area that software development is currently neglecting.



Figure 1: iHealth Blood Pressure Monitor and App

In its raw form, mobile healthcare is the service and evaluation of patients through a sensor or other mobile device. This mobile device collects a data stream about a patient. This data can then be evaluated by the patient or doctor in a non-hospital setting. Currently, the market is dominated by expensive, less mobile first generation healthcare devices. These first generation devices take the form of small, specialized hardware devices, often connected to computers or laptops, designed to monitor a single, and in a few cases, multiple aspect of the patient's health. Using multiple such monitoring systems, a sensor network can be created for a patient that can accurately monitor some aspects of the state of a single patient's health – but this is at best awkward if two or more heterogeneous systems are required. Still, by combining a caloric intake tracker with a heart rate monitor and Bluetooth connected scale, a more complete picture of a patient's health is presented than if we were only monitoring once metric at a time and only during annual checkups. More advanced devices can be used on patients with specific afflictions, such as glucose monitors for those with diabetes or blood oxygen content monitors for those suffering from heart problems.

As mentioned, most current monitoring devices are tied to their own software application and are disjoint from any framework that allows these devices to communicate and share information. This is a problem because, often, only by combining data from multiple sources can a true picture of health arise.

Second generation portable monitoring devices are used in our prototype architecture, based on sensors and smart phones. These devices can monitor a single or multiple aspects of a patient's health and can then deliver that information to the patient via a mobile application. These monitoring devices are more commonly referred to as sensors. Ko et al. provides some background on these sensors in "Wireless Sensor Networks for Healthcare,"

"There is a long history of using sensors in medicine and public health. Embedded in a variety of medical instruments for use at hospitals, clinics, and homes, sensors provide patients and their healthcare providers' insight into physiological and physical health states that are critical to the detection, diagnosis, treatment, and management of ailments. Much of modern medicine would simply not be possible nor be cost effective without sensors such as thermometers, blood pressure monitors, glucose monitors, electrocardiography (EKG), photoplethysmogram (PPG), electroencephalography (EEG), and various forms of imaging sensors." [6]

Second generation mobile healthcare sensors are now being marketed to consumers instead of just healthcare professionals and are being connected to the Internet and to personal computing devices and are rapidly gaining traction in the mobile marketplace. These sensors are being sold as fitness products, to help a user become healthier or gauge their marathon pace with greater ease. Few of these commercial devices are being used in a true healthcare settings. The modular framework proposed seeks to change this mindset. The use of these devices, software and sensors (Mobile Healthcare Framework) in the professional healthcare realm and a hospital setting could increase the availability of professional advice on many common health issues.

Mobile healthcare applications are a growing market segment in application sales in the Apple App Market and Google Marketplace. TechCrunch reports:

“Early last year, PEW Research was already reporting that 17 percent of mobile phone users were using their devices to look up health and medical information, and Juniper recently estimated that 44 million health apps were downloaded in 2011.” [7]

These marketplaces are constantly being updated with mobile healthcare applications but the majority of these apps do not interact with sensors but are rather focused on providing logged health information. The impact of these applications is limited and when personal sensors become more prolific, the use and development of these applications will need to expand to cover real-time data feeds. A mobile framework utilizing these mobile health sensors and modern application design standards is needed.

This shows that a dramatic number of mobile device users are interested in mobilizing their healthcare with their smartphone alone and therefore it appears there is a market for type of framework proposed in this thesis. Such a framework is expected to benefit the patient-doctor relationship:

“...remote patient monitoring — by way of using the smartphone as a hub, — will significantly lower the cost of mHealth services, because it will create a reduced need for costly, tailored devices. In terms of which sectors are out in front, the Juniper report said that the monitoring of cardiac outpatients has become increasingly popular, “as insurance reimbursement in the U.S. market plays a key role.” Next, one can expect to see remote monitoring playing an increasingly central role in the management and ongoing treatment of chronic diseases, specifically of diabetes and Chronic Obstructive Pulmonary Disorder.” [5]

Rapid adoption of mobile healthcare devices is occurring—but with the current lack of a unified framework, the industry is poised to deliver fragmented, expensive, hard to use products. A logical solution is a modular framework that can rapidly adapt to a growing and changing market as well as meet the needs of patients and doctors nationwide and worldwide. The key

would be creating a product that not only competes with the current fractured “single application single device” market but also gives patients and doctors an incentive to switch to the “single application multiple device” style of healthcare. Clearly, the market exists for a modular framework to succeed. While adoption in professional healthcare settings is low, the general consumer populace seems open to bringing in this level of technology.

In recent years, development and conceptual work with mobile healthcare platforms has increased and interesting research has been done in the area. A thesis by J. Li at National Chung Cheng University on “Understanding the Acceptance of Mobile Device of Wireless Health Care” [8] reports on many factors that affect the adoption rate of mobile medical technology in a professional setting. Li notes that adoption rates for health professionals appear low and attributes this to what the professionals perceive about the device or software. The effects of what a healthcare professional sees as useful in the workplace account for much of this. Li finds that “perceived service availability has a positive effect on perceived usefulness” which results in a quicker and higher adoption rate of mobile health technologies. Essentially, the more useful and available a healthcare professional sees a device, the more quickly the professional market is to adopt that device. A modular framework such as the one this project proposes could see a quick adoption rate if it provides quick benefits and none of the drawbacks of other current generation mobile healthcare applications. The choice of devices used is something that should promote speedy adoption among choosy healthcare professionals. Furthermore, Li goes on to cite other key aspects that increase the adoption rate including connection speed, ease of use, and personal innovativeness [8]. Combining these factors, Li produces an effective structure model that shows the influences of various aspects of mobile healthcare devices on healthcare

professionals. This structure model should be used as a guideline for the development of any telehealth device or framework. Mobile healthcare developers should seek to address these issues if they wish to see professional healthcare integration.

Other work on wireless sensor networks is more hardware focused in areas like low-power microelectronics and wireless sensors. This research proved useful in the layout of the framework. Writing in 2005 in “System Architecture of a Wireless Body Area Sensor Network For Ubiquitous Health Monitoring,” Otto et al. described the composition of the sensor network, which guided the design of the sensor layer of the architecture:

“Recent technological advances in wireless networking, microelectronics integration and miniaturization, sensors and the Internet allow us to fundamental modernize and change the way health care services are deployed and delivered.” [9]

While use of PDAs in their system architecture is outdated, the concepts and motivation argue for a modular mobile healthcare framework:

“During the last few years there has been a significant increase in the number and variety of wearable health monitoring devices, ranging from simple pulse monitors, activity monitors, and portable Holter monitors, to sophisticated and expensive implantable sensors. However, wider acceptance of the existing systems is still limited by the following important restrictions. Traditionally, personal medical monitoring systems, such as Holter monitors, have been used only to collect data. Data processing and analysis are performed offline, making such devices impractical for continual monitoring and early detection of medical disorders. Systems with multiple sensors for physical rehabilitation often feature unwieldy wires between the sensors and the monitoring system. These wires may limit the patient’s activity and level of comfort and thus negatively influence the measured results... In addition, individual sensors often operate as stand-alone systems and usually do not offer flexibility and integration with third-party devices. Finally, the existing systems are rarely made affordable.” [9]

Many of these problems still exist in the market today and have not yet been addressed by a widely adopted framework. Still, utilizing smartphones and the fast burgeoning app marketplace should accelerate adoption of mobile health solutions.

2.2 Pervasive Computing

Pervasive computing is the vision of a future where everyday objects contain sensors and communicate with one another in a vast “Internet of Things” that provides real time information on the entirety of one’s surroundings. These sensor networks would be linked to our mobile healthcare framework to integrate lifestyle choices with healthcare data. The general concept of pervasive computing is described by Thompson and Hagstrom in their article on “Modeling Healthcare Logistics in a Virtual World:”

“...everywhere we look, we can see evidence of the pervasive computing macro trend. In the future, everything is alive and can sense, act, think, feel, and communicate. Networked objects (labeled EiA inside) will include equipment, vehicles, robots, clothing, pets, chairs, scissors, and trees. Sensors, actuators, controllers, and communication devices will be vanishingly small and virtually free (think RFID tags and smart dust). They’ll attach to their hosts transparently and intercommunicate and cooperate to help us with our tasks. This Internet of Things will extend the desktop metaphor so that computing is no longer just inside computers but rather inside many physical things.” [10]

Rapid adoption of pervasive computing is related to the idea of networked communication becoming the focus of computing, exactly the foundation that the proposed mobile healthcare framework is built upon. Therefore, an understanding of pervasive computing is fundamental to understanding this architecture. The idea that this framework would one day be linked to other like frameworks or even connected to non-healthcare devices in general means the idea is not limited to just healthcare. Thompson expounds on the future of pervasive computing in his article titled “Everything is Alive:”

“In the EiA future, individual toys will have personalities and will play with children and each other; they’ll help children learn to read and do their homework. The sprinkler system will communicate with your yard’s plants to see how much water each needs and scan the weather channel for news of rain to adjust water rationing accordingly. While exercising at the gym after work, you’ll be able to ask the weight machine to tell your oven at home to turn on at 6:30 p.m. because you’re running late. As you walk to your

car, you'll answer a call from your mother and have a good chat while driving to the shopping mall — all without a visible phone. The clothes racks at the mall will sense your profile and accentuate the right sizes and styles as you walk by. After you purchase a new jacket, wearable computing embedded in the cloth will begin operating on your behalf. At the grocery store, you'll just pick up the items you want and sensors in the store will automatically debit your grocery account as you leave.” [2]

The addition to healthcare monitoring on top of all of these systems will create a real time sensor network of an individual's life. The linkage of these systems could provide doctors feedback on activities or lifestyles that are detrimental to one's health. This would benefit individuals wishing to live healthier lifestyles or insurance companies wishing to “check up” on problem clients. The use of this platform would not be limited to healthcare professionals but anyone wired in to the “Internet of Things” could make use of their own data to further the enhancement of their life. Because of the importance of health in our lives, healthcare monitoring would likely be one of the driver applications that bring about comprehensive pervasive computing that affect many areas of our lives.

2.3 Device Blind Systems

An important concept to this project is the idea of a device blind system. Also referred to as “device blind” or “device agnostic,” this is the idea that our software system will be ignorant of and insulated from what specific device is connecting to it, but will rather focus on the packet of data that it is receiving. Current mobile applications are device-centric, in that they focus on connecting to a specific device only and look solely for that type of nonstandard input. The entire framework must be device-blind to support modular evolution. Without this “blindness” that ignores device brands or types and seeks only for standardized input, the developers of the framework would need to account for each possible type and brand of device and account for an

unlimited number of inputs, making device integration infeasible. Therefore, the architecture will not be focused on what brand or type of sensor is connected but instead, on what type of data is being gathered from that device and how to transmit and store it on our server.

Effectively, the system seeks after a known format and type of input for its data acquisition rather than using the current method (single device to single device) of mobile healthcare device communication.¹ This reduces the complexity of our framework as there is more freedom to focus on data transmission and formatting that data correctly than adding in what types of devices this framework is compatible with. At the same time, this complicates our work. Since the system will not be interested in what type of device is connecting, we must standardize the input the framework accepts and what is sent from the devices themselves. The complexities behind designing this system to be device blind were not addressed in the prototype, since we were only working with two devices. However, were the framework to grow to ten or more devices, some having the same functionality, then the “device blindness” issue would become a larger factor in the development of the framework.

2.4 Application Generation

Application Generation is the use of software applications to create other applications. One example of an application generator is Java Application Generator (JAG) [11], a Java-based open source application that generates working J2EE applications. Although the application is simple, the concept inspired the UI design and backend architecture of the modular home

¹ An alternative is standard application program interfaces and is described in more depth in the final chapter under future work.

healthcare framework. Another application generator is Google's App Inventor, now defunct but being resurrected, which is a web based framework that allows novice programmers to quickly create advanced applications for the Android platform. Google's App Inventor provided a good model whereby third parties could more easily generate applications using a recipe. With Google's App Inventor,

“You build apps by working with:

The App Inventor Designer, where you select the components for your app based from a wide range of predesigned components with specific functions and settings.

The App Inventor Blocks Editor, where you assemble specific program blocks that specify how the components should behave. You assemble programs visually, fitting pieces together like pieces of a puzzle. Components interface with one another visually and settings can be modified to best link the different modules together. Creating a program can be as complex or simple as the user wishes—there is a wide range of complexity in using the Blocks Editor.

Your app appears on the phone step-by-step as you add pieces to it, so you can test your work as you build. When you're done, you can package your app and produce a stand-alone application to install.” [12]

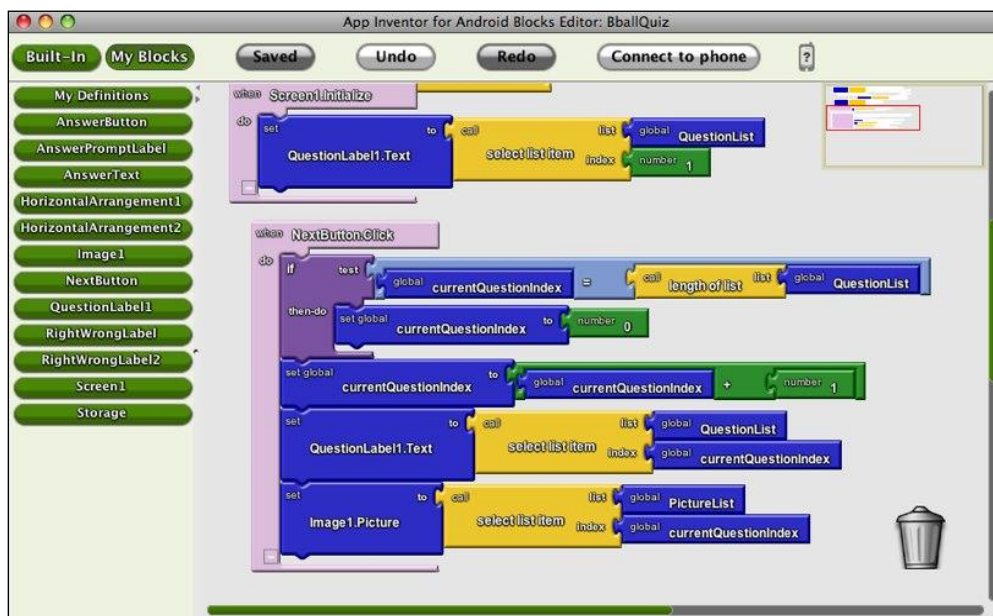


Figure 2: Google App Inventor Blocks Editor

Overall, simple application generators are scarce and those that do simplify things tend to oversimplify and become useless tools. Striking a fine balance between features and usability is crucial.

2.5 Modular Software Architecture

A significant concept that the prototype framework is focused around is modularity. This section describes what modularity is, some background research on modularity and the steps that can be taken to promote modularity. Gabriel in “Definitions of Modularity” states that:

Modularity is the compartmentalization of information that gives rise to behavior and information that achieve a coherent purpose in a (running) system—living or artificial. [13]

In the case of this framework, modularity is the intentional design of the framework to leave hooks for third parties to add additional plugin functionality to support for their own hardware or software. Since we cannot fathom how healthcare devices will grow and change over the years, leaving software and database hooks for devices to plug in is crucial to the development of the framework. In his thesis on “Software Modularity as a Systems Design Principle,” Rellermeyer states:

“...systems which can react to changes in their environment with seamless adaptation so that, without any further intervention from outside, they always operate in an effective manner.” [14]

This standard is what the modular home healthcare framework targets. The ability to adapt to new devices and healthcare monitoring standards without major overhauls in the core of the framework is crucial. This plugin functionality independence ensures longevity for the framework.

Compartmentalizing portions of the entire framework and ensuring that there was a straightforward way to add new functionality to the framework was a challenge but was accomplished, in part, through code design and assistance from the Android framework. Modularity was also a priority during development to ensure that our Bluetooth connections were easy to add to—for the consideration of new devices. A problem, however, arises when a new device does not support Bluetooth or another wireless standard. Currently, no resolution exists for this problem but could be compensated for by including a method of data entry that is not wireless-based but rather based around the user entering their own data. Without the ability to adapt to the ever growing and changing market of mobile healthcare, the framework will die as soon as the devices it supports become outdated. Currently, mobile devices are being outdated and replaced on a frequent basis, sometimes several times a year. For instance, in the case of the FitBit device used later in this thesis, “FitBit Ultra” was released within a year making the original FitBit obsolete. Continued support of modularity throughout the development of the framework is necessary to ensure the quick adoption rate that one would hope to see out of such a large and revolutionizing product.

Next we ask, how should this framework be made modular? Modular software can come in many different forms, from simple code that can be broken off and repurposed, to actual third party modules, to extensions of current sections of the code and a final concept of an “application store” that serves as the resource for all possible modules. We will not want to open the entirety of our framework to piecemeal code modification. Only parts of our connectors will be open to public use, e.g., via an API for a web service. The framework should not necessarily sit at the other extreme either – the ability to add plugins should be free. Monetizing plug-ins is

feasible if the framework becomes widespread and device manufacturers want to include “professional” type software plug-ins with their devices, but this should not be the focus of the framework’s extent of plug-in architecture.

It should be clear why the extent of the modularity of this framework should sit somewhere in between the degrees of modularity explained. We risk compromising the integrity of the framework by choosing a side. For our framework’s degree of modularity, a good example is of Google’s Chrome Web Store, which offers plug-ins that extend functionality rather than change it completely, modifying and customizing a user’s experience. Chrome provides a developer API to Chrome Web developers to ensure that privacy issues are not breached; however, developers are given a wide range of freedom in the Chrome Web Store with many different types of plug-in applications being available. This example is modularity at a grain most favorable to this framework.



Figure 3: Chrome Web Store Screenshot

For a modular home healthcare framework, free modules should exist that can easily be added or removed, all of which should be widely available in one location and each module interfacing with the parent software through the same protocols. If we take the modularity of the framework to one extreme or the other, we face issues as the framework ages and adds additional devices. Should the modularity be limited or the creation of modules made too complex by allowing code modification of the entire framework—the opportunity of widespread adoption is limited. Should the modularity be extreme to the point that a marketplace is required to manage an ever growing list of add-ons, we risk monetization of a layer of the framework that may make it too complex. This framework is not designed around creating a profitable “app market” but

rather around providing a device aggregation and healthcare alert service unlike anything that exists in today's marketplace. The modularity should be balanced to foster and encourage growth but may never reach the point of monetization for "module packs" or "apps." The monetization of the home healthcare framework itself could prove devastating in that it might lead to competing frameworks and thus a fractionalized market. Carefully guided development and support from a developer team and industry standards could continually improve the framework to avoid this perilous path. In conclusion, modularity must be carefully designed and balanced in order to achieve a desirable end result for this framework.

2.6 Trust

A modular framework for health care requires one more ingredient, which we term "trust" to mean that the framework is vetted by trusted authorities. Apple's App Store provides an environment wherein all apps must be vetted by Apple to be included in the App Store. In a similar way, it would be useful if all plugins for a modular home healthcare framework followed the protocols of the framework but were also vetted to insure compliance with HIPPA laws, data and software security requirements, and perhaps other framework requirements for including business rules for integrating healthcare devices and their specific business rules into a marketplace containing many other such devices. In the same way that doctors must monitor drug interactions, there may be unforeseen device monitoring interactions when introducing new devices. For now, we note the problem and leave this as an open research area.

3. ARCHITECTURE AND DESIGN

3.1 High Level Architecture

The high level architecture has four-layers: the sensor architecture, application architecture, data stream layer and business rules/database architecture. Data is captured at sensor(s) attached to a patient or in the patient environment and may flow to a mobile device which connects wirelessly to a fixed healthcare infrastructure. Some steps like data analysis and interpretation can be performed on the mobile device or by the doctor or healthcare analyst for reporting to the doctor or directly to the patient.

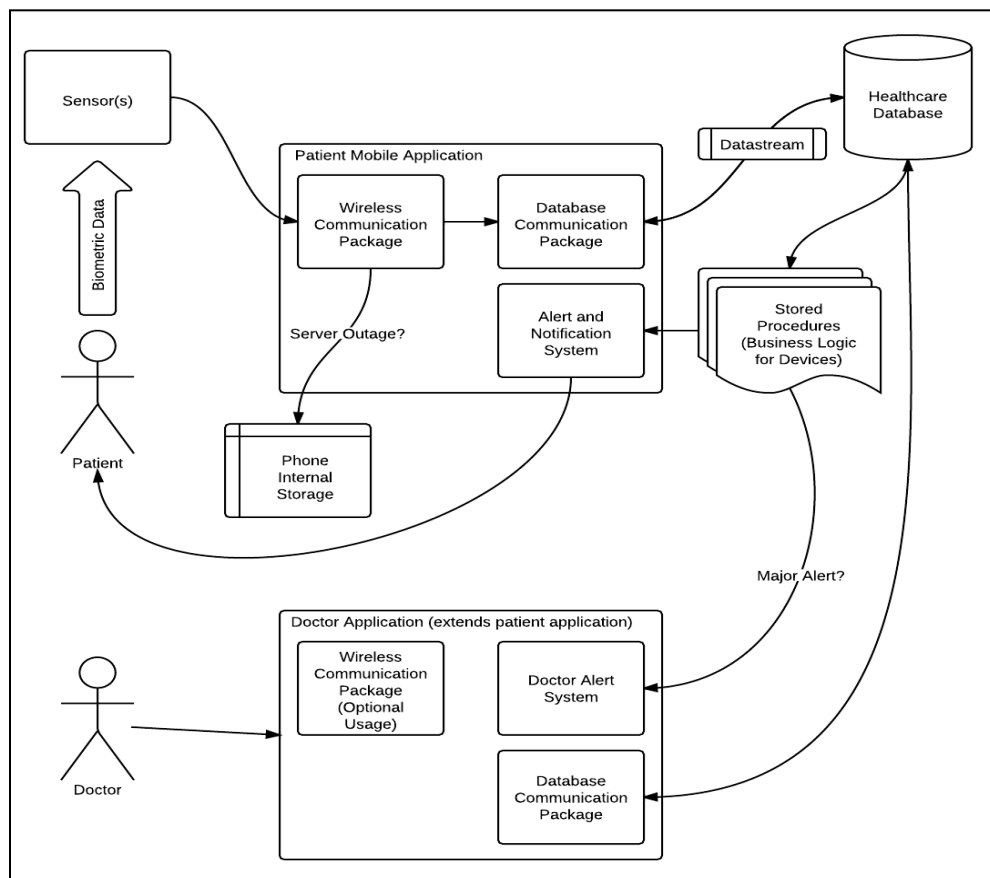


Figure 4: High Level Architecture

3.1.1 Sensor Layer

The sensor layer of architecture is composed of devices that gather data from a human in any number of ways, e.g., from biometric scans, weight scales, blood tests or oxygen levels. Each sensor device has a method of communication with the framework. The sensors can be linked together or may also be separate from one another. Their communication protocols include wireless protocols like Bluetooth or WiFi or can be tethered, and can be connected always or intermittently. The data can be added automatically from a sensor or manually. Each sensor device should gather a biometric dataset from the patient and format it in the framework's standard data stream format which is transmitted to a database after data point acquisition. It is important that the framework can accept intermittent and/or continuous device monitoring since no matter the method of data transmission, any mobile healthcare device should be able to fit into the framework because both kinds of sensors can exist and so the framework needs to be inclusive.

Requirements for data capture are as follows. The sensor device must be mobile so that it can be present in one's home or worn on one's person. The device may be removable from the human body after testing (the framework may or may not later include but does not address them at present). The device must be accurate, so that the data points gathered are of use to healthcare professionals and to the patient. The device should be unique in function though variations or brands of devices are allowed. The framework will not make a distinction between Brand X heart monitor and Brand Y heart monitor, as the data stream the framework receives from each of the heart monitors should be identical. The device should be minimally intrusive and be able to be used in an average patient's home. This means that it must not require surgical

implantation or be overly time consuming to use. These guidelines will improve usability of the framework.

The sensor network must be secure. Personal information privacy is critical in the healthcare realm and many privacy laws surround health information and must never be violated. HIPPA compliance is a set of rules that provide

“federal protections for personal health information held by covered entities and [give] patients an array of rights with respect to that information. At the same time, the Privacy Rule is balanced so that it permits the disclosure of personal health information needed for patient care and other important purposes.” [15]

HIPPA compliance is necessary for the device to be used in any segment of public healthcare and with the nationwide healthcare solutions being implemented, acceptance by the federal government for any framework is necessary. The sensor data must not be stored or accessed locally all the time, but rather will be transmitted to a remote database. Data should only be cached locally when the remote database cannot be reached. All data must be encrypted when sent so that it cannot be read by any source that is not our framework’s logic node layer.

The sensor network must transmit consistently and accurately to the logic node, housing the database and where business rules are processed. Without consistent data, healthcare professionals have less opportunity of recognizing problems. Considerations must be made for outages, storing data temporarily on the mobile device in case of server issues. Considerations must also be made for incorrect data: judging if a certain data point is too far out of “range” to be considered correct. For example, a heart rate reading of 9999 beats per minute would be flagged as incorrect.

The sensor network must be standardized for its data transmission. This will reduce clutter on the database and offload some of the processing work from the logic node. Data

transmission standardization is covered in section 4.1.3. Device manufacturers must agree to, at the very minimum, offering this style of data transmission as a setting on their devices. If a device does not support this transmission protocol, it must not be included in the framework; this prevents additional development time in adapting the framework to use new formats of data.

The sensor network layer of the architecture accommodates an extensible number of sensor devices from multiple manufacturers along with a potentially similar number of connectors to our database. The sensor network is composed of many nodes and connectors but by itself lacks end-user view controls.

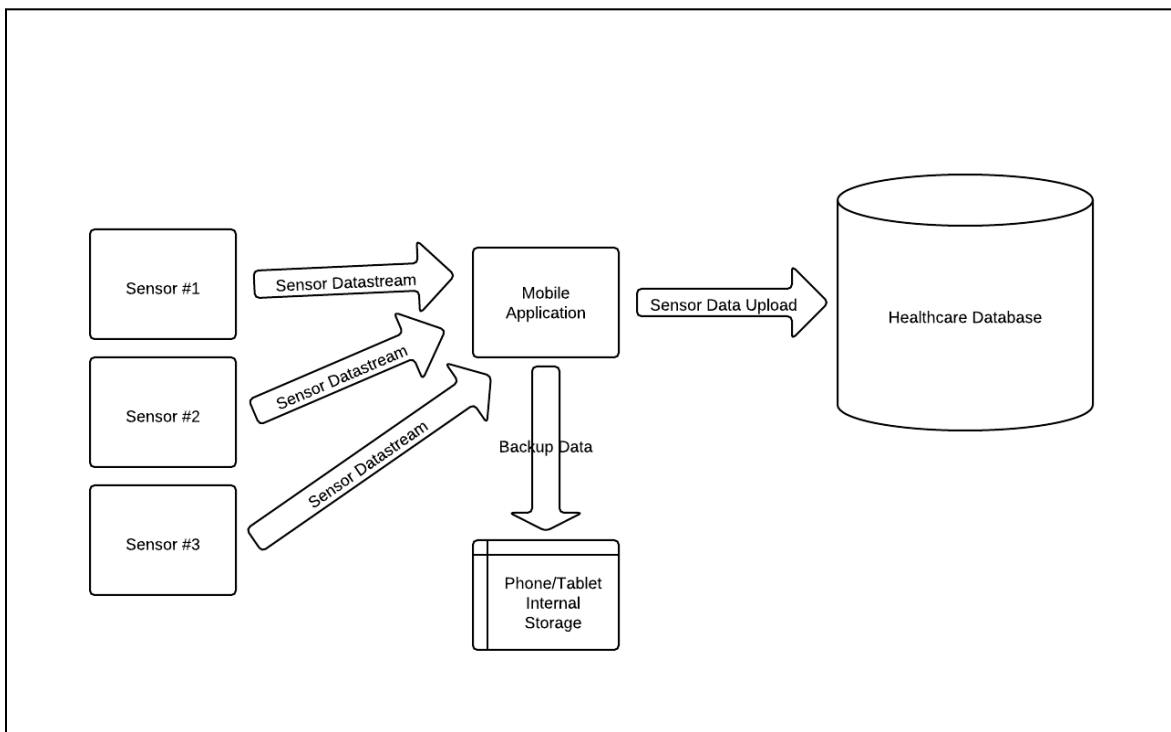


Figure 5: Sensor Layer Diagram

3.1.2 Application Layer

The application layer of this architecture provides the “view” of the framework that the user will interface with. The application layer also functions as the “go-between” for the sensor architecture and the database. The connections to sensors for the application layer will vary from user to user, but this layer can pull and push relevant data from and to the database.

There will be different kinds of applications based on the installed sensors and the user’s need, each one determined through the application generation process. Requirements and design considerations are discussed in the following paragraphs.

The user of the application needs to be taken into consideration and the interface should be designed around that. Complexity needs to be minimized for the patient and large colored buttons need to be provided as one alternative for those not used to using mobile applications. The interface should be tab-based to encourage further modules and ensure the application can be expanded without worry of a total redesign. A notification system should be included. This creates the color-coding health scheme to be discussed in section 3.1.4. The notification system could also be abstracted to utilize emails. Email alerts may be a necessity since the patient might be more likely to read a permanent email than a small notification on their phone. The patient’s application layer should be advanced enough to provide a comprehensive picture of health for the particular patient. Graphs and charts should be provided that are easy to read and present relevant information in an accurate and consistent manner. Graphs may also be present in the patient application for personal use, but with some data obscured as full doctor-patient information overlap is not desired.

The doctor application is entirely different. It should be easy to use, but user acceptance is not as important a focus. More buttons and features can be used and the GUI may be more “raw.” The doctor app should have a consistent user experience with the patient application. Finally, the doctor application should allow for comparison of multiple data points across multiple patients and a review of patient history.

The connectors in this part of the architecture are simply connecting the devices to the mobile architecture and a connection to the database that transmits the information pulled from the sensor devices. The details of the connectors are minimized as transmission is simply passing from the sensors to the database. Nothing is stored locally except in the case of an outage and then data is cached, but later deleted after transmission.

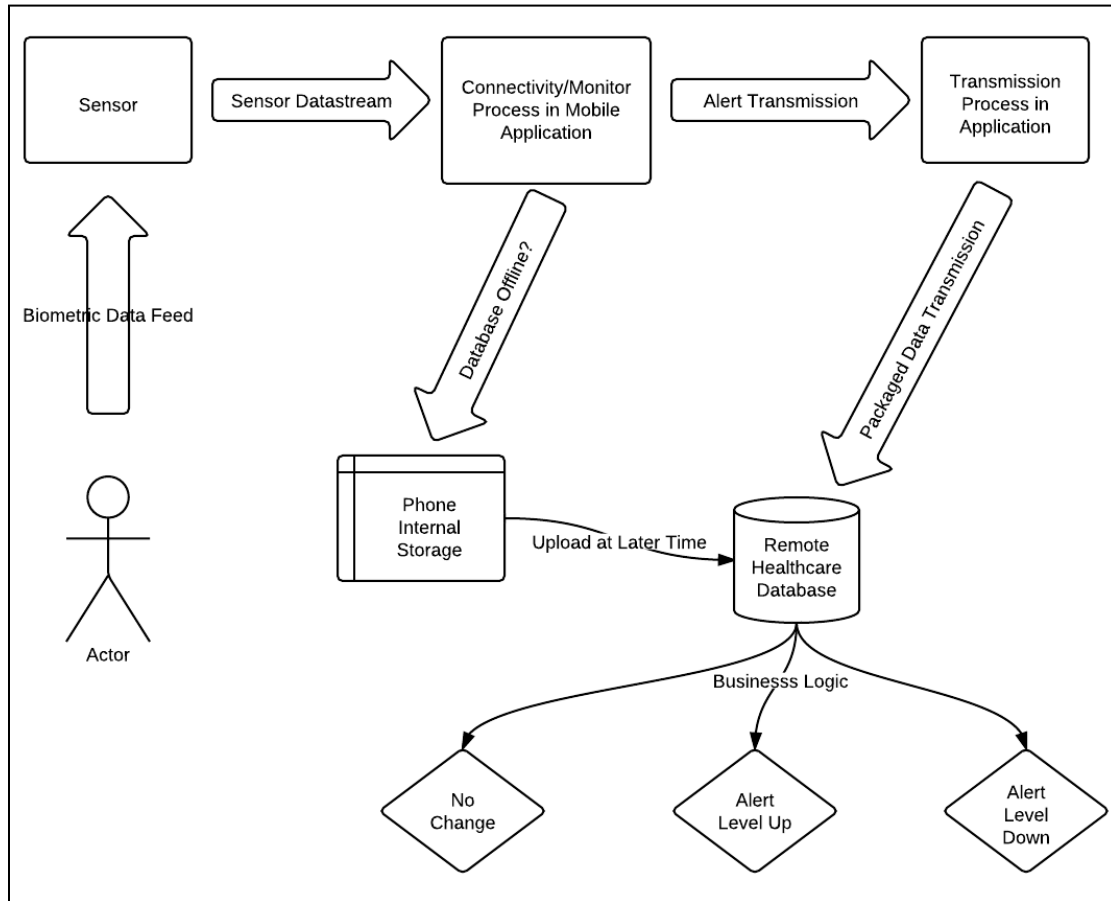


Figure 6: Application Architecture Diagram

3.1.3 Data Stream Layer

The data stream layer consists of a standardized format of data split up into segments. This stream is transmitted intermittently or continuously depending on the device sending the packet of data. This data stream contains specific segments that are known to the rest of the framework—easing the load on the database to interpret the incoming data. The database will only accept this format of data and will therefore minimize data heterogeneity problems. The model data stream would be relatively compact and composed of at least five fields of data: the first field would be a patient identifier represented as a number or a string. The second field

would be the doctor identifier, again a number or a string. The third field would be a date and timestamp. The fourth field would be a device identifier so the framework knows what type of device it is seeing data for. This should be a number joined to a database entry containing more information pertaining to the device. The final field would be the actual sensor data, in whatever format that type of data is stored in: an array of integers for heart rate, a single integer for weight or an array of strings for calorie intake.

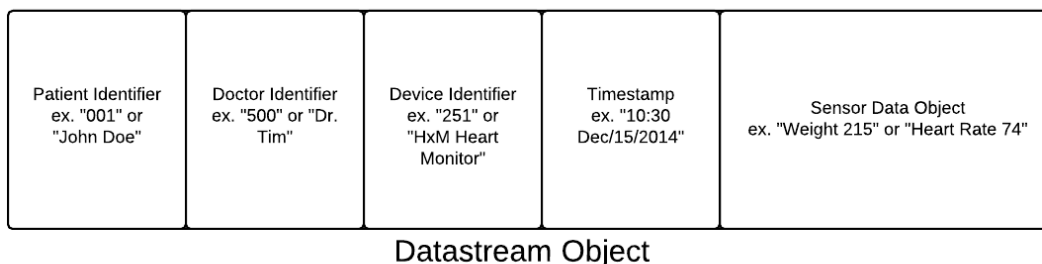


Figure 7: Data stream Diagram

Standardizing the data stream allows the creation of business rules that will be discussed in the next section. This data stream is potentially able to be used in other frameworks and have other uses besides health data aggregation. Social networking of health or fitness “competitions” could be aggregated from the data stream, should the device or framework allow for that level of API access. Pertaining to security, the data stream encrypted during transmission. Figure 8 shows the fields unpacked as they are stored in a relational database and the biometric data is pushed into the business logic processes.

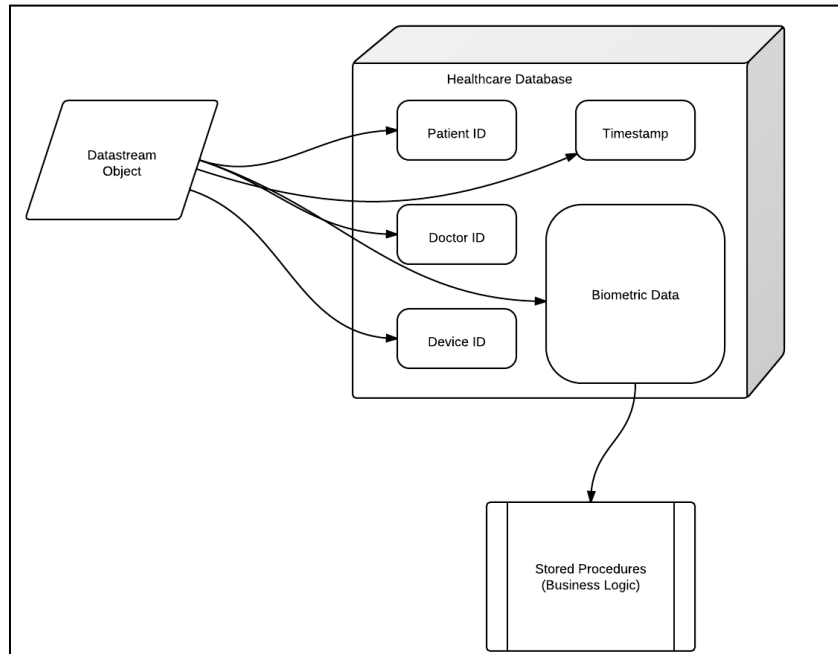


Figure 8: Data stream Communication Sample

3.1.4 Database and “Business Logic” Layer

The final layer of architecture in this system, previously referred to as the logic node, contains our database and “business logic”. This layer will initially host a single database (patient data) and will grow to incorporate multiple databases as the framework expands. These future databases should include lists of diseases, treatment plans or relational data on past treatments that could prove especially beneficial to doctors. The specific database product should not be a factor—support for SQL stored procedures and web connectivity are the only hard requirements.

This layer is connected directly to the application layer and receives input in the form of the data stream. Once the database receives the information from the application (passed on from the sensor) it will analyze the new data using business rules. These logic rules apply to

each data point and look for trends in the data. See Appendix C for the list of rules in the prototype for an example of the logic being used. The business logic is guided by the principal “if something is interesting, provide feedback.” Interesting, in this framework, is operationally defined by rules that seek increasing, stable or decreasing trends and patterns. If the logic detects an increased resting heart rate and an increased caloric intake over the course of multiple weeks, it sends a notification to the patient explaining what was found and a suggestion on how to counteract this problem. The notifications and suggestions follow a stoplight color tiered system. *Green* for a positive change; *yellow* for a problematic but not serious change; and *red* for a dangerous change. The previous example would trigger a “yellow flag” indicating a present but not immediate threat. Green flagged trends send a positive notification. Red flags alert the patient of a serious problem and notify the doctor of the specific issue as well. As a matter of importance, the numerical values guiding from one color level to another are not hard-coded but are instead fuzzy logic “zones”. Because a single data point out of range is an anomaly, it is not desirable to alert a doctor because of a single outlier. Defining the boundaries of these fuzzy logic health zones is implemented in business rules, but will need calibration for different patients who use the system. These zones must be loose enough to allow for small fluctuations but not allow such fluctuations to go unchecked. An attempt at utilizing this style of logic is partly implemented in the current prototype but due to device and user restrictions some hard data and static levels are utilized in favor of fuzzy logic. The alert system must be robust enough to provide different levels of notifications to the patient depending on the degree of violation of these fuzzy logic zones. The use of health zones and the stoplight coloring system encourages the patient into better health instead of relying solely on infrequent doctor visits.

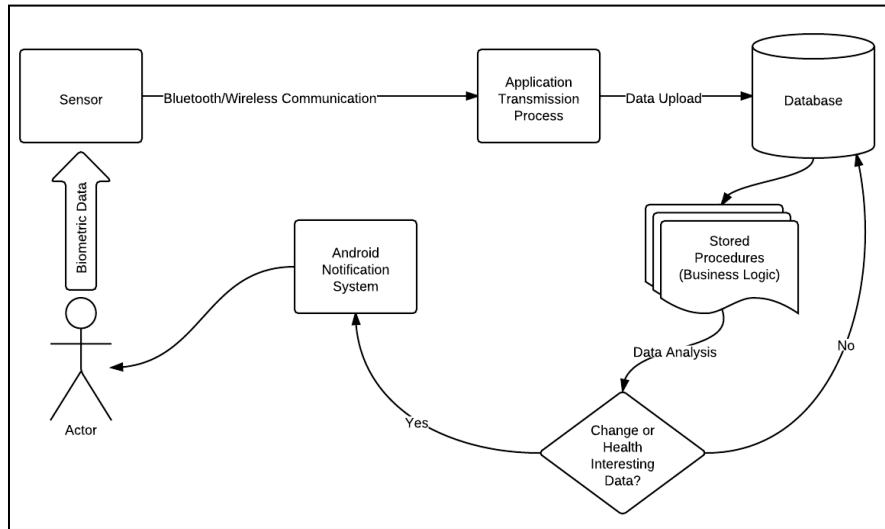


Figure 9: Alert System

These business rules are designed on a device-by-device basis and are related to one another via connections made between the procedure files. Adding new rules is not a complex task—logic combining the new device analysis with existing ones can be done in a matter of minutes. A device manufacturer could provide the framework developers with a list of rules and how they wish them to interact with other devices. The framework developers could then integrate this functionality with the currently existing rules. The database and logic architecture layer moves this framework from a data storage medium into an intelligent monitoring system.

3.2 Mobile Application Design

3.2.1 Patient Application Mockups

The patient application of the mobile home healthcare framework is a horizontal tab based smartphone application centered on usability and user experience. Currently, three tabs are present. One for patient history, the second one for monitoring using a device and the third one

as a help tab that walks the patient through application use. Dropping in new tabs with new functionalities is not complex. The modularity of the application is improved (by the usage of tabs) over the use of a menu based or single pane mobile application. The UI design decisions were made based on considerations for the patient—large buttons were used, color usage was considered and the UI flow was kept consistent with popular mobile applications.

3.2.1.1 Patient Application Screens

The following section demonstrates the design of the application in smartphone mockup format. The mockups were done in iOS format due to available resources and do not reflect iOS exclusivity of the framework. These should be considered generic smartphone mock up diagrams.



Figure 10: Patient Monitor Screen

Upon opening the application, the patient is presented with the monitoring screen. This is the most used pane and should be the first thing the user sees to minimize taps and increase productivity. The tabbed “accordion” style layout seen in Figure 10 is consistent with the high level architecture described in section 3.1. Single buttons are made clear and large for patients with visibility issues. Options are kept at a minimum and the statistical information provided is large and highlighted.

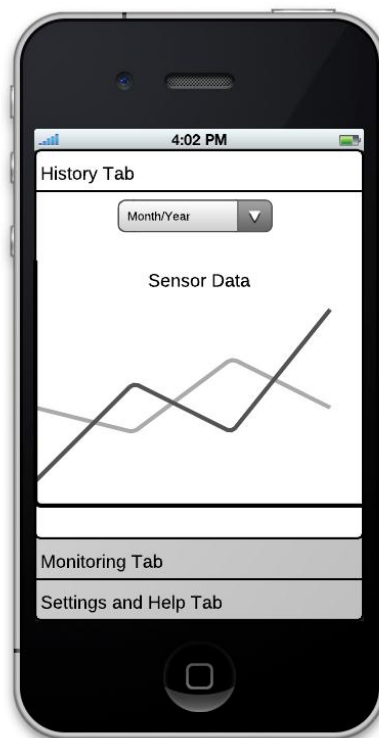


Figure 11: Patient History Screen

The history tab in the patient monitoring application provides a set of graphs for each set of biometric data. The graphs provide the user with visual feedback of health status and allow the user to plot their progress. The graphs are designed to convey the healthcare data in a straightforward fashion.

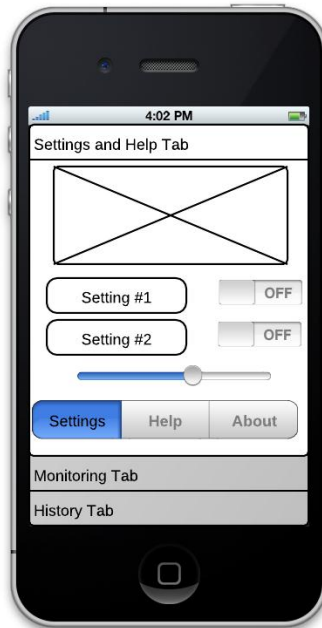


Figure 12: Patient Help Screen

The patient help screen provides information on how to alter settings in the application. The limited settings provided will allow some customization but all major control is delegated to the healthcare provider. The help tab seen at the bottom of the screen switches the current view to a help document providing instructions for connected devices and for general application tutorials. The about tab provides information about the developer and relevant documentations pertaining to healthcare and HIPPA compliance.

3.2.1.2 Patient Alert

The pop up should not take the form of a banner that could be easily missed. The pop up should take precedence over everything on the screen and should be forced to be cleared before

the user can return to normal phone usage. An alternative alert could simply use the phone's built in email client to notify the user.

3.2.2 Doctor Application Mockups

Visually, the doctor application it is similar in appearance to the patient application to promote a consistent design pattern. For security reasons, the doctor must log in each time they open the application. The doctor may then compare different patients by selecting names from a list. Real world checkups do still exist and the framework would require that data to be added into its database for comparison purposes, therefore the doctor can initiate a monitoring session. The doctor is more likely to be concerned with a finer grain of detail than the patient and thus the doctor application has much more control over graphing utilities. The doctor also has more freedom in modifying and adding patients—a patient may not monitor their own file.

3.2.2.1 Doctor Application Login and List Patients Screens

This section describes the doctor application and provides a walk through showing the use of the application via smart phone mock ups. These were done in iOS format due to available mockup resources and do not reflect an iOS exclusivity.



Figure 13: Doctor Application Login Screen

The doctor login is required upon each opening of the application (due to HIPPA) and will time out after inactivity. The framework should acknowledge healthcare laws and make conforming to these a priority—otherwise widespread market integration may be an impossibility due to legal issues.

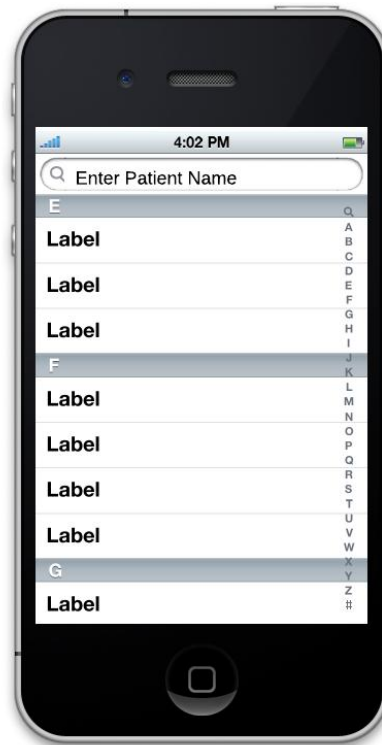


Figure 14: Doctor Application Patient Selection Screen

The patient selection screen is a list view of patients in the healthcare database assigned to the doctor logged in to the application. Clicking on a patient opens their “version” of the monitoring screen and allows the doctor to view and modify all patient data.

3.2.2.2 Doctor Application Patient Monitoring Screen

The doctor application patient monitor is the same as the patient application. The doctor is provided the same options to test a patient in-clinic and record the data. This functionality is provided as a convenience for a checkup or annual exam.

3.2.2.3 Doctor Application History Screen

The doctor application patient history is the same as the patient history, except with the ability to handle comparing multiple patients at once.

3.2.2.4 Doctor Application Settings Screen

The doctor application settings screen functions as the patient application, with the inclusion of additional settings related to comparing patients as well as modifying patient data.

3.2.3 Devices

Current devices being used in this framework are the HxM Bluetooth Heart monitor and the FitBit health communication device. These two devices were chosen because of their ease of use, popularity, availability and price. Both devices cost less than \$150 and therefore are more marketable than more expensive devices.

The HxM Bluetooth Heart monitor is a small sensor that is worn using a strap and placed just below the patient's chest. This device communicates via Bluetooth and provides instant speed and heart rate information to the framework. The HxM Zephyr company graciously provided us with a Dev Kit that exposed their Bluetooth protocol and allowed us to link the device to an Android smartphone. The HxM can monitor continuously, but this is extremely battery consuming— development decisions opted for a lighter battery load situation—daily tests.



Figure 15: HxM Device on Charging Stand [16]

See Appendix A for further information on the HxM device, including the entire Dev Kit and Bluetooth communication protocol.

The Fitbit [17] total health communication device is a small clip that attaches to a piece of clothing the user is wearing and connects with a pre-existing app to record caloric intake, steps taken, stairs climbed, distance traveled and sleep metrics. The benefit of using the Fitbit is the gigantic resource of data available and the lack of user responsibility in uploading this information. The FitBit is always recording and syncs whenever it is near a base station. The patient's responsibility for data acquisition is taken out of the equation. The framework polls the FitBit database at set intervals to gather the new information. While data entry for the FitBit is handled through the FitBit corporation's own mobile application, the FitBit Beta API allows the framework to access this data via a set of Java classes and JSON type web connectivity. Considering its many limitations and highly restrictive API, the FitBit provides some very interesting metrics for analysis and is the major factor in the prototype's combination

logic/business rules layer. While the API would be invaluable in most situations, for the prototype architecture—it didn't interface with Android in the desired ways. Instead of utilizing the API, the decision was made to switch to the use of a WebDriver and screen scrape the information from the FitBit website. Using the WebDriver, any health device with a website or software program that displays text on the screen could be incorporated into the framework. Were this a real world deployable framework, using the official FitBit API would be a must. See Appendix B. for more information on the FitBit and the FitBit API, including a code reference of the connection to the OAuth system currently used in the FitBit Beta API.

Through careful design the framework was built to support future devices and the addition of new devices. The only reason the prototype does not support a number of other devices is due to cost and the lack of availability of open source mobile healthcare devices. The final contribution made to the modularity of this system is the Appendix D guide to device addition. This guide will cut down on time that the next developers need to spend learning the framework and analyzing code.

3.3 Application Generator Design

The application generator design layout here reflects a future design and is not implemented to the extent described in the current prototype. Due to resource and device constraints, a working test version of this application generator was implemented and tested in Second Life, a 3D virtual world; this implementation is covered in section 4.3.

The ideal architecture for an application generator most closely resembles that of a “State Logic Display” also known as three-tier system architecture. The Display is the applications and other screens, while our “Business Logic” is the stored procedures or batch processes that sit

between the databases and discerns what devices are compatible with which diseases. Finally, the state is the array of patient and device databases. In deploying this system, whether it is in one hospital or in many; several key issues will have to be addressed. These issues are narrowed down in Software Architecture Foundations, Theory, and Practice. The first issue the book mentions is,

“Different software components may require different hardware configurations for their successful execution...”[18]

With this deployment, we will need to be wary of the user’s computer devices, whether they are mobile or desktop. The next issue is:

“Typical system life spans may stretch over decades and require periodic maintenance...” [18]

The system needs to have an extreme longevity, like most medical software. The final issue we have with deployment is that:

“the...system is likely to evolve overtime, again requiring redeployment. New functionality may be introduced and individual components be redeployed.” [18]

The application generator needs to be architected as to support new types of databases to prevent locking the generator into supporting only a few devices or databases. The application generator should be linked to multiple databases storing tables on patients and the metrics of various home healthcare-monitoring devices. The system could potentially be integrated with a hospital or medical monitoring companies’ inventory system in order to keep track of what packages or devices are readily available. The connectors for this portion of the system would not be complex—linking with an inventory database would not be difficult—interpreting the data would be the complex aspect of this portion of the system. In a networked hospital, patient data could be automatically brought in to the framework. The decisions on the devices could then be

made based on the patient's history. It is necessary to have a streamlined system connected and sharing data at a pace fast enough to be acceptable for the users.

The software for this application generator is a tab-based "pane driven" application and is to be used by the doctor when creating patient plans. The application was designed to provide information in quick glances and hold to design standards common in tablet applications. One of the main sources of inspiration for the "pane driven" interface comes from the Notion Ink Tablet's [19] custom pane based UI. The application generator currently has five screens: New Device Plan, Devices, Patients, Plans and Settings. The GUI should automatically handle incompatible sets of devices and continually update the user as to what devices/device package it is going to recommend.

application generator would need extensive customization during development. No patient use cases exist, as the patient would not have access to the application generator in version zero of this framework—eventually the patient may be able to select devices catered to their condition but currently this should be a task left to a technician or doctor.

3.3.2.1 Doctor Use Cases

<i>Use Case</i>	Add New Device Plan
<i>Description</i>	Addition of a new device plan into the database which stores medical monitoring device plans
<i>Actors</i>	Doctor, Medical Databases, Doctor Application
<i>Assumptions</i>	Doctor must have application up and running Database must be connected in to the application. Doctor must have patient selected in application.
<i>Steps</i>	Doctor opens application (if not already opened) Doctor selects New Device Pane Doctor enters in patient information. Doctor selects devices to add to plan. Doctor presses “Select New Plan” button. Database then filters new information and redistributes updated information to the app and to other healthcare professionals.

<i>Use Case</i>	Add Patient
<i>Description</i>	Addition of a new patient into the monitoring database. Not the addition of a new patient into the hospital main database. Existing disease framework must exist.
<i>Actors</i>	Doctor, Medical Databases, Doctor Application
<i>Assumptions</i>	Doctor must have application up and running

	<p>Database must be connected in to the application.</p> <p>Doctor must not have any other patient selected in application.</p> <p>The patient must exist in the main patient database connected to the application.</p> <p>Patient must have diseases supported by one or more monitoring devices.</p>
Steps	<p>Doctor opens application (if not already running)</p> <p>Doctor selects patient pane.</p> <p>Doctor fills in required information and submits the forms.</p>

Use Case	Add/Remove Device
Description	A simple removal or addition to the device database.
Actors	Doctor, Medical Databases, Doctor Application
Assumptions	<p>Application is open and running.</p> <p>For Removal—at least one device exists in the database</p>
Steps	<p>Doctor opens the application (if not already running)</p> <p>Doctor selects Device Pane</p> <p>Doctor navigates to the extra options menu</p>

Use Case	View Patient Data
Description	Viewing of patient plan information within the application
Actors	Doctor, Medical Databases, Doctor Application
Assumptions	<p>Application is running</p> <p>Doctor is on Patient Pane</p>
Steps	<p>Navigate to the patient pane</p> <p>Enter Patient Information</p> <p>View Patient Data</p>

<i>Use Case</i>	View Devices
<i>Description</i>	Viewing of devices within the application
<i>Actors</i>	Doctor, Medical Databases, Doctor Application
<i>Assumptions</i>	Application is running Database is linked in to Applications Doctor is on device pane
<i>Steps</i>	Doctor must navigate to the device pane Scroll through the list of devices Select a device to view more detailed information.

<i>Use Case</i>	View Patients by Device
<i>Description</i>	Viewing a list of patients by what device(s) they are using in their plans
<i>Actors</i>	Doctor, Medical Databases, Doctor Application
<i>Assumptions</i>	Application is already running Doctor is on Device Pane Database is connected to application
<i>Steps</i>	Navigate to Device Pane Select a Device Open extended options View Attached Patients

<i>Use Case</i>	View Devices by Patient
<i>Description</i>	View of a list of devices by what patient's plans are attached to them.
<i>Actors</i>	Doctor, Medical Databases, Doctor Application
<i>Assumptions</i>	Application is already running Doctor is on Patient Pane

	Database is connected to application
Steps	Navigate to the Device Pane Select a Patient Open extended options View Devices Attached to Patient

Use Case	Deactivate Plan
Description	Deactivate a user's plan if it is no longer needed.
Actors	Doctor, Medical Databases, Doctor Application
Assumptions	Application is open Doctor is on Patient Pane Doctor has Patient Selected Database is connected to application
Steps	Navigate to Patient Pane Select a Particular Patient Deactivate Plan

Use Case	Settings Management
Description	Change settings of the application.
Actors	Doctor, Medical Databases, Doctor Application
Assumptions	Application is open Doctor is on Settings Pane
Steps	Navigate to Settings Pane Adjust Selected Settings Apply Setting Changes

3.3.3 Application Generator Mockup Screens

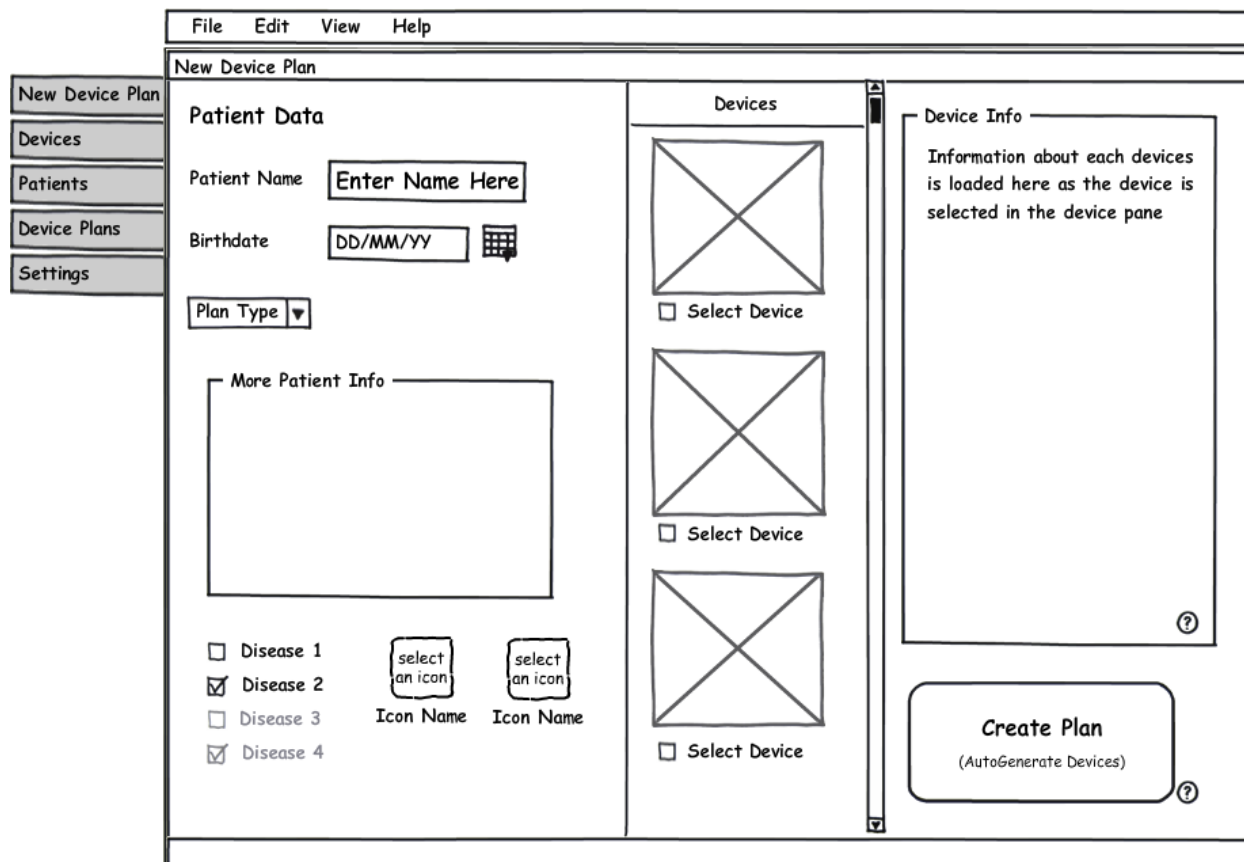


Figure 17: Doctor New Device Plan Screen

The new devices screen is the main hub of the application. The layout is a tri-pane view—patient information, devices and device information are displayed simultaneously. The user can input patient information in the left most pane to allow the program to dynamically fill the Devices pane with plans compatible with the patient’s afflictions.

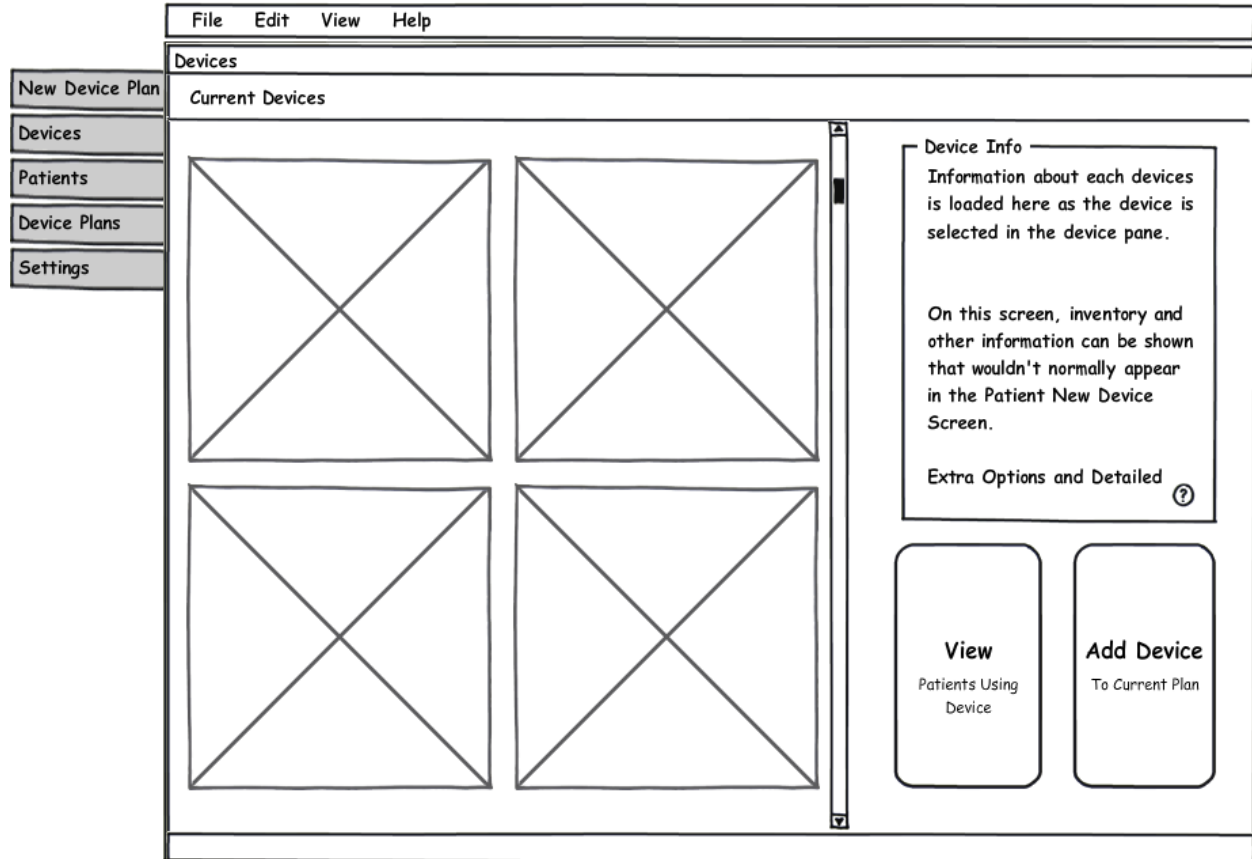


Figure 18: Doctor Devices Screen

The devices screen is where the user can view all current-gen devices available in the database. It focuses on large images of the devices so that the user can easily see what he is looking for. The devices are sorted by manufacturer so that if the user knows the manufacturer but not the device, he/she can still find it. The user can read all relevant information on the device in a textbox in the right pane. The user can also view all patients using the device or add the selected device to the device plan in progress.

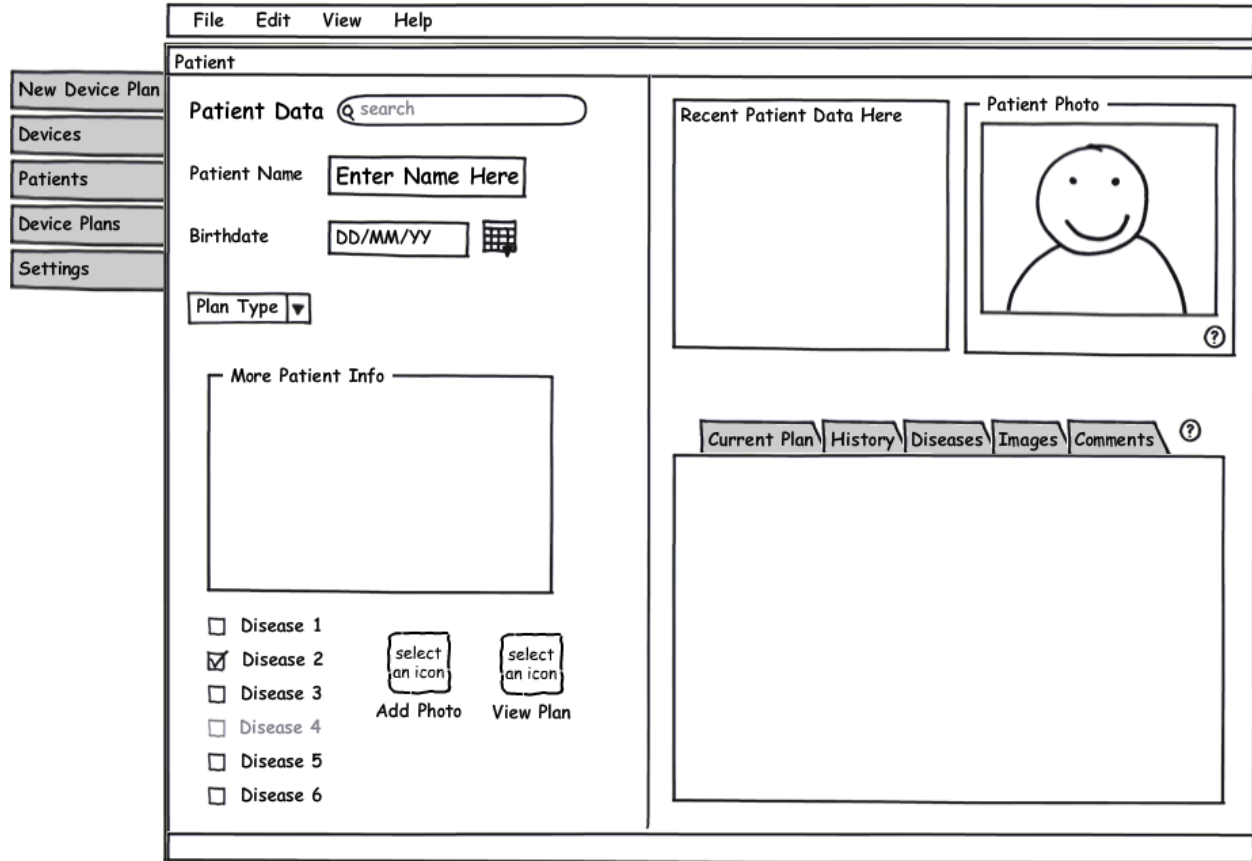


Figure 19: Doctor Patients Screen

The patients screen is a view of all patients registered with the monitoring database. The user can search by name, birth date or plan type. Some detailed information is given, but this is not the record keeping application. It is targeted at providing information about the patient's device plan. The doctor can edit some details here and can provide comments or alter the current plan.

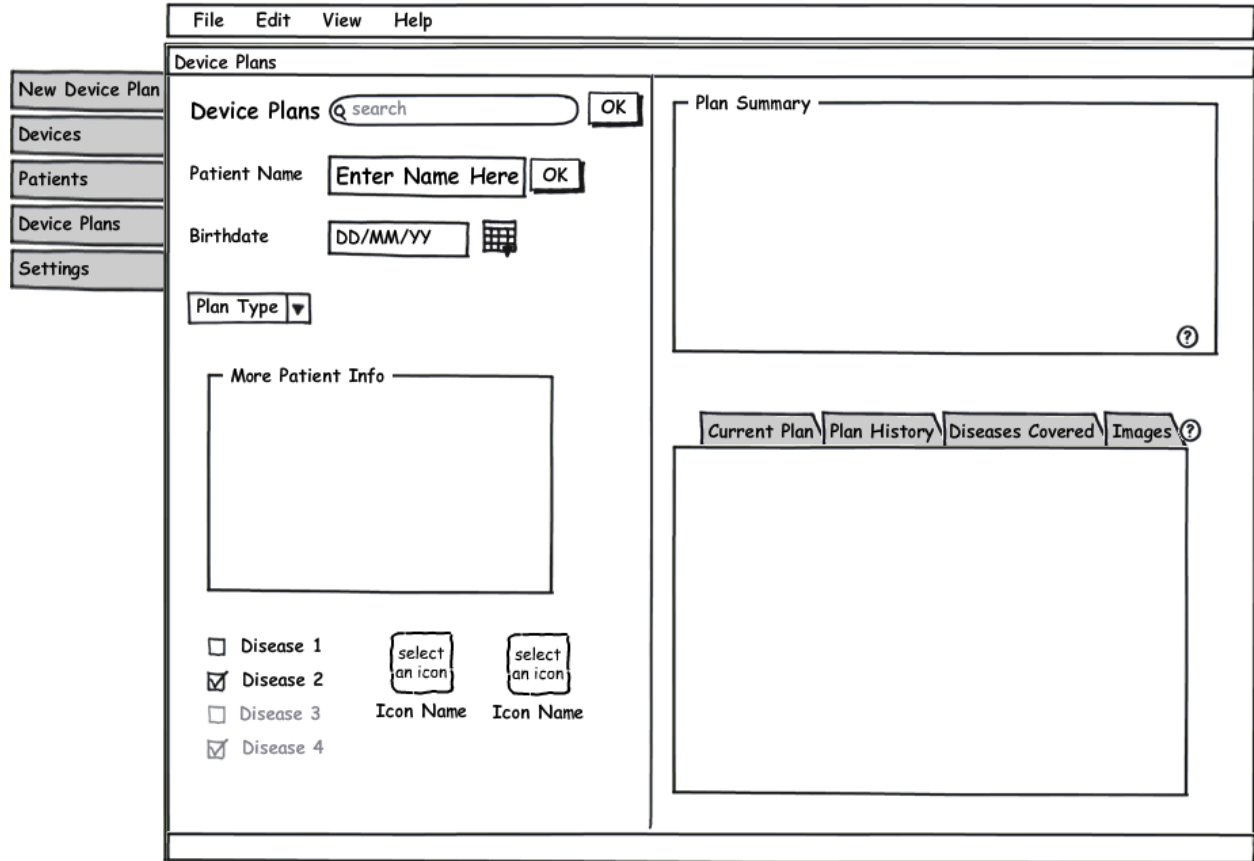


Figure 20: Doctor Device Plans Screen

The doctor can sort and filter through the patient plans in this screen. The screen is useful to track trends with devices or to analyze what to do in case of a recall. Plans can be searched by device name, patient name and many other options.

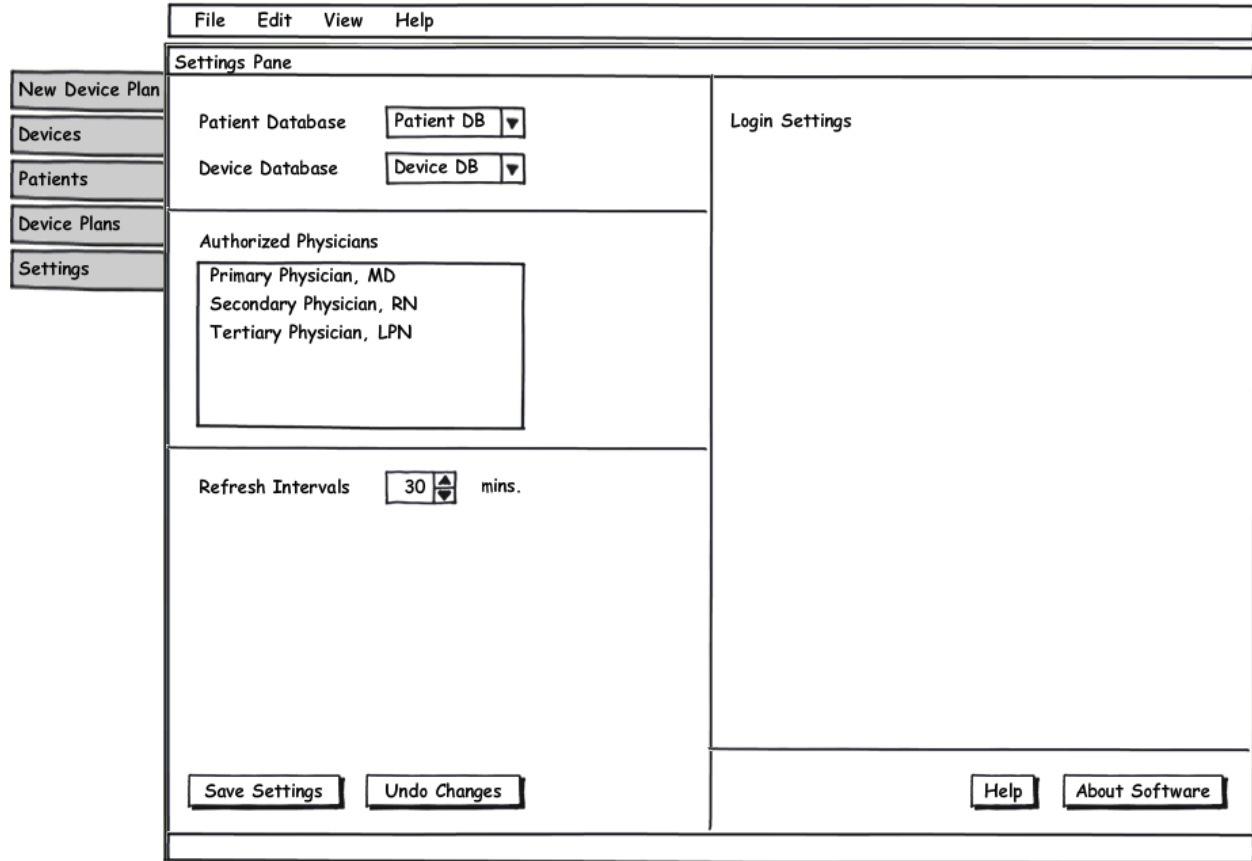


Figure 21: Doctor Settings Screen

The final screen available on the main application is the device settings screen. This is where the user can tune any settings that the device may offer. The connected databases are given here and the allowed users are also filtered in here. The patient can change login options and refresh intervals here as well. Help and About Software options are also present.

4. METHODOLOGY, RESULTS AND ANALYSIS

The prototype developed from this architecture is only a portion of what the future of home healthcare could potentially provide. The system developed provides support for two devices and analyzes data from these two concurrently, providing recommendations to both patient and doctor in a timely manner. The following section describes the results of the findings from the use of this system.

4.1 Methodology

Based on testing and the deployment of a prototype the various components and steps needed to create a modular monitoring framework are better known. These components can be abstracted to create a sensor based framework outside of the healthcare domain.

- **Sensor Layer** – must be robust as to provide a metrics for monitoring. If the number of sensors grows significantly, an application generator will be needed to manage these devices. These devices must “plug into” the framework.
- **Application Layer** – must be developed using current-generation mobile software platforms and designed keeping primary users in mind. There may be more than one application in the framework depending on the domain.
- **Data stream Layer** – a connector that connect sensors to mobile communication devices (or also connects mobile communication devices to community services at a health care provider)
- **Business Logic and Database Layer** – the database is presumed to be a standard relational database management system that can use SQL. Web connectivity is a

must. The business logic layer can be as robust or as small as the framework requires. The database schema for a given monitored symptom and the business rules associated with that symptom are plugins to the framework.

Each component of the framework is necessary. Should alternative, similar frameworks be created, the implications of linked frameworks should be explored in the future.

4.2 Prototype Development and Results of Prototype Testing using Android

A prototype healthcare framework was developed and tested using Android. Because of limitations of devices and development standards, the prototype design took some liberties with the outlined architecture. The system supports two of the devices mentioned earlier, so the creation of an actual integrated application generator was not completed with real devices and was instead implemented in the Virtual World *Second Life* using healthcare data that previously existed. The two devices gather data from the user and then transmit that data into our framework to be analyzed. We have a batch process that aggregates all current data and then analyzes it using a set of rules. The system then provides an email feedback notification instead of integrated notifications through the smartphone operating system. This design decision was made due to limitations of the “push” notification system in the Android operating system. Batch emails were a much more logical choice and perform the same function as the notification system outlined in the architecture but instead conveys relevant information through emails. See Appendix F for a video demonstration of the prototype.

Testing the Android prototype was straightforward. We measured the minimum taps to accomplish necessary actions, looked at interface flow and connectivity speed. The wide range of situations the application could encounter was not systematically tested, but by using common

use cases and monitoring situations we were able to come to a conclusion on the efficiency of the Android application. Interface flow appears reasonable, with minimum taps to accomplish necessary tasks. An improved and professionally designed interface would benefit the framework—as currently we are lacking art assets and a graphic designer.

Connectivity in the prototype needs to be improved. The Bluetooth devices take too much time to connect and there is a lack of feedback on connection problems. Implementing a connection feedback loop would be useful but we want the connectivity to work without user intervention since most users are presumed naive. The devices themselves interact flawlessly with the framework. The sensor-to-database connection functions as intended and the correct data streams are pushed to the database upon each patient's test upload. The information is time stamped and processed correctly. The business logic then feeds information to a batch process running on the machine which can alert the patients and doctors of any problematic health concerns.

The following figures reflect the prototype UI as it was undergoing testing, these are provided to give a decent image of the framework in a real world implementation. They do not reflect a final UI or design.

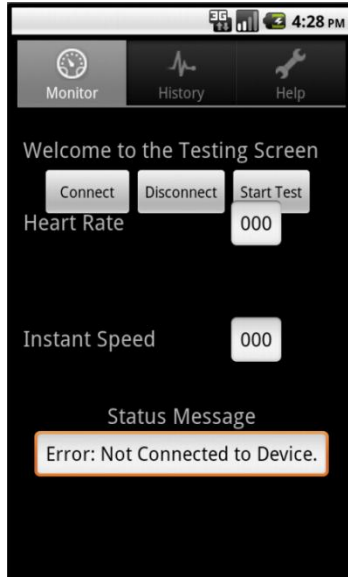


Figure 22: Patient Monitor Screen (Android emulator has issues with the tab layout)

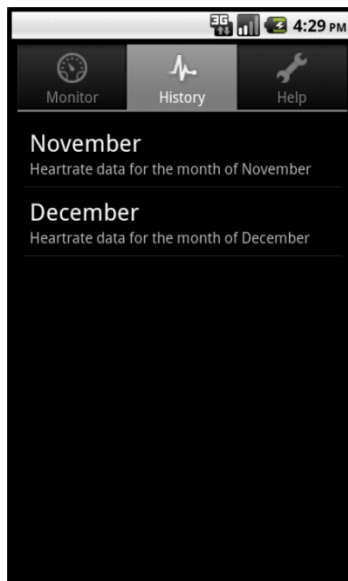


Figure 23: Patient History View

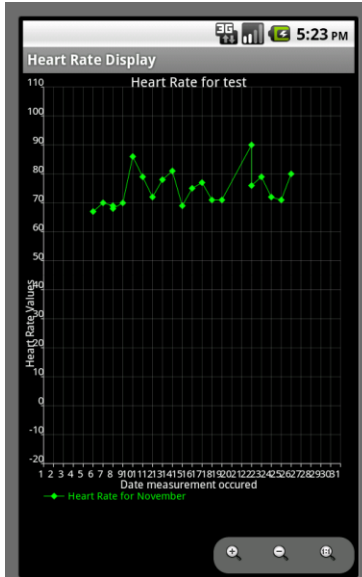


Figure 24: Patient Chart View

The screenshot shows a mobile application interface titled "Patient Settings Screen". At the top, there is a status bar with signal strength, battery, and time (4:29 PM). Below the status bar, there is a navigation bar with three tabs: "Monitor", "History", and "Help". Below the navigation bar, there is a section titled "Monitor Tab:" with the following text: "To monitor your heart rate, navigate to the monitor page and ensure that your device is connected. The IStatus Bari item will show you the current connection status and any warning messages. Once you have ensured your device is connected, press IStart TestI to monitor your resting heart rate. Sit still for the next minute to ensure that the test completes correctly! Once this has completed, the information will be saved and can be viewed in the IHistoryI tab". Below this section, there is a section titled "History Tab:" with the following text: "Select the graph you would like to view by tapping on the list item. This will bring up a graph showing your relative heart rate over".

Figure 25: Patient Settings Screen



Figure 26: Doctor Application Patient List

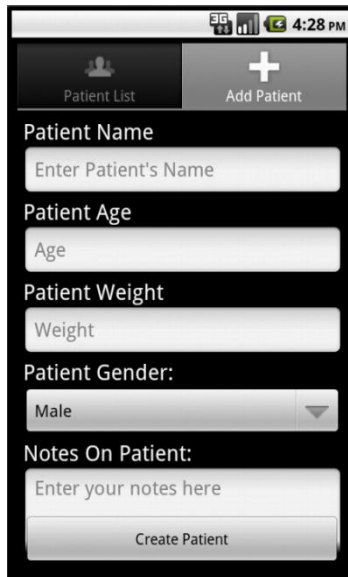


Figure 27: Doctor Application Add Patient Screen

4.3 Prototype Development and Results of Prototype Testing using a 3D Virtual World

Part of the real world testing included the creation of a small scale application generator within the popular virtual world of Second Life. This was done due to the lack of devices in the current prototype and for rapid development purposes. In a virtual world, it is easy to create new devices and to connect them to avatars in a variety of ways. In this prototype, the application generator was linked with previous healthcare objects to test its feasibility. Using the various virtual monitoring devices and connecting them to the systems developed earlier, this application generator functioned much like the real world application would – since the rest of the system cannot tell that it is connected to a real or virtual world. This prototype presents the user with a set of options for diseases, and then, based on choices made, gives the user the set of objects required to monitor the diseases chosen. Our Application Generator simply allows the user to click the object, go through a few dialog trees and then receive the appropriate monitoring items. The Application Generator was built using a framework found on the Second Life Linden Scripting Language Wiki for nested dialogs. Using combinatorial logic a method is created for the correct devices and monitoring devices to be added to the user's inventory when they choose an item to use. The user can then attach these items and use them as intended. The Application Generator should be looked at as proof of concept, but any further work should be done outside of Second Life to provide for a much better foundation and ability for expansion. This small implementation was successful in helping to build a better understanding of the requirements for an application generator.

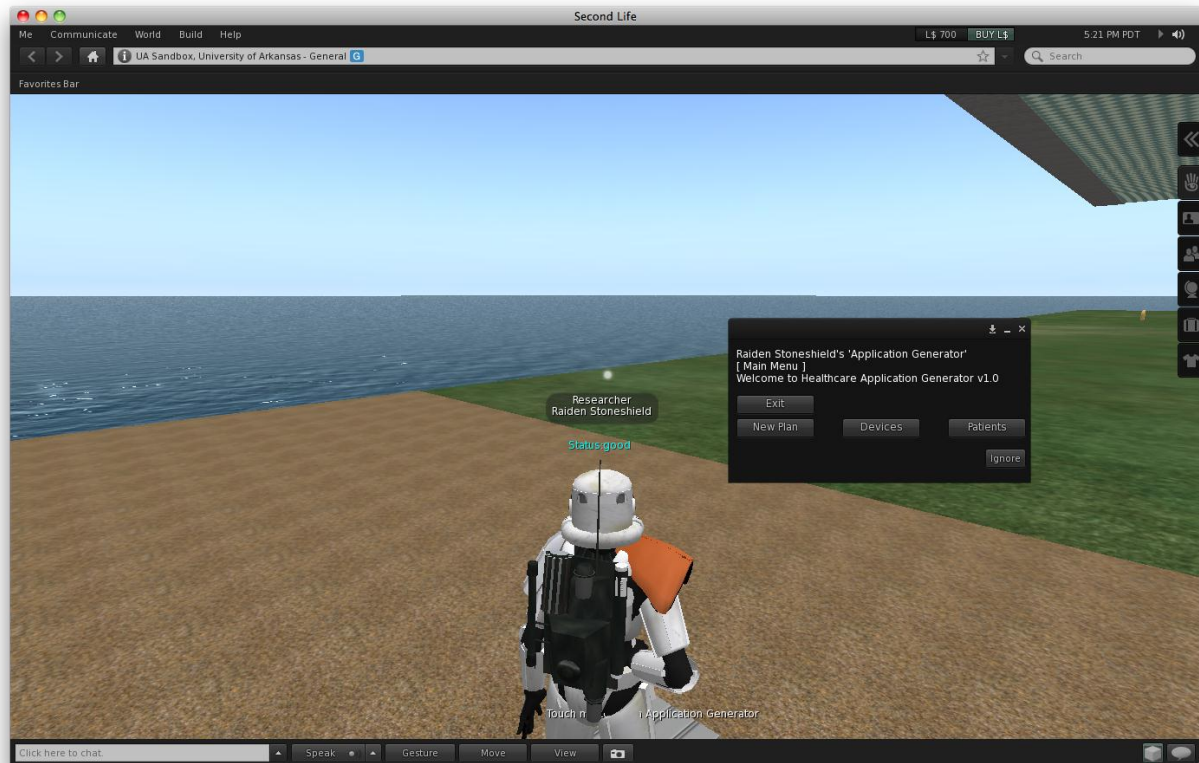


Figure 28: Main Generator Screen

Figure 22 is the main view a user is presented with when they open the Application Generator in Second Life. They can create a plan, view the devices or view the patients. In Figure 23, the user is going to select to create a new plan. Other use cases can be seen if one deploys the source code in Second Life.

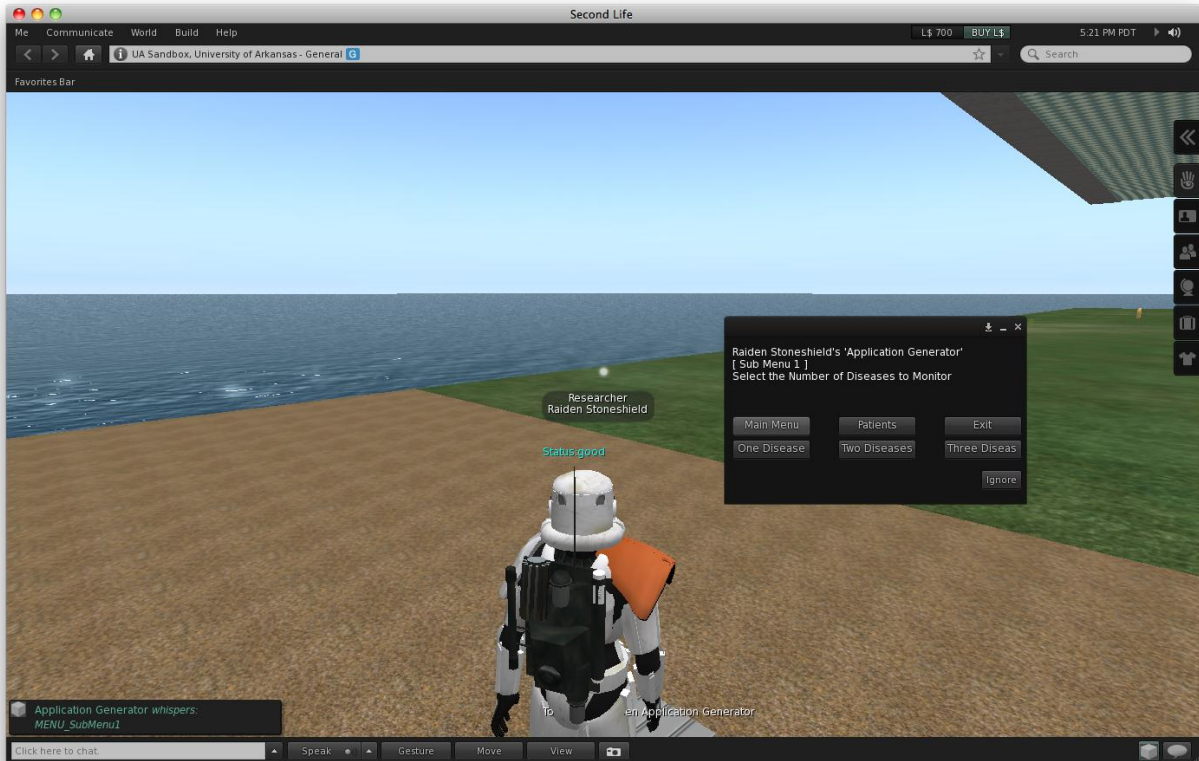


Figure 29: Disease Selection Screen

After selecting to create a new plan, in Figure 24, the user chooses how many diseases to treat. In the current prototype, a maximum of three devices are presented. The lack of drop-down lists also makes this implementation cumbersome.

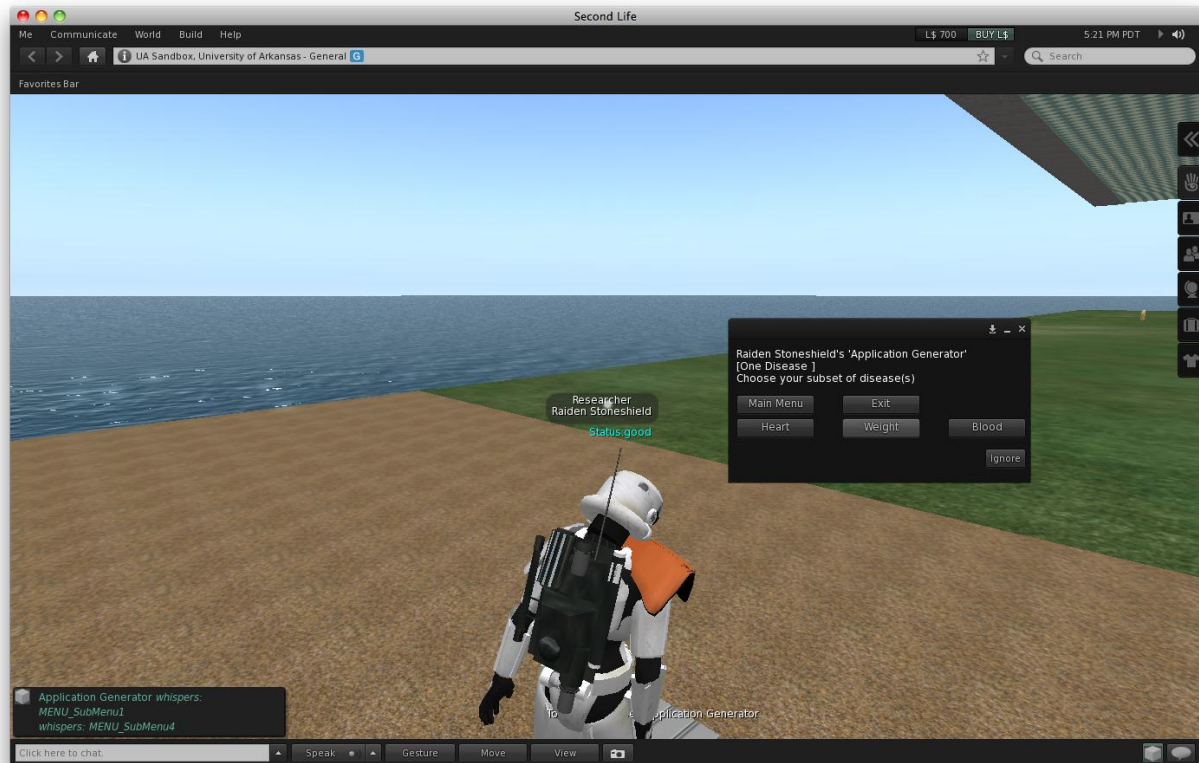


Figure 30: Disease Selection Screen

After the user selects one device, in Figure 25, he chooses which aspect that he wishes to monitor. This user chooses weight and should therefore get the correct “weight monitoring” devices out of the Application Generator object.

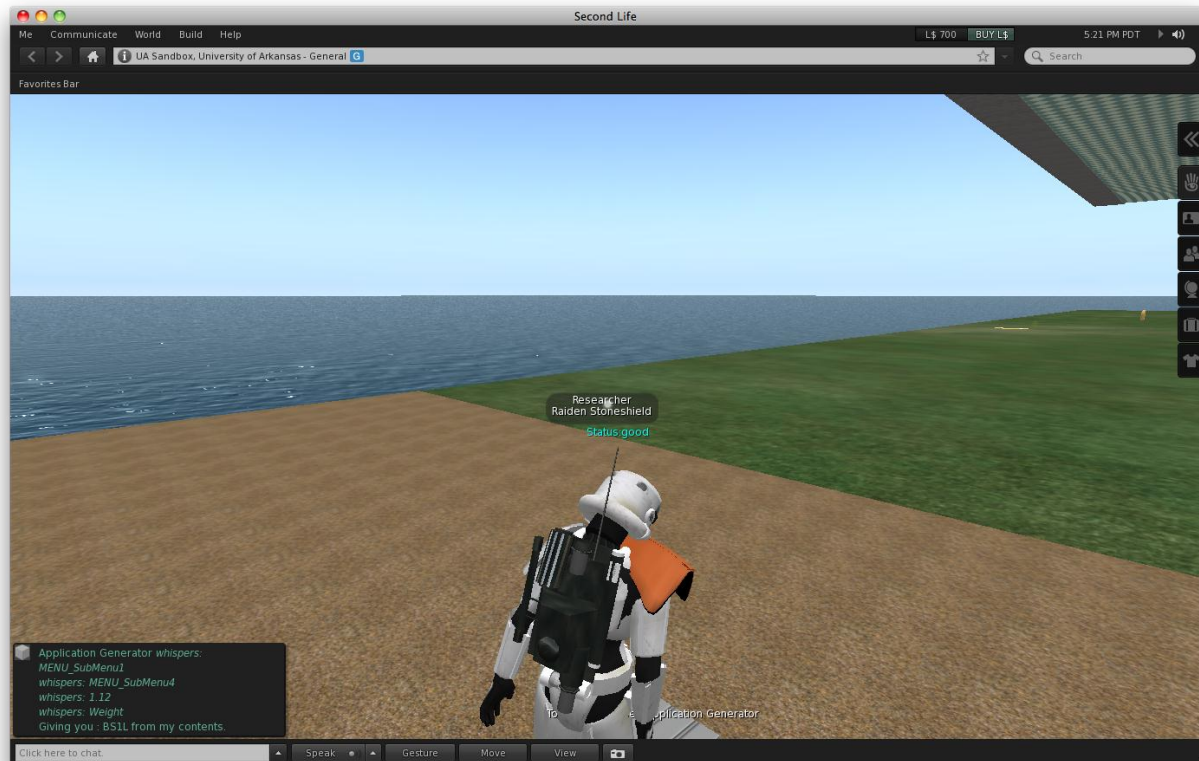


Figure 31: Inventory Addition Screen

The user has selected weight and the corresponding item has been added to his inventory. He can now access this item and use it to monitor his health problems.

4.4 Analysis of the Benefits of a Modular Framework for Home Healthcare Monitoring

When considering the benefits of the implementation of this framework over the current state of mobile healthcare, we've found the following benefits:

- This modular framework would extend current aggregation efforts of health monitoring devices and create a more comprehensive real time view of health for patients.

- This framework would reduce the need for constant hospital visits for patients with chronic illnesses by allowing doctors to constantly monitor progress and visually see health trends.
- This framework would extend medical coverage in developing nations and increase overall health in areas that are currently under served like rural areas.
- The modularity of this framework would ensure its survivability in what could be an extremely competitive market. By being able to adopt new device, the framework would not go out of date.
- The architectural design fosters a growing framework with open interfaces to allow for integration with new sensor based frameworks.

5. CONCLUSIONS

5.1 Summary

This thesis outlines the development of the first modular framework that the author is aware of for home healthcare monitoring based on smartphones and multiple current consumer devices. Started as a spin-off of Dr. Craig Thompson's *Everything is Alive* pervasive computing project, this effort has developed into its own project with many potential sub-projects. The architecture provided in this thesis outlines the potential construction of a healthcare framework that would improve mobile health globally. The prototype provides a proof of concept that can be tested with an initial, limited set of monitoring devices. Some roadblocks do exist for the project's success, but they are not insurmountable.

5.2 Potential Impact

The impact of an extensible, integrated, industrial strength modular home healthcare system would be significant. This framework would change how patients and doctors interact moving from occasional office visits to continuous monitoring and from an emphasis on healing sickness to preserving wellness, for reduced overall cost, with benefit to industrial nations and third world countries alike. A unified and standardized framework and rule set would create an environment of devices and applications that could provide reasonably comprehensive remote healthcare to patients. No longer would invasive and time consuming doctor visits be required and no longer would doctors have to travel as much in developing countries for minor checkups. Rural areas especially in developing countries would especially benefit. This would improve standards of living around the world and ease the workload of healthcare professionals. New

jobs could be created to implement this framework and opportunities for competition among mobile healthcare software and hardware companies would arise.

While a totally unified and standardized framework of this type is currently beyond reach, a simplified network of devices with a general set of standards is not. The largest hurdles to the implementation of this framework will come from the device manufacturers. This issue was encountered even while designing this framework and searching for more devices to add to our prototype. Many manufacturers that were contacted refused to cooperate or provide an API, while questioning why their software was not “good enough.” It is obvious that harmonizing device manufacturers and forcing them to see the greater good that could potentially arise from such a framework will be necessary to get a real world implementation of mobile home healthcare off the ground. Device creators must realize that their application is not the end-all and that someone will always be able to create a better mousetrap. Once their current insular mindset is broken and devices are allowed to communicate and interoperate, the framework will come into its own and be able to push mobile healthcare into a new era. Getting device manufacturers to agree to framework standards is essential because without their support the framework falls apart since nothing works without the sensor networks. A lack of support from the device manufacturing community means that each device will have to be manually tuned and added into the framework, a daunting task that could prove impossible.

Another significant hurdle that this project faces is HIPPA compliance, as mentioned earlier, since the complexities of privacy concerning healthcare information are daunting and even a slight violation of these complex laws can result in a lawsuit. Getting each device and application to comply with government HIPPA standards could be a complex process as not

many existing HIPPA rules or standards apply to software, let alone a modular framework supporting devices from multiple companies. Still, such a framework could itself be separately validated so that only plugin components would require further validation, thereby offloading having to validate a diversity of heterogeneous systems.

These barriers aside, in an era of rapid app market growth, the framework could take off and quickly revolutionize both mobile healthcare and the Internet of Things. Linking this framework with other aspects of technology could create a unique and life changing environment for any human being. Allowing the network to be “social” and fostering health contests or comparisons, even if anonymous, to provide monitoring individuals in comparison to communities. Gamification could change the face of this framework. Gamification, according to “*Gamification.org*,” is the concept of “applying game design thinking to non-game applications to make them more fun and engaging.” [20] The gamification of this framework would keep people interested and motivated in staying healthy. The continued linkage of networking of this platform with other forms of data could also lead to an increased amount of information about human health and humans in general and could potentially provide a wealth of information for sociologists and those interested in dynamic human culture.

Even if a fully developed system is never implemented, the modular framework could be used in specialty hospitals as a multi-device monitoring system for patients with one or several afflictions. This might be the early adopter niche market that advances the framework into wider use. Profit margins in this scenario might be high enough to get the idea off the ground and might encourage device manufacturer to participate.

The potential impact of a fully developed version of this framework would benefit humanity in general and would also be a boon for the Internet of Things community. The ability to monitor health is just one kind of monitoring that might use a similar framework – monitoring your exercise program and diet are similar applications but even monitoring your home electrical use fits into a similar design pattern.

An industrial strength modular home healthcare scenario will require software developers, device manufacturers and healthcare professionals working together to succeed. This will be a daunting task but if accomplished could truly benefit healthcare and change the face of mobile healthcare.

5.3 Future Work

Continued this project and framework should be straightforward and interesting for those interested in sensor networks, gamification, mobile healthcare, mobile devices and development of social networking.

One direction for future work would be social networking integration. Connecting the Facebook API could add an emergency contact number based on relationships.. The integration could be pushed to an “app” that enables a user to compete with others to become healthier. Comparing activity levels within a community would be the next step and could be accomplished using the FitBit API once it is pushed out of beta status. Challenges based on activity levels could be linked across multiple sensors and be application-based or Facebook/Social Network-based. A further step for the doctor applications would be the comparison of multiple patients or entire communities at one time.

Security and HIPPA compliance would be a required direction for this project to reach actual patients though considerable testing could occur with health monitors outside the HIPPA environment. Adding encryption protocols and HIPPA compliance could require a year or two of added development. Following HIPPA rules is necessary for the integration of this framework into public healthcare [15], and security protocols safeguard patient privacy. In the current prototype, logins for doctors and patients are already included, but these can be bypassed if one knows the password or can gain access to the data server side. Being a health industry product, biometric security would be a great direction to take. This could lead to a facial recognition scan (already implemented in the Android 4.0 framework) or in the case of password recovery, a check of a few biometric quantities like heart rate, weight and blood pressure. Another security issue would be ensuring the exclusivity of data in the database and adding new logic to keep this data from being accessed from non-approved sources. Malicious devices could potentially access all of a patient's data and offload that onto their own server—much like the Path Contacts fiasco earlier in 2012 [21]. To prevent this, rules allowing a device to only access certain data would be needed. This would be a straightforward addition to the project and could be the subject of another EiA or Capstone project. Increasing the level of security for this framework only makes the project more attractive to the real world and will be an important stepping-stone for the advancement of modular home healthcare.

Further work on this project could also include a migration to Apple iOS platforms. A difficulty would be encountered in that Apple does not provide for framework apps with third party plugins. A web app could also be useful for those not using smartphones. This web app could borrow GUI elements from the mobile application and appear just like its cousin in the

mobile space, but instead would function inside a desktop browser. Utilizing HTML5 and JavaScript to keep the environments as similar as possible would encourage portability across environments and reduce development costs and fracture from drastically different development environments.

A final direction for this project would be expanding the framework idea into a standard API which would be provided to device manufacturers and third party add-in developers. Instead of requiring the device manufacturers to format their data in a certain way, this involves restructure the sensor-to-database communication layer architecture to fit more of an “API” style architecture. This augments the current architecture but off loads device extensibility to device manufacturers or third party integrators.

Clearly, the work on this project is nearer the beginning than the end. It is my hope that the framework will eventually be widely available in the coming years. Integration with other EiA projects would be of significant benefit, and I believe the architecture laid out here is a good first step and could apply to other sensor network applications.

6. REFERENCES

- [1] C. Thompson, "Everything is Alive," Architectural Perspective Column, *IEEE Internet Computing*, January-February 2004. See: <http://vw.ddns.uark.edu/content/2004-01--PAPER--IEEE-Internet-Computing--Everything-is-Alive.pdf>
- [2] "2008 National Statistics, Outcomes for All Discharges," Agency for Healthcare Research and Quality, HCUPnet, See: <http://hcupnet.ahrq.gov/HCUPnet.jsp>.
- [3] "National Diabetes Fact Sheet: General Information and National Estimates on Diabetes in the United States," Center for Disease Control, 2005. See: http://www.cdc.gov/diabetes/pubs/pdf/ndfs_2011.pdf.
- [4] B. Horowitz, "Intel Telehealth System Helps Heart Failure Patients," June 2010. See: <http://www.eweek.com/c/a/Health-Care-IT/Intel-Telehealth-System-Helps-Heart-Failure-Patients-153973>.
- [5] J. Singh, "Market For Mobile Healthcare Applications Will Grow To US \$1.3 billion in 2012," PalmOS/WebOS Blog, January 2012. See: <http://tamspalm.tamoggemon.com/2012/01/30/market-for-mobile-healthcare-applications-will-grow-to-us-1-3-billion-in-2012>.
- [6] J. Ko, C. Lu, M. Srivastava, J. Stankovic, "Wireless Sensor Networks for Healthcare," *IEEE*, 11/2010. See: http://www.vs.inf.ethz.ch/edu/HS2011/CPS/papers/ko10_wsn-for-healthcare.pdf
- [7] R. Empson, "mHealth: Remote Patient Monitoring Is On The Rise, With Smartphones Leading The Way," *TechCrunch*. 8 Feb 2012: See: <http://techcrunch.com/2012/02/08/mhealth-remote-patient-monitoring-is-on-the-rise-with-smartphones-leading-the-way/>
- [8] J. Li. "Understanding the Acceptance of Mobile Devices for Wireless Health Care," Institute of Healthcare Information Management, National Chung Cheng University, 2008, See: <http://www.iis.sinica.edu.tw/~louisjyli/doc/MasterThesis.pdf> pp6,20-24.
- [9] C. Otto, A Milenkovic, C. Sanders, E. Jovanov, "System Architecture of a Wireless Body Area Sensor Network for Ubiquitous Health Monitoring," *Journal of Mobile Multimedia*, 2006. See http://www.eng.uah.edu/~jovanov/papers/coamej_jmm06.pdf.
- [10] C. Thompson, F. Hagstrom, "Modeling Healthcare in a Virtual World," Architectural Perspectives column, *IEEE Internet Computing*, September-October 2008. See: <http://vw.ddns.uark.edu/content/2008-10--IEEE-Internet-Computing-->

Modeling%20Healthcare%20in%20a%20Virtual%20World--
Craig%20Thompson%20DRAFT.pdf.

[11] “JAG - Java Application Generator,” *JAG*. N.p., n.d. See: <http://jag.sourceforge.net>.

[12] “Google App Inventor,” *About - App Inventor for Android*. Google, 2/17/2011. See: <http://www.appinventor.mit.edu>.

[13] R. Gabriel. “Aspect-Oriented Software Development: Definitions of Modularity,” *ASOD.net*, N.p., 2011. See: <http://aosd.net/2011/files/PerspectivesOnModularity/ModularityDefinitions.pdf>.

[14] J. Rellermeyer. “Modularity as a Systems Design Principle,” *Eth Zurich*, 2011. See: <http://systems.ethz.pubzone.org/servlet/Attachment?attachmentId=555&versionId=1635923>

[15] “Understanding Health Information Privacy,” U.S. Department of Health & Human Services. See: <http://www.hhs.gov/ocr/privacy/hipaa/understanding/index.html>.

[16] “Zephyr HxM BT,” Zephyr Technology. 01/2010. See: <http://www.zephyr-technology.com/wp-content/uploads/2010/11/ZephyrHXM.pdf>.

[17] “FitBit Wireless Activity Tracker,” FitBit, 2012. See: <https://www.fitbit.com/product>.

[18] R. Taylor, N. Medvidovi, E. Dashofy. Software Architecture Foundations, Theory, and Practice, John Wiley & Sons, Inc., 2010.

[19] “Notion Ink,” Notion Ink Adam: Features. Notion Ink Design Labs Inc. See: <http://www.notionink.com/legal/User%20Manual.pdf>.

[20] “*What is Gamification?*” Definition, Gamification.org. Last Updated: 2/07/2012. See: <http://gamification.org/wiki/Gamification>.

[21] R. Emerson, “Path App Contacts Privacy,” Huffington Post, published 2/08/2012. See: http://www.huffingtonpost.com/2012/02/08/path-app-contacts-address-book-privacy_n_1262390.html.

APPENDIX A. HXM DEV KIT

Zephyr's HxM Dev Kit provides a base for analyzing the hardware choices of device manufacturers. Zephyr graciously opened their source to our development which accelerated the development of the prototype. See link to zip download of the HxM Dev Kit:

<https://www.box.com/shared/c169gssedk2nrgu4t41f>. The HxM Dev Kit provides sample code for Android Connections and provides a full API for the Android Operating System. See the *Android API Guide* (.pdf) in the .zip file for more information.

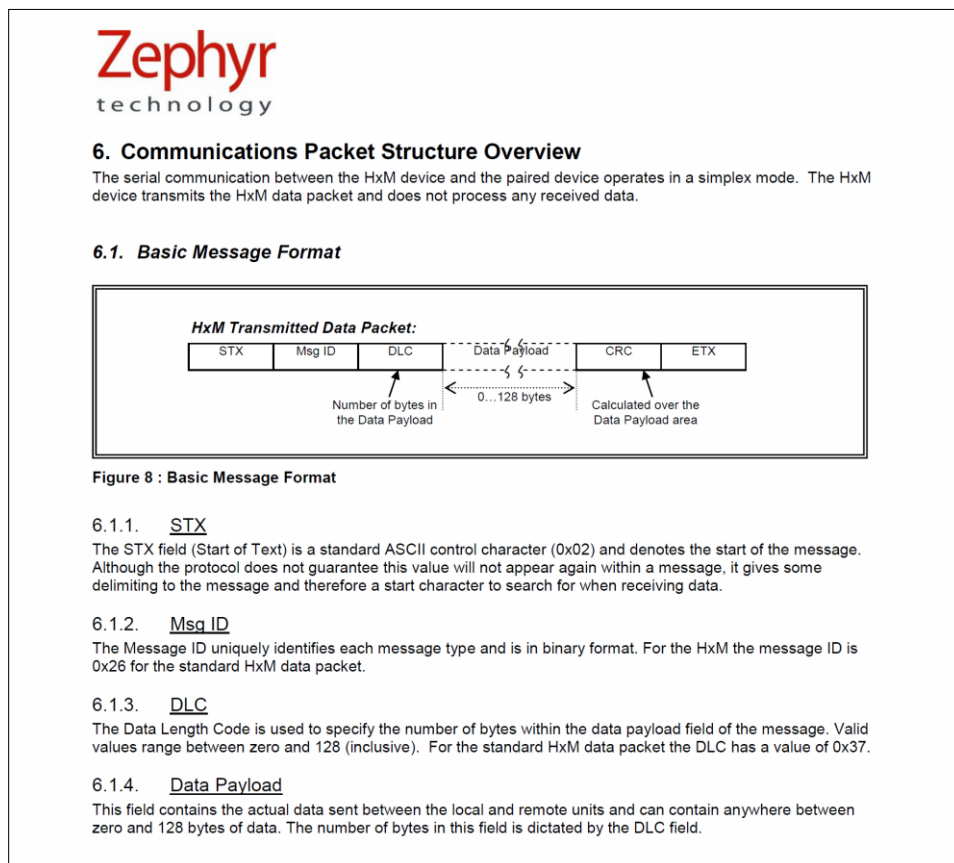


Figure 32: Zephyr Bluetooth Packet Structure

APPENDIX B. FITBIT API INFORMATION

Link to FitBit Developer Website: <http://dev.fitbit.com>

Link to FitBit Java Client Walkthrough: <https://wiki.fitbit.com/display/API/API+Java+Client>

Fitbit API Authentication Process Code Example

```
private FitbitAPIEntityCache entityCache = new FitbitApiEntityCacheMapImpl();

private FitbitApiCredentialsCache credentialsCache = new
FitbitApiCredentialsCacheMapImpl();

private FitbitApiSubscriptionStorage subscriptionStore = new
FitbitApiSubscriptionStorageInMemoryImpl();

public void init(ServletConfig config) throws ServletException {
    super.init(config);
    try {
        Properties properties = new Properties();
        properties.load(getClass().getClassLoader().getResourceAsStream("conf
ig.properties"));
        apiBaseUrl = properties.getProperty("apiBaseUrl");
        fitbitSiteBaseUrl = properties.getProperty("fitbitSiteBaseUrl");
        exampleBaseUrl = properties.getProperty("exampleBaseUrl");
        clientConsumerKey = properties.getProperty("clientConsumerKey");
        clientSecret = properties.getProperty("clientSecret");
    } catch (IOException e) {
        throw new ServletException("Exception during loading properties", e);
    }
}

FitbitAPIClientService<FitbitApiClientAgent> apiClientService = new
FitbitAPIClientService<FitbitApiClientAgent>(
    new FitbitApiClientAgent(apiBaseUrl, fitbitSiteBaseUrl,
credentialsCache),
    clientConsumerKey,
    clientSecret,
    credentialsCache,
    entityCache,
    subscriptionStore
);
```

APPENDIX C. BUSINESS RULES

These are a sampling of the type of rules (alert flags) included in the business logic processes (batch procedures) of our Apache MySQL Database:

Heart Monitor

HR above certain range

HR below certain range

HR increasing regularly

HR decreasing regularly

HR up weekly

HR down weekly

HR major up weekly

HR major down weekly

NO HR reading for week

NO HR reading for day

FitBit

Weight UP

Weight DOWN

Water LOW

Water HIGH

Fitness LOW

Activity DOWN

Activity UP

Steps UP

Steps DOWN

Calorie UP

Calorie DOWN

Distance above certain range

Distance below certain range

Heart Monitor and FitBit

Weight UP and HR up

Weight UP and HR down

Weight UP and NO HR

Activity DOWN and HR UP

Activity UP and HR DOWN

Sleep DOWN and HR UP

Steps UP and HR DOWN

Steps DOWN and HR UP

Calorie UP and HR UP

Calorie DOWN and HR DOWN

Sample Java Code from Batch Process:

Example of portion of alert system alert email subject composition. This is the initial run of the alert system and sets flags for the body portion of the message. It creates an easy to read subject for the email/alert and allows the system to keep track of what issues the patient is having.

```
public String AnalyzeData(List<UserDTO> userData) throws
AddressException,
    MessagingException {

    String messageBody = "";
    // String emailaddress = "";
    String subject = "Healthcare Alerts: ";
    int alertnum = 0;

    if (heartRateHighLow(userData) == "Yes") {
        messageBody.concat(" High Heart Rate,");
        alertnum++;}
    if (heartRateHighLow(userData) == "No") {
        messageBody.concat(" Low Heart Rate,");
        alertnum++;}
    if (heartRateIncreaseDecrease(userData) == "Yes") {
        messageBody.concat(" Increasing Heart Rate,");
        alertnum++;}
    if (heartRateIncreaseDecrease(userData) == "No") {
        messageBody.concat(" Decreasing Heart Rate,");
        alertnum++;}
    if (heartRateUpDownWeekly(userData) == "Yes") {
        messageBody.concat("Heart Rate Up Weekly,");
        alertnum++;}
    if (heartRateUpDownWeekly(userData) == "No") {
        messageBody.concat(" Heart Rate Down Weekly,");
        alertnum++;}
    if (noHeartRateReadingWeekly(userData) == "Yes") {
        messageBody.concat(" No Weekly Reading,");
        alertnum++;}
    if (fitbitWeightUpDown(userData) == "Yes") {
        messageBody.concat(" Weight Up");
        alertnum++;}
    if (fitbitWeightUpDown(userData) == "No") {
        messageBody.concat(" Weight Down");
        alertnum++;}...
```

...and so on for the rest of the list of rules composing the subject.

Sample Logic Code for High or Low Heart Rate

Note the HighHRoccurrences variable; this is modified on a patient to patient basis. Only when this number is increasing or approaches a set limit, is this flag for a high or low heart rate set.

```
public String heartRateHighLow(List<UserDTO> hrdata) {  
  
    // hr data sort  
    int highHRoccurrences = 0;  
  
    //get occurrences of high heart rate necessary for patient  
    allowedoccurrences = getHighHRoccurrences(hrdata);  
  
    List<UserDTO> currentWeekData = getCurrentWeekData(hrdata);  
  
    for (int i = 0; i < currentWeekData.size(); i++) {  
  
        if (currentWeekData.get(i).getHeartrate() > HIGHHR) {  
            highHRoccurrences++;  
  
        }  
    }  
    if (highHRoccurrences > allowedoccurrences) {  
        return "Yes";  
    } else if (highHRoccurrences > 0)  
        return "Somewhat: " + highHRoccurrences;  
    else  
        return "No";  
}
```

These “Yes” or “No” statements are further processed (compared with other fitbit data) and then passed along to the email composer. When “Somewhat” is returned, this is noted by the system outside of the subject composition and can be further analyzed (alert decisions). The system has variable levels of “truth” that allow a finer grain of analysis than a high-low system provides.

APPENDIX D. PROTOTYPE VIDEO

A video demonstration is included in the thesis defense and is available on the web at

<http://sdc.csce.uark.edu/projects/modhealth/>

