2007

WARM: Wearable Assistant with Remote Monitoring

Lisa Michelle Provenzano Heugel

University of South Florida

Follow this and additional works at: http://scholarcommons.usf.edu/etd
Part of the American Studies Commons

Scholar Commons Citation

This Thesis is brought to you for free and open access by the Graduate School at Scholar Commons. It has been accepted for inclusion in Graduate Theses and Dissertations by an authorized administrator of Scholar Commons. For more information, please contact scholarcommons@usf.edu.
WARM: Wearable Assistant with Remote Monitoring

by

Lisa Michelle Provenzano Heugel

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Computer Science
Department of Computer Science and Engineering
College of Engineering
University of South Florida

Major Professor: Nagarajan Ranganathan, Ph.D.
Dewey Rundus, Ph.D.
Stuart Silverman, Ph.D.

Date of Approval:
March 29, 2007

Keywords: Mobile, Elderly, Reminder, Medical, Software

© Copyright 2007, Lisa Michelle Provenzano Heugel
DEDICATION

For my husband, Ben
ACKNOWLEDGEMENTS

I would like to thank Dr. Ranganathan for his guidance and understanding of my hectic work schedule. I would also like to thank Drs. Rundus and Silverman for agreeing to be on my review committee and to give me comments to improve this thesis.

I would also like to thank my family for having faith in me. Finally, I want to thank my husband for putting up with me and for motivating me when I was ready to throw in the towel.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF TABLES</td>
<td>iii</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>iv</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>v</td>
</tr>
<tr>
<td>CHAPTER ONE  INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Definition of Wearable Computers</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Traits of Wearable Computers</td>
<td>2</td>
</tr>
<tr>
<td>1.2.1 Functionality</td>
<td>2</td>
</tr>
<tr>
<td>1.2.2 Hardware</td>
<td>2</td>
</tr>
<tr>
<td>1.2.3 User Interface</td>
<td>3</td>
</tr>
<tr>
<td>1.3 Motivation for this Work</td>
<td>3</td>
</tr>
<tr>
<td>1.4 Organization of Thesis</td>
<td>4</td>
</tr>
<tr>
<td>CHAPTER TWO WEARABLE COMPUTING RESEARCH</td>
<td>5</td>
</tr>
<tr>
<td>2.1 History of Wearable Computers</td>
<td>5</td>
</tr>
<tr>
<td>2.2 Research</td>
<td>6</td>
</tr>
<tr>
<td>2.2.1 Carnegie Melon University</td>
<td>7</td>
</tr>
<tr>
<td>2.2.2 Massachusetts Institute of Technology</td>
<td>8</td>
</tr>
<tr>
<td>2.2.3 Georgia Institute of Technology</td>
<td>10</td>
</tr>
<tr>
<td>2.2.4 Other Research</td>
<td>12</td>
</tr>
<tr>
<td>CHAPTER THREE CHALLENGES</td>
<td>15</td>
</tr>
<tr>
<td>3.1 Power</td>
<td>15</td>
</tr>
<tr>
<td>3.2 Privacy and Security</td>
<td>17</td>
</tr>
<tr>
<td>3.3 Network Connectivity</td>
<td>19</td>
</tr>
<tr>
<td>3.3.1 Wearable to Fixed Location</td>
<td>20</td>
</tr>
<tr>
<td>3.3.2 Inter-device Communications</td>
<td>21</td>
</tr>
<tr>
<td>3.3.3 Wearable and Stationary Object</td>
<td>22</td>
</tr>
<tr>
<td>3.4 User Interface and Unobtrusiveness</td>
<td>23</td>
</tr>
<tr>
<td>3.4.1 Human Axis</td>
<td>23</td>
</tr>
<tr>
<td>3.4.2 Computer Axis</td>
<td>24</td>
</tr>
<tr>
<td>3.4.3 Application Axis</td>
<td>25</td>
</tr>
<tr>
<td>CHAPTER FOUR WARM OVERVIEW</td>
<td>27</td>
</tr>
<tr>
<td>4.1 Users</td>
<td>27</td>
</tr>
<tr>
<td>4.2 Features and Functions</td>
<td>28</td>
</tr>
</tbody>
</table>
LIST OF TABLES

Table 2.1 Three Generations of Wearable Computers 8
Table 3.1 Protecting a User’s Privacy 19
Table 6.1 Summary of Functional Scenarios 53
Table 6.2 Data from WARMClient..reminderSpec 55
Table 6.3 Data from WARMClient..reminderSchedule 56
Table 6.4 Sample Entries from reminderInProgress and sentToWearable 57
Table 6.5 Pseudocode for Synchronization 58
Table 6.6 Data from WARMActivity..reminderLog During Simulation 59
LIST OF FIGURES

Figure 5.1 WARM Architecture 36
Figure 5.2 WARMClient Relational Database Diagram 38
Figure 5.3 WARM Web Application Login Page 40
Figure 5.4 WARM Web Application User Options Page 41
Figure 5.5 WARM Web Application Contact Page 42
Figure 5.6 WARM Web Application General Reminder Setup 44
Figure 5.7 WARM Web Application Medical Reminder Setup 46
Figure 5.8 WARM Wearable General Reminder Screen 48
Figure 5.9 WARM Wearable Medical Reminder Screen 48
Figure 6.1 WARM Process Flow 52
WARM: WEARABLE ASSISTANT WITH REMOTE MONITORING

Lisa Michelle Provenzano Heugel

ABSTRACT

Wearable computing has been in existence as part of our attire for many years, a few examples being the watches, cell phones and MP3 players. Recently, many researchers have been devising ways to incorporate wearable devices seamlessly into our lives with the goal of making them an indispensable part of our daily routines. In this thesis, a new wearable computing device called WARM (Wearable Assistant for Remote Monitoring) for use by the elderly and the handicapped is proposed and its software architecture is defined and described. The physically challenged or the elderly, living on their own, but who require assistance with daily tasks, could retain some of their independence with the use of device like WARM.

The software architecture of this proposed device is being designed so that the level and types of assistance necessary will be customized based on the user’s needs and their family members can remotely monitor them and be alerted to any issues. In order to maintain extensibility and scalability, a table driven approach is used. The information for reminders is entered through the web-based front end portion of the application which is then written out to relational database tables using stored procedures. The synchronization program, which runs on the user’s PC, also uses a series of stored procedure calls to determine while a reminder is to be sent, when contacts should be
alerted and finally moves the cleared reminders to the log table for future reporting. The architecture of the proposed WARM device, its salient features, the software interface, the simulation set up and the results are presented.
CHAPTER ONE
INTRODUCTION

Many researchers of this field discuss the paradigm shift that comes along with wearable computers. “Wearable computers deal in information rather than programs, becoming tools in the user’s environment much like a pencil or a reference book [48].”

A laptop computer can provide portable access to information, but the wearable takes this leaps (not just steps) farther by allowing information to be automatically accumulated and modified according to the user’s environment. A desktop computer with a calendaring system can alert a user of an upcoming meeting, but once again the wearable expands on this concept by unobtrusively reminding the user in a format specific to his current environment.

The conference proceedings, technical journal and magazine articles on wearable computers are speckled with buzzwords like “pervasive”, “ubiquitous” and “seamless”. All of these adjectives are proper terms to describe this emerging technology. This field has the potential to vastly change the role that computers play in our lives.

1.1 Definition of Wearable Computers

In its most simplistic definition, a wearable computer is a computer that is attached to the user, usually as part of clothing or an accessory. Researchers add to this by listing functionality they offer and characteristics that wearable computers should posses. They

1
say these computers should be embedded into everyday life and have enough intelligence for the user to interact with them as easily as they do with their desktop personal computer [7]. According to Gorlick, “one goal of wearable computing is to create digital devices that are as easy to don and as comfortable to wear as common articles of clothing such as blouses, pants and belts [15].” These computers should be an extension of the users and seamlessly enhance their abilities to do their everyday tasks [6].

1.2 Traits of Wearable Computers

1.2.1 Functionality

From a functionality perspective, all wearable systems have some traits that set them apart from the standard personal computer. A wearable computer must have the ability to be cognizant of its own state, the user’s state and the current external environment. This affords it the ability to “act” accordingly. For example, if the user is in a meeting, the wearable should communicate in such a way as to not disrupt. But, when the user is in the car, the communication should be audible to not distract from driving. Along these same lines, the privacy of the user preserved when appropriate using methods only audible or visible to the user [53].

Since the wearable is meant to be an extension of its user, it needs to provide constant access to the information and services it offers. Therefore, a high degree of connectivity at a high speed and an intuitive user interface are key [6].

1.2.2 Hardware

The hardware traits of wearable computers fall right in line with the functional traits. Most of the time, a user will be mobile. Because of this, there will be a lot of information transfers of varying amounts that need to happen as a very quick speed. This
consumes far more power than standard mobile devices [6]. The wearable computers need a way to power themselves for extended periods without performance degradation or the need for constant recharging. If the users cannot readily access the tools necessary to perform their everyday tasks, the wearable becomes a hindrance for them as opposed to a tool.

1.2.3 User Interface

Not only should a wearable be unobtrusive, but it should not cause the user to change their routine in any way. This characteristic most likely will cause there to be a shift in thought from the way desktop and laptop computers are designed. The user input can’t simply be a keyboard. This would definitely cause the user to modify their routine thus violating the rule that wearables must blend seamlessly with their user’s routine [49]. For example, “A display in the glasses and a motion sensor on the wrist” could be used to tell the wearable information about the user’s current external environment [6].

1.3 Motivation for this Work

Computers are such a large part of our lives, no one can even imagine living without them. Despite how much computers have become ingrained into our lives, wearable computers are still in their infancy stage. Research and design on wearables is growing and there is potential to truly make a difference in people’s lives.

The Baby Boomers are our largest population group and they are all approaching a point in their lives where they may need a helping hand. A wearable device designed with their needs in mind could really help them keep their independence and quality of life. The device proposed in this work will allow them to determine their own needs and create their own tool that does just what they require.
1.4 Organization of Thesis

The remaining chapters will give background on wearable computers and describe the proposed wearable in detail. Chapters 2 and 3 go over the history of wearable computing and summarize some of the research that has been done. Chapter 3 outlines the challenges developers of wearables face. In Chapter 4, an overview of the WARM device will be given and the design specifics will be discussed in Chapter 5. Chapter 6 will describe a simulation of the functionality and performance of WARM. Finally, Chapter 7 will go over future proposed work.
CHAPTER TWO
WEARABLE COMPUTING RESEARCH

“The fabric is the computer [37]” This statement refers to the relationship that textiles and computers have shared for centuries. Interestingly enough, the first binary information system actually dealt with fabric. Around 1801, Joseph Marie Jacquard invented the Jacquard head [37]. Jacquard is a fabric, still popular today, whose pattern is made by thread woven either on the front or the back of fabric. Pattern cards were made by punching a hole where the thread was on the front (in the on position) and no-hole where the thread was to be in the back (in the off position). A loom read the card and created the pattern accordingly.

2.1 History of Wearable Computers

The concept of wearable computers is not a new one. In fact, the idea of attaching tools to one’s body to assist in everyday tasks has been around for close to 100 years. According to Rhodes’ “A Brief History of Wearable Computing”, in 1907, aviator Alberto Santos-Dumont commissioned Louis Cartier to make him a timepiece with a wristband. He needed this to keep his hands free for piloting [3].

Research on wearable computing has been going on for decades at universities and labs around the world. The first documented wearable computer was invented by Ed Thorp and Claude Shannon of Massachusetts Institute of Technology. It was a 2-part
system for predicting a roulette wheel. The data-collector would use buttons on an analog computer to indicate the speed of a roulette wheel. The computer would then send tones to a hearing aid in the bettor’s ear [3]. In this first wearable system, privacy and unobtrusiveness were key because of the consequences for getting caught with an electronic device at a gaming table [54]. “Worried about getting caught, they delayed revealing their system until 1969, when they described it in a statistics journal. [54]”

Since then, a steady stream of research and prototypes has been presented. In the beginning, many of them had something to do with gambling probably because these tasks are easy to quantify and a computer’s ability with predictive computations is already known. In the past 15 – 20 years, the trend has shifted and it seems that a lot of research funding is going towards wearable computers to assist elderly or disabled people and smart cards for access to secure areas.

2.2 Research

In 1997, Carnegie Mellon University, Massachusetts Institute of Technology and Georgia Institute of Technology hosted the first IEEE International Symposium on Wearable Computers for an audience of 382 scholars and researchers [3]. These three universities have paved the way for research and prototypes on wearable computing.

It is interesting to note that most of today’s more prominent members of the wearable research community have been involved since the beginning and have grown right along with it. They seem to have had affiliations with one or more of these three universities at some point during their academic career. They contribute to each other’s research and build upon ideas developed by colleagues. This is truly a community of researchers working together to explore this emerging field.
2.2.1 Carnegie Melon University

At CMU, there is a collaborative group with members from three departments whose goal is to study the user interface and architectural design of a wearable system [4]. Dozens of prototypes and actual systems have come out of this group, quite of few of which are from students in a rapid prototyping course which is offered every semester.

In 1991, the students in a summer design course released VuMan. VuMan is a wearable computer that allows users to “walk through” a blueprint using a 3-button device, similar to a mouse, for input and a heads up display for output. This first attempt was quite rudimentary consisting of an 8 MHz 80188 processor, .5 MB of ROM and a commercially available display. A few months later, the original VuMan design was improved upon when VuMan 2 was released by another semester of the design course. The second generation was similar to the first, but VuMan 2 allowed for more than one application and came with one for going through an instruction manual for an alternator. VuMan 3 had even more hardware than the previous two [4].

Another wearable application, Navigator, also came to be as a result of the system design class at CMU. It’s a navigation tool like the VuMan series, but Navigator take speech as input, allowing the users hand-free operation. Also, Navigator is a whole computer and not just an embedded system like the VuMen. This allows for application development in a Unix environment [28]. Navigator’s architecture is modular so its hardware can be reconfigured according to the software.

Table 2.1 [28] shows a summary of the 3 generations of wearable computers developed at CMU in their student design courses.
Table 2.1 Three Generations of Wearable Computers [28]

<table>
<thead>
<tr>
<th></th>
<th>VuMan 1</th>
<th>VuMan 2</th>
<th>Navigator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delivery Date</td>
<td>Aug-91</td>
<td>Jan-93</td>
<td>Jun-93</td>
</tr>
<tr>
<td>Proc/Speed</td>
<td>80188/8MHz</td>
<td>80188/13MHz</td>
<td>80388/25MHz</td>
</tr>
<tr>
<td>Memory</td>
<td>0.5 MB</td>
<td>0.5MB,1MB FLASH</td>
<td>16MB,85MB dsk</td>
</tr>
<tr>
<td>Input</td>
<td>3 Buttons</td>
<td>3 Buttons</td>
<td>Speech</td>
</tr>
<tr>
<td>Dim (in.)</td>
<td>2.5x5.5x12</td>
<td>1.5x4.0x4.4</td>
<td>3x8x10</td>
</tr>
<tr>
<td>Weight (lbs)</td>
<td>2</td>
<td>0.5</td>
<td>8</td>
</tr>
<tr>
<td>Power (Watts)</td>
<td>3.8</td>
<td>1.1</td>
<td>13</td>
</tr>
</tbody>
</table>

More recently, the Wearable Group at CMU has been working on their latest wearable called Spot. “Spot provides workstation-class processing, storage and connectivity in a wearable form factor [4]. It takes the lessons learned in the previous generations of research at CMU one step further adding increased power levels and storage space.

2.2.2 Massachusetts Institute of Technology

Like CMU, MIT has a lab dedicated to researching and prototyping wearable computers. The applications and tools they’ve developed all serve different purposes than those from CMU. This wide variety of ideas is just a testament to the amount of area there is to be explored in this field.

One of the many wearables invented at the MIT Lab is the Remembrance Agent. This interesting tool is constantly with the user and provides one-line summaries (on a heads-up device) of information that may be relevant to the current context. Quite possibly even more fascinating, in addition to producing output, the Remembrance Agent is constantly taking in information about the user’s actions, what is happening in the surroundings and (if they are wearing a smart badge) who the user is talking with. All of this information is indexed and stored for retrieval later. The user can ask direct queries...
to the Agent and save more notes and observations that the computer may not have seen or known about (notes that the user was thinking about) [41].

The potential applications for such a tool are numerous. The Remembrance Agent can be invaluable to students because it will be able to detect that they are in the same location at the same time as a previous day and bring up all of their notes [41]. Or, if the user can’t remember how to get to a location where they were before, they can ask the Agent and will get back maps or notes from their previous trip. Finally, The Agent can keep track of who was met and at what meeting. Then when the user encounters that person again, the user can be alerted that they already know the person and pass some facts that may jog their memory.

Stochasticks is another unique application of the wearable paradigm, in this case to show its users more strategic billiards shots. This system uses a video camera attached to the user and aligned with their sight-line for input and a graphic display for output. Stochasticks automatically (through Computer Vision algorithms) the most strategic shots and shows them graphically to the user. It isn’t quite as robust an application or as useful in the “real world” as Remembrance Agent, but Stochasticks does demonstrate the development and use of a system which adheres to the basic rules of wearables. It provides “an integration of real and virtual environments which enhances the experience of playing and learning the game of billiards without encumbering the player.” [19]

Probably one of the most intriguing projects in the wearable arena is MIThrl. Towards the end of the 1990s, the researchers at MIT saw a need for a platform to assist in prototyping and testing wearables. They were trying to correct one of the most pressing issues plaguing researchers, making wearable computers comfortable to use and
unobtrusive. In previous years, the wearable computers served their purposes of being cognizant of the user’s context and acting accordingly. But, none of them had truly attained the unobtrusive status. Some involved cameras attached to the user’s head or headgear with goggles. They certainly didn’t blend into the user’s environment at all. They tried to fill the “need for a modular, flexible sensing and interaction system that was light-weight, reconfigurable, and easy to extend, use and maintain. [11]”

The first generation of MIThril was in 2000. While it served some of its intended purpose, there was still room for improvement on others. They had succeeded on creating a research platform that was flexible and modular allowing for easy reconfiguration and tuning when parts were interchanged. But, the hardware was expensive and not very lightweight [11].

MIThril researchers point out on their website that MIThril is an architecture and not a product. It’s a toolbox to draw from when designing a wearable system. They have fully documented everything for both the hardware and software on their website which is open to the public. Also, their CVS codestore is available to anyone who wishes to see how modules are written and perhaps offer suggestions for improvement [2]. It’s a similar school of thought to that of Open Source software. They feel like they are just members of a large community trying to use technology that already exists with some new innovations to enhance everyday lives.

2.2.3 Georgia Institute of Technology

Much of the wearable computing research done at Georgia Institute of Technology has its basis in the work conducted at MIT. This is primarily because of Thad Starner. Even though wearables date back almost a century, their acceptance into
mainstream research was just over 10 years ago and Starner is one of the pioneers. He is one of the founders of the MIT Wearable Group and the founder of the Georgia Tech Contextual Computing Group.

While he was a student at MIT, Starner worked with Bradley Rhodes on the Remembrance Agent project (described above). A key feature of wearables that Starner has pointed out in numerous publications is their ability to be aware of the user’s context and “behave” accordingly. “Suddenly, for the first time, our computers have the ability to see and hear the world from our perspective. Instead of being deaf, dumb and blind sitting on our desks or in our pockets, our computers might be able to observe what we do all day” [1]. He has continued making this a cornerstone of his research starting with his thesis at MIT and continuing with his work at Georgia Tech. He and his students proposed a context-based document system for wearable computers in 2000. At that time, none of the wearables supported the three forms of context (context linked to specific documents, context of the user and time context) in their storage systems. With their system, users can query with any of the three contexts as criteria to retrieve the document they desire [22].

Starner and his students continue to research ways to allow users to be more productive through the use of a wearable. A number of his students have begun to wear their computers and he has actually been wearing his computer since 1993. His wearable computer has evolved over the years, becoming more unobtrusive and comfortable to work with. And, with the advances in network connectivity and battery life, he is able to be connected to the internet in most places and use his computer for about 13 hours without having to switch the batteries. It is also interesting to note that none of the
applications he uses were written especially for a wearable. Starner is so adept at using his wearable computer; he wrote his 250 page thesis on it as opposed to using a standard desktop computer.

Another project to come from Starner’s Contextual Computer Group is the Mobile Sign Language Translator. This tool utilizes a wearable to assist the Deaf in communications with the hearing population. American Sign Language (ASL) is an actual language complete with its own grammar, vocabulary and culture. It is not simply a direct word-for-word translation of spoken English. In fact, for many of the Deaf, it is their first learned language with English being the second. This and the obvious issue that many hearing people don’t know how to sign makes communication without a translator quite difficult. Oftentimes, a simple pen and paper method is used, but that is almost 10 times slower than spoken English and also requires a pen and paper to be readily available. The tool Starner’s group is developing will translate the signer’s words into English, show the signer what the phrase is once translated and (if the signer approves), will be spoken aloud [14].

2.2.4 Other Research

Even though the vast majority of research on wearables has been done at CMU, MIT and Georgia Tech, other institutions have made significant contributions as well. Two interesting projects are the Electric Suspenders from The Aerospace Corporation and the Wearable Sensor Badge from UCLA.

Michael Gorlick from The Aerospace Corporation recognized that a major part of wearable research involved the actual design of the device. He proposes modifying articles we already wear to support a computer instead of devising something new.
Electric Suspenders are basically traditional suspenders made with conductive webbing as opposed to a regular cotton weave. Gorlick chose Nomex as the webbing fabric. Nomex is used in traditional suspenders and carrying straps. He makes the webbing conductive by incorporating two stainless steel conductors directly in it. Despite the fact that they are steel, they are still flexible [15].

To this conductive webbing, electrical conductors in the form of standard metal snaps are attached. Each part of the snap has a wire called a stinger underneath it. This is what causes the connection between the snap and the conductive weaving. There are quite a few advantages to using simple snaps to attach devices. First, they won’t have an effect on the flexibility of the suspenders. Secondly, snaps can be sewn on at virtually any location on the suspenders making this literally a plug and play technology [15].

Since the suspenders are conductive and equipped with a method to attach devices, the only part left is to design the components. Gorlick proposes a shared power hub. This greatly reduces the risk of battery failure since, the batteries will not all lose power at the same time. And, the weight the user must carry is also reduced because there doesn’t need to be a single power source for each attached device [15].

The various devices, including the battery packs and network, are attached to the suspenders using cloth packets that are snapped on. The placement of the devices is completely up to the user. If they generally carry a bag on their left shoulder, they can attach all of the wireless components to the left strap of their suspenders [15].

The Smart Kindergarten project at UCLA attempts to use wearable computers to answer questions about how children learn and interact with others in their class. Unlike the Electric Suspenders and the devices from Starner’s group at Georgia Tech, this tool
gets its input passively and the output is not readily available to them. The students don’t
do anything special to record data and, only after it is compiled, the teachers and parents
query a system to see the output. In this case, active input and output viewing really
don’t make sense [36].

For this study, the children all wear an iBadge that is virtually flat, approximately
3” by 2” and weighs 2.3 oz. Developing a device that was unobtrusive and light enough
for a child to wear was important to this study. The entire classroom was involved in this
project. The Sensing Infrastructure consisted of the students with their iBadges and toys
and books dispersed throughout the room all with embedded iBadges. The Middleware
Infrastructure provides the storage, management and presentation of data. It has
components like microphones, the sensors that collect iBadge data and the networking
piece [36].

The iBadges provide information about where the child was during the day, what
toys they played with and what other children they were with. The iBadges placed
throughout the classroom are used to calculate the student’s absolute location as well as
their relative location. AdHoc queries can be run against the data collected to provide
teachers and parents information about the students [36].

This tool provides useful information, but is still in its very rudimentary stages.
The authors suggest future research to include more networking and speech recognition.
Despite the work it needs to be more robust, a study such as this one demonstrates once
again the opportunity available in this field and the varying situations where this
technology can be useful.
CHAPTER THREE

CHALLENGES

It seems like a wearable computer is another one of those “I know what it is when I see it” kind of things. The first challenge in this field is simply the basic question of defining wearable computers. This paper follows the example set by other prominent researchers in this field and defines a wearable computer by the set of characteristics it must have.

After a wearable is “defined”, other more concrete challenges in both hardware and software present themselves for developers to handle. The list of challenges discussed here is not exhaustive, but it provides a good cross section of what developers must face.

3.1 Power

One of the biggest challenges facing researchers in the wearable computer field is power. Finding the best way to power a device is not limited to wearables. But, as the name suggests, one of the required characteristics of a wearable computer is that it be wearable. This eliminates using a power cord, the obvious solution, since the user would be tethered to an outlet.

Starner has suggested many potential solutions to the problem of powering a wearable, but despite the positives of each, none is without drawback. Batteries have
worked well for other mobile devices like a cell phone, so they would seem like a logical suggestion for a wearable computer. But, as has been pointed out previously, a wearable computer presents unique issues. Since it is carried with the user everywhere, any battery must be kept as light as possible. Also, a wearable device is meant to be unobtrusive. If a user has to worry about curtailing their usage because their battery may not stay charged and then have to make the effort of recharging it, the wearable computer becomes more of a hassle than something that could be used as a constant companion [53].

Also, a wearable computer is not always one single device. Generally, there are different components that can be attached depending on the user’s situation. If all the devices use one single power source, that source would require the capability to handle the strain of multiple devices. This may require that the power source be large and quite possibly heavy and bulky. Also, this creates a single point of failure if the batteries die. On the other hand, each device could have its own power eliminating the single point of failure. But, in that scenario, the user must keep track of the charge left in each particular battery and recharge accordingly. The developer needs to evaluate this on a case by case basis and determine which solution would work best, assuming batteries are the option their option of choice [6].

Over the years, batteries have become smaller and more economical which has had a large impact on mobile computing. But, this has actually been a double edged sword. “Although economical batteries are a prime agent behind this expansion, they also limit its penetration; ubiquitous computing’s dream of wireless sensors everywhere is accompanied by the nightmare of battery replacement and disposal” [34]. As long as batteries require recharging and eventual replacement, they will be a limiting factor.
But, luckily a battery isn’t the only possible source for power. Energy harvesting research is not something new; it dates back to using windmills to power water wheels. But, interest has been renewed by the mobile and wearable computing industries [34]. Starner suggests generating power from the user themselves. A user’s actions can create energy that can be harnessed and fed back into the wearable. Capturing the energy produced by a person just walking has shown to be a promising way to power a device. Research done on “electric shoes” has shown that an average weighted person can generate 7 Watts of power from one step [34]. The key now is to harvest this energy in such a way that it is unobtrusive and seamless to the user.

Research is underway to allow a wearable to be powered by the environment. This seems to fit the best with the wearable concept and characteristics. The user would go about their daily routine and the wearable would be cognizant of the environment and garner power in whatever was appropriate for the conditions. The user would not even be a part of the process, making the wearable even more unobtrusive [53].

3.2 Privacy and Security

Starner and others feel that privacy and security are completely separate concepts and should not be discussed together. Privacy is the user’s right to shelter their personal information from others while security is keeping unauthorized users away from personal information [54]. At first glance, it seems like Starner and his colleagues are incorrect in this statement and that privacy and security can be discussed together. On the surface, the two terms are similar, but digging deeper shows their true differences especially in the wearable field. But, researchers do need to be aware of the existence of both when creating their prototypes and making design decisions.
General security can be handled with standard methods like encryption. Like any other system, there is always the concern that the security can be breeched by a lack of human integrity. The issue of security with a wearable system is most visible in the case where it actually conflicts with privacy.

Starner illustrates this by describing a situation quite common in today’s workplace. Employees in most companies are required to wear an identification badge to allow them access to areas, both non-restricted and secure, and to positively identify them to other employees. These active badges are wearable computers because they gather and store data. While these wearables’ purpose is creating a secure environment, employee privacy and personal security issues arise. Since data is gathered whenever an employee goes in and out of a badge-protected door, it is possible for an employer to take this information and form trend reports on how often they use the break room, for example. Even if they have nothing to hide, most people feel this type of monitoring is a violation of privacy. Of even more concern is the possibility that someone could break into the database that stores this information and find out what time an employee generally arrives and leaves for the day. This could easily be used by a stalker [53].

Situations like these could be rectified by allowing the employee to determine what information about them is accessible and by whom. Also, the employer could agree to delete location information after a certain amount of time and not to use it for anything other than emergency procedures (for example, making sure all employees who badged in are accounted for in the event of an evacuation) [53].

Starner further addresses the privacy issue by creating a list of five components, summarized in Table 3.1, a wearable should have to protect the user’s information.
### Table 3.1 Protecting a User’s Privacy

<table>
<thead>
<tr>
<th>Method</th>
<th>Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physical</td>
<td>Carry any crucial data on their bodies</td>
</tr>
<tr>
<td></td>
<td>Shield wearable from unintentional wireless emissions</td>
</tr>
<tr>
<td>Technological</td>
<td>Encryption</td>
</tr>
<tr>
<td></td>
<td>Biometric Identifiers</td>
</tr>
<tr>
<td>Legislative</td>
<td>Make laws specific to technologies</td>
</tr>
<tr>
<td></td>
<td>Define when privacy is considered to be violated</td>
</tr>
<tr>
<td>Social</td>
<td>Utilize existing social standards</td>
</tr>
<tr>
<td>Obscuring</td>
<td>Hide sensitive information amongst non-sensitive</td>
</tr>
<tr>
<td></td>
<td>Make personal information look standard by using a deceptive filename</td>
</tr>
</tbody>
</table>

#### 3.3 Network Connectivity

In today’s day and age, connecting to another computer or the internet is almost always a necessity. A stand-alone system simply doesn’t fill the need of users anymore. A user may need to access a remote file or application, schedule a meeting with colleagues or search for information on the internet.

Many laptops use a wireless network. There is not a problem with this because they are generally less mobile than a wearable user. Once a wireless connection is made on a laptop, it is kept (unless the network itself goes down). With a wearable computer, that is not really the case. Because of this inherent problem, network connectivity creates another challenge for wearable designers.

There are a few approaches to handle this challenge depending on the user’s requirements for their wearable. With a tool like the Remembrance Agent, access to a real-time network may not be a requirement. For the Remembrance Agent, the wearable
accesses files and databases to assist its user with daily tasks. These resources can be kept directly on the wearable’s hard drive, assuming it is large enough, and persistent network access is not needed. The user can back up their wearable periodically either by connecting to a LAN or wirelessly, when they are in range. Also, keeping all of the information on their wearable device can make the user feel more secure, eliminating another challenge, as well [41].

But, not all wearables can serve their purpose without a persistent or semi-persistent network. For these, other options must be explored. Starner divides the possible network interfaces into three categories depending on their requirements. Each of these has their own design concerns.

### 3.3.1 Wearable to Fixed Location

The first category is communication from a wearable to a fixed location. This is probably the most popular and the most needed type. Cellular phones illustrate this configuration. They get their signals from cellular towers at fixed locations. Any cell phone user already knows the inherent problems with this design. There are times when a tower can’t be reached and either a call gets dropped or one isn’t even able to be made. And, cellular towers are proprietary and often times, won’t accept a signal from a different company’s phone. Finally, there is a predetermined finite number of connections that a cellular tower can offer. At heavy traffic times, like during an emergency situation, when quite a few people are attempting to use their phones the towers can’t keep up and service levels are not as reliable.

Wearable computers have the same issues as cellular phones, but they have unique features that make the solutions different. Repeaters, devices which take a signal
and rebroadcast it, are used for cellular networks. To apply this technology to wearable computers, Starner suggests a solution that slightly out of the box. He points out that wearable users are seldom more that a few miles away from their car. Also, he notes that cars are generally not very far from each other. So, he proposes creating a network of automobiles. The cars would have repeaters for the stationary network to which the users need to connect [53]. While this is a unique potential solution, Starner assumes that there will be a lot of wearable users and they will all have repeaters in their cars. In the future, this may be the case, but for now this will probably only work in a university parking lot.

Another quality of wearables is their potential to “predict” their user’s actions. Since they serve as a constant companion to their user and people are generally creatures of habit, it’s possible for a wearable to anticipate what the user will be doing within any given time. For example, if everyday, the wearable user accesses a particular file from their computer at their office at 11 AM from their wearable, it’s probably safe for the wearable to go ahead and download it at 10:30 if they happen to be near a network at that point. Starner calls this aggressive caching [53].

3.3.2 Inter-device Communication

The second type of networking Starner feels is important for wearable computers is inter-device communication. In a wearable configuration with multiple devices, these devices may need to communicate with each other. For example, with the Electric Suspenders, each component can be attached as deemed necessary based on the situation. The devices will need a way to communicate with each other. There are two potential solutions to this problem.
First, a wireless network with a protocol such as Bluetooth could be used. This would create an expandable network and with the devices in such close proximity, a high level of reliability would be achieved [31]. But, a wireless interface would need to be installed in each device. In most systems, this would not be an issue, but wearables need to be as small and unobtrusive as possible. Depending on the physical design of the wearable, this could pose an issue. But, if the developers decide this is their method for communication in advance, then they can account for it in the design.

The second approach would be the one that the developers of Electronic Suspenders already use. They implement a hard-wired network that is far from old fashioned. Instead of the standard Ethernet cables snapping in from each device to a hub or wall socket, with the Electronic Suspenders, the threads in the suspenders themselves provide the connectivity. As described above, conductive threads run through the fabric and metal snaps (which also serve as conductors) are used to attach each device to the suspenders [15]. For a wearable whose components connect physically to each other, a design such as this is ideal. It eliminates the need for additional hardware and utilizes the physical connection already in the fabric.

### 3.3.3 Wearable and Stationary Object

The final network configuration is the communication between a wearable computer and a stationary object. The need for this type of communication is seen mostly in location systems. Generally, this communication requires a lot of power (which is already a known issue with wearables). One team of researchers proposes using a tag in some part of the wearable and reader tags in the objects they want to track. A small stream of bytes would be send back to the wearable from the stationary object and that
would be used at a key. This small transmission won’t take much power. After this key is obtained, the wireless network (which doesn’t consume quite as much power) is used to track the location after that [53].

3.4 User Interface and Unobtrusiveness

If a wearable is meant to be a constant companion to its user, then it must blend seamlessly into their daily activities. This is one of the fundamental characteristics of a wearable computer. That being said, after power, two of the biggest challenges facing wearable computer developers are the interface and making the device unobtrusive.

In “Wearable Computers: A New Paradigm in Computer Systems and Their Applications”, Smailagic discusses the three axes that must be accounted for when designing a wearable. These three axes offer a high level guide detailing the issues. But, the three axes are not distinct. Some issues are a part of more than one axis [48].

3.4.1 Human Axis

First, there is the human axis which is the relation and interaction between the user and the wearable device [48]. Comfort is a key element in the human axis design. The users will have their wearable on their bodies for long periods of time (theoretically all day). Therefore, they will require a device that is comfortable and does not inhibit their movements in any way. Design for comfort includes size and weight of the device, the placement on the body and the fabric from which it is constructed.

The wearable needs to be both small and light enough to be worn without distraction or hindrance. One approach to this is the Electric Suspenders, previously discussed. This design distributes the components in whatever configuration the user chooses [15]. This technique satisfies the comfort need, but suspenders may not be an
accessory that most users will want to wear. This brings us another design concern, the social aspect. A wearable also has to be something a user will want to wear and not be embarrassed.

The research in eTextiles addresses the social aspect. These research teams propose outfitting traditional articles of clothing with wearable components. Some parts, like the battery pack, are worn in a pocket, but the rest of the system is integrated directly into the fabric. Some of the eTextiles are even washable.

3 4.2 Computer Axis

The computer axis involves the actual physical construction of the wearable system. Here is where the lines between the axes can get blurred. The actual size of the wearable and the fabric/material type used for the construction are considered to be part of the human axis as well as the computer axis. These two traits provide examples of how, in addition to satisfying the requirements within each axis, a balance between the axes must be found [48].

The computer axis says that the fabric/material used to construct the wearable must be durable and able to endure almost constant usage without getting damaged. But, the human axis requires a fabric that is comfortable to the user. The most strong and durable fabric may not be very comfortable.

The computer axis also includes power consumption and the software user interface [48]. The issues with power consumption go hand in hand with those dealing just with powering the wearable. The designers need to think both of finding a way to power a wearable and conserving power whenever necessary. The goal is to make the wearable able to function for as long as possible without any attention from the user to
change a battery or recharge. Power consumption also deals with heat. If the user is going
to have their wearable on their body almost constantly, they certainly don’t want it to be
giving off a lot of heat.

The size of the device is an issue, as well. At first, the smaller the better seems
like the best approach. But, given some further thought, this may not always be true. If
typing is a means of input, the keyboard must be large enough to be able to be used. The
keyboard on a Blackberry is an example of a small keyboard that can be useful. But,
Starner’s wearable uses a one-handed keyboard called a Twiddler as its input. This is a
great approach to this problem, but requires some training for the user to get accustomed
to a chord type keyboard. This same logic applies to output from the wearable, too. The
output display needs to be large enough for the user to distinguish what is on it. This
could be solved by using a head up display [54]. But, as with suspenders, not all people
will feel comfortable wearing a heads up display.

3.4.3 Application Axis

The final piece in the paradigm is the application axis. This axis addresses the
challenges in dealing with the actual applications written for the device. These challenges
have nothing to do with the hardware or physical construction, but still can have an effect
on the design and look of the wearable.

For wearables that target one specific use, designing applications to meet the
user’s needs, as well as to satisfy the general requirements of a wearable, is not as
difficult. Devices that fall into this category generally have users who are “experts” at
their tasks and the interface doesn’t need to be quite as suited for the general audience.
For example, applications written for a wearable used by home inspectors may simply be
a listing of all the forms they use for the various types of inspections ordered. They can select the proper form and a merge type of application would fill in all the known information. They complete the inspections and upload the form to their office. The output can be a heads up display and the input can either by a Twiddler chording keyboard or something similar to a Blackberry since there won’t be much text typing, just checkboxes.

But, for a wearable for general purpose use, assumptions about what the user will be doing can’t be made. One approach would be to make a wearable usable and accessible to all users regardless of needs, abilities and environment. But, this would create a device that would be quite cumbersome and confusing to use. A second approach that is gaining a lot of interest is to allow users to customize the interface based on their own personal requirements. This thought arose from studying a disabled person’s specific needs for public tools like ATM Machines. If they could “bring along” their own interface, they could use tools such as these without additional assistance. This is called an “alternate user interface” [14].
CHAPTER FOUR
WARM OVERVIEW

As its name describes, WARM is a wearable device designed to assist its users. This assistance can come in multiple forms depending on each user’s specific needs and their lifestyle. WARM is modular and can be customized to fit each individual user’s situation.

The second part of WARM stands for Remote Monitoring. This feature is especially useful for the elderly population. The remote monitoring system makes use of a secure web application and wireless networking technology. It allows family members, or others granted access, to check on the status of the elderly person and to be alerted in the event of an emergency. The functionality and user interface for this part of WARM will be discussed in detail in Chapter 5.

4.1 Users

“The fastest growing disabled group is not people aging with a disability; rather, it is people aging into disability” [44]. While they may cause an adjustment in daily routines, diseases such as failing eyesight, hearing loss and dementia don’t warrant assisted living or a nursing home. People with these conditions can live independently, but may require just a little assistance to give themselves and their loved ones peace of mind. It is this population that was the inspiration for WARM.
WARM is not a device for the severely ill or infirmed. There are wearable devices currently available and in the prototype phase which fit that need. These use medical monitoring protocols to check many aspects of their user’s health. Generally, individuals who require this kind of constant monitoring are already residing in a nursing home or an assisted living facility.

WARM’s target user group is people in their early 80s living either by themselves or with another person around the same age. They don’t have any major medical problems, just the issues that naturally come with getting older. But they may need some assistance with their medication schedules, appointments etc.

4.2 Features and Functions

WARM is more than just a system to remind its users that it’s time to take their medication. WARM users have the flexibility to add any type of reminder to the system and control the actions that the wearable device will take. It’s been shown that similar wireless systems have greatly improved the quality of life of users by helping them retain their independence [45]. WARM’s purpose is enhancing its users’ lives by serving as a companion to assist them with daily tasks.

4.2.1 Reminder

As previously mentioned, WARM has a reminder feature for its users. These reminders can be grouped in predefined or custom categories. Within each category, the notification style, icon and action taken in the event of an unanswered alert can be customized.

The alerts can be as detailed or as simple as the user chooses. The description entry field is free form text and can therefore be written in a way that is meaningful to the
user of the WARM wearable device. For example, if a user needs to take medication twice daily, when the alert arrives at their device, a picture of the medicine will be displayed along with the name and physical description to help make sure the correct medicine and dosage are taken. Once the user has taken the medicine, they can clear the alert from the WARM device. If the timeliness of the medication is important, the alert can be configured to re-alert the user if the alert is not cleared within an acceptable number of minutes. If the alert is still not cleared after the second attempt, the user’s primary contact will be notified. All of these features are configurable at the time the alert is defined and can be modified at any time by the user or some one with the proper level of authority.

The WARM server is secure. The WARM user themselves and trusted remote users can log in at anytime to check the status of alerts, modify existing alerts or create new ones. Each state of an alert is logged on the server and can be queried by those with the proper access level. The WARM user’s privacy can still be protected because they can set alerts to be viewable by just them or members of their trusted group.

4.2.2 Remote Access

One feature that makes WARM unique amongst other wearable companion devices is remote access. This access serves two distinct purposes.

First, it allows for the WARM wearable device itself to be smaller and to have a simpler interface. All setup is performed on a remote computer through simple input screens which will be described in detail in Chapter 5. Since this is a web-based application, the setup can be done on any internet connected computer. Because all input is done on a “normal” keyboard, the descriptions can be made more meaningful. No one
will have to use a tiny mobile keyboard or learn to use a chording device like the Twiddler. Also, any information entered can be more easily verified for correctness and modified if necessary quicker with a standard sized monitor. The more simple and convenient the interface is to use, the more likely the product is to be used often and properly.

Second, remote access allows those with the proper authority level to “check in” on the user. For example, if a user’s child wants to verify that their parent has taken his or her medication, they can log on and quickly run an activity report to verify. Also, the logging capability of the WARM system could prove invaluable in an emergency situation. If a doctor needs to know which medication a patient has taken, the length of time, dosage and frequency, WARM could generate this report and save any guesswork.

4.2.3 GPS

If the user chooses, the WARM device can be equipped with a GPS tracking chip. This may not be wanted in all cases, so it is an additional feature that can be added upon request. Access to GPS location information is confidential and viewable only by those with the specific access in order to protect the WARM user’s privacy. Location information is logged to the WARM Activity database just as reminders and can be accessed via similar reports.

This level of monitoring brings up the question of privacy. There is a tradeoff. In “Managing Care Through the Air” one user of a health monitoring system said “I am perfectly willing to share my information with my daughter or whoever in order to continue to live in my own home” [45].
4.2.4 Health Monitoring

The WARM application comes with a basic health sensor that monitors pulse and body temperature. This sensor communicates with the WARM wearable device through Bluetooth. Readings are logged to the WARM Activity database. These readings can be coupled with other data to provide trend information which may be useful to a doctor in the case of an emergency. And, research is being done to use similar type information to help diagnose serious cardiac problems long before the usually symptoms start to occur [45]. The health information can be accessed by running a report by those with proper authority.

4.2.5 Emergency Call

One of the goals of WARM is to provide peace of mind to its user and their loved ones. A way of doing this is incorporating an emergency call feature into the WARM wearable device. An emergency call can be initiated in two ways. First, there is a recessed button on the device itself. If this button pressed for 5 continuous seconds, emergency officials and the user’s emergency contact will be notified. Second, if the user’s pulse and body temperature are not within the acceptable range; emergency officials will be called to their assistance.

4.3 Components

While the WARM device gets a lot of the focus, the system is comprised of four separate components.

4.3.1 Health Sensor

The WARM system is not devised for severely infirmed individuals. But, a basic component of the overall system design is still its health sensor. The sensor monitors
body temperature and pulse. It uses Bluetooth to communicate the results to the WARM device. The next time the synchronization process between the device and the server occurs, the user’s health statistics are updated, as well.

In recent years, there has been a large amount of research on using sensors for health monitoring. Issues that are driving the interest in sensor monitoring are the predictions about the worldwide population of over-65 persons reaching 761 million by 2025, more than double the 1990 figures [45].

The focus of this thesis is the WARM application, not the architecture of the health sensors. The data tracked by the sensors is basically the same regardless of the implementation. There are two viable options for health sensors that would work with WARM. In “An Architecture for Wearable, Wireless, Smart Biosensors: The MoteCare Prototype”, an architecture for wearable health sensors called MoteCare is proposed [21]. MoteCare uses wireless broadband for the sensors to communicate to the server.

Another architectural approach for health sensors uses mobile ad-hoc networks (MANET) to transmit data back to the server [61]. This is an alternative to transfer the information back to the server which would work even if the user was out of the Bluetooth networking range. The researchers are working on prototypes and ways to handle issues which are problems for wearables like privacy and power.

As with most aspects of WARM, the health sensor is a configurable feature. If the user is diabetic, for example, their sugar could be monitored. But, when deciding what needs to be monitored, the comfort of the user needs to be considered. Not all statistics can be measured with the same sensor. Too many additional sensors could get to be cumbersome and start to adversely impact the user’s daily activities.
4.3.2 WARM Device

The heart of the WARM system is the wearable device itself. The device is worn on the wrist, in the fashion of a watch. Instead of a watchband, WARM has more of a cuff which has a Velcro closure. Its display is 2 inches high and 4 inches across and the text is displayed in a landscape fashion. This is because it’s more natural for someone to turn their forearm parallel to their body to read, like a watch. And, the landscape display gives more real estate to the GUI. The messages displayed and contextual action buttons can be more meaningful since there is more screen real estate. This leads to a simpler, more user friendly experience since more explanation can fit on the screen.

The display screen is full-color and standard settings such as contrast and brightness are adjusted by buttons on the left. The volume controls are on the right. Below the screen are the action buttons. These buttons’ purpose is based on context and is clearly noted directly above them on the screen. The GUI has a consistent design where if the OK button is on the left in one case, it will be there in all cases. This consistency also enhances the user’s experience because a level of comfort is quickly gained.

4.3.3 User’s PC

The hub of the WARM application is the personal computer at the user’s home. This computer can be used to connect to the WARM web application, just like any remote computer. But, it additionally serves as the “link” between the WARM wearable and the web application server. This computer needs to be connected to the internet and also run wireless networking and Bluetooth. In addition, it requires a modem connected to a dedicated phone line and a Universal Power Source in case of network issues and power outages.
The computer at the user’s residence also runs the synchronization program. This executable has a small footprint and is the means by which the WARM wearable makes and receives updates.

### 4.3.4 WARM Application and Database Servers

The final two components of the overall WARM system are the application and database servers. In essence, these are really the first components because the application could not exist without them. Since WARM is a web application, users log into the application server through an internet browser to handle all setup and configuration and also to run reports and query activity. The application server, in turn, updates or queries the database server via stored procedure calls.

As events occur on the wearable device, they are sent to the synchronization module on the user’s computer and an on demand update is triggered. Information from the health sensors are on a schedule and are pushed to the server at predetermined time intervals. Likewise, an update from the server is pushed to the WARM wearable device at soon as the synchronization module receives it. This constant communication and update policy keeps all components in synch.
CHAPTER FIVE
WARM DESIGN

This chapter looks in depth at the WARM design. First, the architecture of the whole system will be discussed. Then, the user interface of each of the components will be detailed, including screen designs.

5.1 WARM Architecture

The function of each of WARM’s components was discussed in Chapter 4. Figure 5.1 shows the architecture of the WARM system including the users.

5.1.1 User’s Personal Computer

The personal computer at the user’s home performs scheduled and on demand synchronizations between the WARM wearable device and the database server. The footprint of the synchronization programs is very small, so there isn’t much to install or to maintain. The run as scheduled tasks in the Windows Scheduler, but are called on demand, as well.

But, since the user’s PC is the link between the wearable device and the web application, the internet connection needs to be as reliable as possible. Because of this, the PC must have a modem with a dedicated phone line to connect to the internet in the event of a high speed internet or fiber optic outage. Third party vendors have tools to handle these situations. When the primary NIC (network interface card) loses its
connectivity to the internet, the modem automatically initiates a connection. If the web application server does not receive a signal from the PC, an informational alert is sent.

![Figure 5.1 WARM Architecture](image)

In the event of a power outage, the PC will switch to its battery backup. This condition will trigger an informational alert, as well. The PC performs self-checks periodically and will send an alert if it detects a problem. This is done through a third party tool, as well. None of the tools required to monitor subsystems and perform self-checks are CPU intensive or require a lot of additional memory. Because of WARM’s table driven architectural approach, any of these tools can be integrated into the system and use the contact and user information already stored.

Since the WARM users are not infirmed and don’t use WARM for essential life functions, it is sufficient to alert the user and contacts when there is either a network or power outage. As stated before, this is a companion device meant to assist its users.
5.1.2 Application Web Server

The web application part of WARM runs on a separate server. This server is housed at a production site with generator power backups and is maintained like any other standard production system. Since no user or processing data is stored on this server, it is not backed up nightly. But, a full backup is taken after each new installation of the product and prior to any operating system or system software changes.

The application server is a Pentium II machine running the Windows 2003. It should have 4 gigabytes of RAM. This amount seems high, but is required for the high availability setup, described later. The WARM application requires the .Net 2.0 Framework and Internet Information Services (IIS). The Graphical User Interface code is written in .Net and the code behind files are in C#.

5.1.3 Database Server

There are three Microsoft SQLServer databases which reside on a dedicated database server. These databases reside in a production environment like the application web server. Since this information is constantly changing, a full backup is taken nightly and incremental backups are handled throughout the day via logging.

To maintain high availability, the database server and the application web server act as hot backups for each other. This is called clustering. They have the same levels of all operating system software and maintain a constant connection. If one box goes down for whatever reason, the other box will seamlessly take over the name and IP address of the other so it can be resolved via DNS. The application as a whole will operate in this mode until the other server can be repaired. In order to continue to run the application and database without a noticeable degradation in performance and response time to the
user, the servers both have more than the minimum recommended memory and CPU requirement. As the numbers of users increase, analysis should be done to determine any new hardware requirements.

The first database, WARMClient, holds the user information and drives security with authentication and access rights validation. Its model is shown in Figure 5.2. This relational database is where data specific to the user setup resides including contact information, general and medical reminders and the user-defined details associated with each. Each addition and modification to rows in the database is marked with a user ID and timestamp. This could prove helpful if, for example, the dosage of a medication changed and a doctor questions when it occurred.

5.2 WARMClient Relational Database Diagram
The second database, WARMActivity, stores the information about the history of each reminder. It logs when the reminder has been triggered, the time it was closed and the number of times it was sent. It is from here that emergency alerts to user’s contacts are triggered. Also, the health sensor data and GPS location data are kept in tables in WARMActivity. Because trending information from the sensors and alert activity may be valuable, the data is kept for a year and then rolled off to an archive system. The archive database has the same table structures, so a report can be run to trend for more than a year with just a simple join to additional tables.

The third database, WARMReference, holds information that will be helpful to all WARM users. The pharmacist’s description of medications is stored here along with a link to an image file with a color picture of the medication. And, if the medication has special instructions, like “take with food”, a column in the table will flag this and the user will be asked if they would like to add a reminder for this event, as well.

As WARM’s functionality grows with future releases, additional information of this type will be in this database. It is important to provide an accurate description and an image of the medications to the user. Some patients are prescribed multiple medications which may have similar sounding names and appearances. This helps to ensure that the correct medication is taken at the correct time.

5.2 User Interface

Because of its design architecture, the WARM application as a whole required two completely different user interfaces, one for the web application and one for the wearable device.
5.2.1 Web Application Design

The web application serves multiple purposes in the WARM application. The homepage for the application is the login page, shown in Figure 5.3. Users and all contacts, even those without access to secure areas, all have their own id and password. As seen in Figure 5.4, the user id and password are validated and the first screen has a list of functions to which they have access. Each function will be described in the following sections.

![Figure 5.3 WARM Web Application Login Page](image)

5.2.1.1 User and Contact Setup

The web application is the mechanism to set up the main user and their contacts. The user’s contacts may also have access to add/modify alerts, add/modify contacts and
run reports to determine if the user has been clearing their reminders and their health situation. Also, if the GPS feature is turned on, contacts may also be able to view the user’s location. These privileges are set up through the web application’s interface. In order to assure the user’s privacy and to avoid mistakes, the process to establish the various levels of access must be easy to use and understandable. Confirmation is provided after any additions or updates are made to verify that the user’s intended purpose was followed. Figures 5.5 and 5.6 show a sample of the contact input screen and a confirmation screen.

Figure 5.4 WARM Web Application User Options Page
The same screen is used to both add a new contact and to update an existing contact. If the user wants to modify an existing contact, they first search for the contact by name. This search pulls back the information and access information for the contact and fills in the data fields at the bottom of the screen. The label on the lower panel changes to “Update Contact Information” to make the update action clear.

![Figure 5.5 WARM Web Application Contact Page](image)

### 5.2.1.2 General Reminder Setup

The input screens for the general and medical reminders are similar, but there are still two separate options on the home page. There are a couple reasons for this. First, different levels of access are required for these two options. With two options, this difference in access levels is explicit. The main user always has the option, at an
individual reminder level, to protect reminders from anyone else. WARM is supposed to assist its users, not rob them of all privacy. If a user wants to create a reminder for themselves, they are most certainly allowed to do so.

Secondly, the required input fields are different for the two categories of reminders. Actually, there are quite a few more required fields for the medical reminder and more stored procedures are called. So, in the interest of usability, keeping the medical reminder and the general reminder as separate screens makes the most sense. The business rules of which fields are required and which are optional get more complex when two types of reminders are involved. The user will have a better experience if the input screens are simple and tailored to what they are doing, this will also lead to less confusion and incorrectly entered reminders.

General reminders are for basically anything except medication. There are less input fields and more room for descriptive text. But, these reminders can still be classified into categories such as appointment, birthday, household task etc. By default, these types of reminders won’t trigger a notification to any emergency contacts if they’re not cleared. But, if the subject matter of the reminder warrants, the creator can change this setting.

These reminders can be set anytime into the future and can be either a single occurrence or a recurring event. The start date can be either typed into the input field, or the calendar icon may be selected. Standard date edits are run on any dates and an invalid case (for example, 29 days in February when it’s not leap year or a start date later than an end date) will not be allowed.
As with the contact setup, when a new reminder is created, a confirmation is posted to the screen. Until the OK button is selected, the reminder won’t be stored to the database.

Figure 5.6 shows the input screen for general reminders. The confirmation screen is similar to the one for contacts.

![General Reminder Screen](image)

Figure 5.6 WARM Web Application General Reminder Setup

### 5.2.1.3 Medical Reminder Setup

As stated before, the medical reminder setup is a separate option on the landing page. The basic idea of this input screen is the same as for the general reminders. But, for medical reminders, additional fields are populated once the name of the medication is selected.
The names of many medications are similar and often difficult to spell. To make sure that the user is creating the medication reminder for the correct medication, the input box allows them to type in some (at least 3) letters of the name or all and then select SEARCH. This will call a stored procedure on the WARMReference database and return matching medications as a drop down list. Once the medication is selected, a picture of the medication along with a written description is populated into the “picture” and “description” fields. The information in the reference database is updated often, but to be safe, the user should verify that the medication looks like the one they’re intending to enter. If there is a discrepancy, they should contact their pharmacist for clarification.

Like in the general reminder setup, the medication reminders can be one-time-only or recurring. Generally, medication reminders are recurring and there is a start date, end date and number of doses a day. If the medication is supposed to be taken four times a day, for example, WARM will calculate the times by dividing the day evenly.

Additionally, the reference database stores information about special instructions. If there are any, a note will pop up before the reminder is saved informing the user of this and asking if they’d like to be reminded of this information, as well. For example, if the medication is not to be taken on an empty stomach, the WARM application will set up an additional reminder 30 minutes before the medication reminding the user to eat something.

As with the general reminder setup, nothing will be saved to the WARMClient database until the user selects OK on the confirmation window. Figure 5.7 shows the medical reminder setup screen once the user has selected their medication from the drop down list and the reference fields are populated.
5.2.1.4 Activity Reports

The final piece of the user interface on the WARM web application is the activity reports. Users and contacts, with the proper level of authority, may run a variety of reports. They first select the type of report (general reminder, medical reminder or health sensor). Then, they select which specific information they want to see. Once the report is complete, the results are shown on the screen. The user can print the report or export it to Excel format.

5.2.2 WARM Wearable Device

As discussed in Chapter 4, the WARM wearable device is worn in a similar fashion to a watch. On the average person, this doesn’t give a whole lot of room for a
graphical user interface. Because of this, the interface is as simple as possible, but is still designed to give the user the information they need in a clear and understandable form.

Figure 5.9 shows a rendering of the WARM wearable device. The screen is just about the whole device. This is because the only input from the user is the selection of the two buttons on bottom.

5.2.2.1 Reminder Screen

The Reminder Screen is the core of the WARM wearable. When the user hears the audible signal or feels the device vibrating, the reminder which triggered it will be on their screen. The placement of the information on their screen will be the same whether it’s a medical or general reminder. This consistency makes the reminders easier to understand. This makes the most use of the small amount of screen real estate available.

The actions of the two buttons are configurable during setup. For a medical alert, by default, only one button has an associated action and it is “OK”. Generally, this are not changed because the timeliness of medication doses is important. For a general alert, the second button defaults to “LATER”; meaning that the reminder will be resent in 15 minutes.

Figures 5.8 and 5.9 each show a sample screen for both of the reminder types that the WARM user will receive. They can choose to be alerted silently (vibration), audibly or both.
5.2.2.2 Contact Lookup

The WARM Wearable is not a PDA, but a nice feature to offer its users is a way to look up information on their contacts. When a contact is entered, their name, address and phone number are required fields. So, a small database of this information is kept on the WARM wearable. The general assumption is that a user will only have a handful or contacts. So, the fact that there isn’t a means of text input is not a problem for the contact lookup.
The user can access this screen at anytime that there isn’t a reminder awaiting a response by one of the two action buttons. Once they’re on the contact lookup screen, an alphabetical list of their contacts will be displayed. The left action button will move the selection bar down and the right button will move it to the right. This is a rudimentary interface, but for this first release of the product, it will suffice. Chapter 7 will outline future enhancements, and a better interface for user input will be discussed.
CHAPTER SIX
WARM APPLICATION SIMULATION

The WARM application is in an early stage of the development cycle. While the hardware has been mentioned, this thesis has focused on the software aspect. The user interface, database design, process flow and software architecture have all been described. These systems are what will be simulated to test the functionality and features of the WARM application.

6.1 Simulation Environment

The goal of this simulation was to, as closely as possible, mimic the actual production environment in which WARM would run. In the proposed application, the database and web servers are two separate Windows servers. For this simulation, they are running on one server. This change does not modify the functionality of the system and the source code does not require a change. Running the both database and web application on one device can cause performance issues in high usage, and is not recommended for production applications. But since this is a simulation to demonstrate functionality, it is an acceptable risk.

A laptop running the Windows XP operating system with 1.5 gigabytes of RAM was used as the PC in the WARM user’s home. The synchronization module was installed on it, and the necessary entries were made in the Windows scheduler just as they
would be in an actual environment. But instead of communication via Bluetooth from the user’s PC to the wearable device, rows are written out to a database table using a stored procedure. This was a modification in functionality from the proposed application, but was necessary since an actual wearable was not created. Since the rows are all date and time stamped as they are written out to database tables, it made it possible to gather timing information of the actual application without the variable of network speed. See Figure 6.1 for a summary of how the actual application and the simulation process flows work.

The WARM Wearable device itself was simulated with an emulator running on the laptop which served as the user’s PC. Physically creating the wearable device was not in the scope for this thesis. But, to demonstrate the functionality, an emulator which most closely matched the actual proposed device was used. The emulator communicates with the database in the same way as if it was the actual device. This was sufficient for demonstrating functionality of the application and for discovering other functionality and features which should be included in future iterations.

6.2 Simulation Setup

In order to create a simulation that is as realistic as possible, some setup work needed to be done in the databases. When a user signs up for the WARM service, the WARMReference database already has rows of data about thousands of medications. For the purpose of this simulation, these reference tables were pre-loaded with data to be able to test the performance of the stored procedures. But, only 10 of these rows actually represented real medications and their associated information and image.
Also, the main user was created in advance. This is actually how it would work on the real system. When a user purchases the WARM service, the main user is setup when the contract is signed. Their preferences and contacts are setup during their first login.
6.3 Simulation Scenarios

The goal of this simulation is to verify the functionality and integration of the WARM software components and to quantify the performance of the stored procedures. Stored procedures were chosen for the database access because they are precompiled and the additional overhead associated with each dynamic SQL call is thereby reduced. But, stored procedures must be tuned more than dynamic SQL because the query plan is determined with the database in a static state. When that state changes, the plan which was created can become inefficient and the queries run far longer than expected.

Table 6.1 lists the scenarios run to verify functionality. These scenarios may seem quite basic, but they actually brought to light some design flaws in the WARMClient database and instances where information in the GUI was not clear.

<table>
<thead>
<tr>
<th>Function</th>
<th>Component</th>
<th>Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Contact</td>
<td>all access rights</td>
</tr>
<tr>
<td>Add</td>
<td>Contact</td>
<td>limited access rights</td>
</tr>
<tr>
<td>Add</td>
<td>General Reminder</td>
<td>one time only</td>
</tr>
<tr>
<td>Add</td>
<td>General Reminder</td>
<td>Recurring</td>
</tr>
<tr>
<td>Add</td>
<td>Medication Reminder</td>
<td>additional reminder suggested, user does not add</td>
</tr>
<tr>
<td>Add</td>
<td>Medication Reminder</td>
<td>Additional reminder suggested, user adds</td>
</tr>
<tr>
<td>Add</td>
<td>Medication Reminder</td>
<td>no additional reminder</td>
</tr>
<tr>
<td>Modify</td>
<td>Contact</td>
<td>remove access</td>
</tr>
<tr>
<td>Modify</td>
<td>Contact</td>
<td>add access</td>
</tr>
<tr>
<td>Modify</td>
<td>General Reminder</td>
<td>change frequency</td>
</tr>
<tr>
<td>Modify</td>
<td>Medication Reminder</td>
<td>change end date</td>
</tr>
</tbody>
</table>
The process to run and verify each scenario was as follows:

- Perform the task using the WARM web application front-end
- Run a command like query against the tables involved to make sure the desired action occurred.
- Use the front-end once again to make sure the changes seen in the tables are communicated into the GUI for the end-user to see’

After each scenario was successfully run and the issues fixed, the functionality of all the stored procedures within the WARM web application had been verified.

The second half of this simulation focused on the process flow and performance of the WARM application. The scenarios run for this part of the simulation didn’t involve the addition and modification of contacts. This suite of cases dealt with the addition of both types of reminders for varying frequencies and the reaction of the system to clearing, postponing and not clearing them. As stated before, the communication to the wearable device was simulated with database tables instead of Bluetooth networking. This allowed for each action to be verified and timings to be reported without the network performance variable.

Ten reminders of varying type, range and frequency were entered using the WARM web interface. For two of these reminders, the user was asked if they would like to add an additional reminder about eating food prior to their medication. In both cases, the user accepted. This brought the total number of simulation scenarios to 12. An export of the raw data from the WARMClient..reminderSpecs table is shown in Table 6.2.
Table 6.2 Data from WARMClient..reminderSpec

<table>
<thead>
<tr>
<th>specKey</th>
<th>typeKey</th>
<th>Description</th>
<th>startDate</th>
<th>endDate</th>
<th>inKey</th>
<th>manKey</th>
<th>reqKey</th>
<th>inMd</th>
<th>alertCls</th>
</tr>
</thead>
<tbody>
<tr>
<td>510</td>
<td>400</td>
<td>S1 - General CTO 6PM</td>
<td>03/24/07</td>
<td>03/24/07</td>
<td>18</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>520</td>
<td>400</td>
<td>S2 - General CTO 7:30PM</td>
<td>03/24/07</td>
<td>03/24/07</td>
<td>19</td>
<td>30</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>530</td>
<td>400</td>
<td>S3 - General 2X 5:30PM</td>
<td>03/24/07</td>
<td>03/25/07</td>
<td>17</td>
<td>30</td>
<td>12</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>540</td>
<td>400</td>
<td>S4 - General 2X 7:30PM</td>
<td>03/24/07</td>
<td>03/24/07</td>
<td>19</td>
<td>30</td>
<td>12</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>550</td>
<td>399</td>
<td>S6 - Medical 1X 3 days 8PM</td>
<td>03/24/07</td>
<td>03/26/07</td>
<td>20</td>
<td>0</td>
<td>24</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>560</td>
<td>399</td>
<td>S6 - Medical 2X 2 days 7:30PM</td>
<td>03/24/07</td>
<td>03/24/07</td>
<td>19</td>
<td>30</td>
<td>12</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>570</td>
<td>399</td>
<td>S7 - Medical 2X 2 days 6PM + rem</td>
<td>03/24/07</td>
<td>03/24/07</td>
<td>18</td>
<td>0</td>
<td>8</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>580</td>
<td>470</td>
<td>S7 - Medical 3X 2 days 6PM + rem</td>
<td>03/24/07</td>
<td>03/25/07</td>
<td>17</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>590</td>
<td>440</td>
<td>S8 - General CTO 8:30AM</td>
<td>03/25/07</td>
<td>03/25/07</td>
<td>8</td>
<td>30</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>600</td>
<td>430</td>
<td>S9 - General 4X 9AM</td>
<td>03/25/07</td>
<td>03/25/07</td>
<td>9</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>610</td>
<td>399</td>
<td>S10 - Medical 3X 9AM + rem</td>
<td>03/25/07</td>
<td>03/25/07</td>
<td>9</td>
<td>30</td>
<td>8</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>620</td>
<td>470</td>
<td>S10 - Medical 3X 9AM + rem</td>
<td>03/25/07</td>
<td>03/25/07</td>
<td>9</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

The details about each alert such as number to send before alerting contacts, are stored in other tables in the WARMClient database and can be retrieved by joining to them using the specKey and typeKey. For this simulation, the description column was used to describe the scenario. In actual use, this is where the user would store information pertinent to the reminder.

Once the user enters and confirms their reminder through the WARM web interface, the scheduler determines what time it will run for the range of dates chosen. For example, if the user enters a reminder for a range of two days and requests that they be notified twice on each day starting at 8:00 AM, the scheduler will update the WARMActivity..reminderSchedule table with two rows. One row will be for the 8:00 AM reminder and the second for the 8:00 PM reminder. The date range of the reminder is represented by the startDate and endDate columns. Table 6.3 shows the raw data from this table after all scenarios for the simulation were entered into the system. The update to this table happens after each reminder is entered; it is not a batch process. Once the endDate has past, a batch job moves those rows to an archive table for future reporting.
Table 6.3 Data from WARMActivity..reminderSchedule

<table>
<thead>
<tr>
<th>specKey</th>
<th>startDate</th>
<th>endDate</th>
<th>hrKey</th>
<th>mnKey</th>
<th>retries</th>
</tr>
</thead>
<tbody>
<tr>
<td>510</td>
<td>3/24/07</td>
<td>3/24/07</td>
<td>18</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>520</td>
<td>3/24/07</td>
<td>3/24/07</td>
<td>19</td>
<td>30</td>
<td>2</td>
</tr>
<tr>
<td>530</td>
<td>3/24/07</td>
<td>3/25/07</td>
<td>17</td>
<td>30</td>
<td>2</td>
</tr>
<tr>
<td>530</td>
<td>3/24/07</td>
<td>3/25/07</td>
<td>5</td>
<td>30</td>
<td>2</td>
</tr>
<tr>
<td>540</td>
<td>3/24/07</td>
<td>3/24/07</td>
<td>19</td>
<td>30</td>
<td>2</td>
</tr>
<tr>
<td>540</td>
<td>3/24/07</td>
<td>3/24/07</td>
<td>7</td>
<td>30</td>
<td>2</td>
</tr>
<tr>
<td>550</td>
<td>3/24/07</td>
<td>3/26/07</td>
<td>20</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>560</td>
<td>3/24/07</td>
<td>3/25/07</td>
<td>19</td>
<td>30</td>
<td>2</td>
</tr>
<tr>
<td>560</td>
<td>3/24/07</td>
<td>3/25/07</td>
<td>7</td>
<td>35</td>
<td>2</td>
</tr>
<tr>
<td>570</td>
<td>3/24/07</td>
<td>3/25/07</td>
<td>18</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>570</td>
<td>3/24/07</td>
<td>3/25/07</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>570</td>
<td>3/24/07</td>
<td>3/25/07</td>
<td>10</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>580</td>
<td>3/24/07</td>
<td>3/25/07</td>
<td>17</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>580</td>
<td>3/24/07</td>
<td>3/25/07</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>580</td>
<td>3/24/07</td>
<td>3/25/07</td>
<td>9</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>590</td>
<td>3/25/07</td>
<td>3/25/07</td>
<td>8</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>600</td>
<td>3/25/07</td>
<td>3/25/07</td>
<td>9</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>600</td>
<td>3/25/07</td>
<td>3/25/07</td>
<td>15</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>600</td>
<td>3/25/07</td>
<td>3/25/07</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>600</td>
<td>3/25/07</td>
<td>3/25/07</td>
<td>21</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>610</td>
<td>3/25/07</td>
<td>3/25/07</td>
<td>9</td>
<td>30</td>
<td>2</td>
</tr>
<tr>
<td>610</td>
<td>3/25/07</td>
<td>3/25/07</td>
<td>17</td>
<td>30</td>
<td>2</td>
</tr>
<tr>
<td>610</td>
<td>3/25/07</td>
<td>3/25/07</td>
<td>1</td>
<td>30</td>
<td>2</td>
</tr>
<tr>
<td>620</td>
<td>3/25/07</td>
<td>3/25/07</td>
<td>9</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>620</td>
<td>3/25/07</td>
<td>3/25/07</td>
<td>17</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>620</td>
<td>3/25/07</td>
<td>3/25/07</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Once a reminder has corresponding rows in the reminderSchedule table, the synchronization programs running through Windows scheduler control the notification to the WARM wearable device and any necessary updates. In WARM’s current design, the start time for a reminder can only be on the hour or half hour. So, every 30 minutes, the reminderSchedule table is queried to see if any reminders are ready to run. If any meet the criteria, a row is inserted into the WARMActivity..reminderInProgress table. At the same time, a row is inserted into the WARMActivity..sendToWearable table. This is the table the WARM wearable polls to determine if there are new reminders for the user. For
purposes of this simulation, the “consumed” column of sendToWearable is set to 1 as soon at the row is inserted. In actual use, this column has the value of 0 until the wearable device reads the row and sends an update back to the database. If there is a communication problem with the wearable device and it cannot communicate with the database, these rows will persist until communication is restored. Table 6.4 shows the entries in reminderInProgress and sendToWearable at one point during the simulation.

Table 6.4 Sample Entries from reminderInProgress and sendToWearable

<table>
<thead>
<tr>
<th>specKey</th>
<th>sendAttempt</th>
<th>timeSent</th>
<th>cleared</th>
<th>sentAlert</th>
<th>timeCleared</th>
<th>retriesAllowed</th>
</tr>
</thead>
<tbody>
<tr>
<td>570</td>
<td>2</td>
<td>3/25/07 10:00 AM</td>
<td>1</td>
<td>0</td>
<td>3/25/07 10:12 AM</td>
<td>2</td>
</tr>
</tbody>
</table>

When a user receives a reminder, they have three options. First, they can press the OK button. This implies that they’ve received and read the reminder and have taken the action required. The cleared column in reminderInProgress table gets updated with the value of 1 and timeCleared gets set to the current time. Second, they can press LATER. This will send the reminder again in 10 minutes. This is handled by decreasing by one the value in the sendAttempt column in the reminder. Finally, the user can take no action. This also will send the reminder again in 10 minutes, but the sendAttempt value is not deceased. If the alertContacts flag is turned on for this reminder, contacts will be alerted if no action is taken after the number of allowed retries has been exceeded.

The second piece of the synchronization program handles the reminder once it has been sent to the wearable. It wakes up every 10 minutes to check the reminderInProgress table for updates and to determine when to alert contacts and when to move the record to the reminderLog table. This portion of the synchronization is described in table 6.5.
Table 6.5 Pseudocode for Synchronization

<table>
<thead>
<tr>
<th>if reminder(cleared) = TRUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>insert into WARMActivity..reminderLog (specKey, sentDateTime, clearedDateTime, notifySent)</td>
</tr>
<tr>
<td>remove row from WARMActivity..reminderInProgress</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>if reminder(cleared) = FALSE and reminder(sendAttempt) &gt;= retriesAllowed</th>
</tr>
</thead>
<tbody>
<tr>
<td>update WARMActivity..reminderInProgress</td>
</tr>
<tr>
<td>set sentAlert = 1, cleared = 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>if reminder(cleared) = FALSE and reminder(sendAttempt) &lt; retriesAllowed</th>
</tr>
</thead>
<tbody>
<tr>
<td>update WARMActivity..reminderInProgress</td>
</tr>
<tr>
<td>set sendAttempt = (sendAttempt + 1)</td>
</tr>
</tbody>
</table>

The life of an instance of a reminder is complete when it is moved to the WARMActivity..reminderLog table. As soon as the number of allowed retries is reached, the reminder moves from the reminderInProgress table to the reminderLog table. Not all uncleared reminders trigger a call to emergency contacts. Medical reminders default to alert contacts after two tries. General reminders default to not alert contacts. If the user does not change the defaults, a general reminder will not send an alert, but it will still be moved to the reminderLog table. Table 6.6 shows the reminderLog table during the simulation. The timeComplete column represents either the time that a notification was sent to contacts or the time of the final retry. When this table is queried to run an activity report, the timeComplete column must be coupled with the notifySent column to determine if contacts needed to be alerted.
Table 6.6 Data from WARMActivity..reminderLog During Simulation

<table>
<thead>
<tr>
<th>specKey</th>
<th>timeSet</th>
<th>timeComplete</th>
<th>notifySent</th>
</tr>
</thead>
<tbody>
<tr>
<td>530</td>
<td>3/24/07 5:30 PM</td>
<td>3/24/07 5:31 PM</td>
<td>0</td>
</tr>
<tr>
<td>510</td>
<td>3/24/07 6:00 PM</td>
<td>3/24/07 6:08 PM</td>
<td>0</td>
</tr>
<tr>
<td>560</td>
<td>3/25/07 7:35 AM</td>
<td>3/25/07 7:46 AM</td>
<td>0</td>
</tr>
<tr>
<td>590</td>
<td>3/25/07 8:30 AM</td>
<td>3/25/07 8:32 AM</td>
<td>0</td>
</tr>
<tr>
<td>600</td>
<td>3/25/07 9:00 AM</td>
<td>3/25/07 9:02 AM</td>
<td>0</td>
</tr>
<tr>
<td>610</td>
<td>3/25/07 9:30 AM</td>
<td>3/25/07 10:00 AM</td>
<td>1</td>
</tr>
<tr>
<td>570</td>
<td>3/25/07 10:00 AM</td>
<td>3/25/07 10:12 AM</td>
<td>0</td>
</tr>
<tr>
<td>600</td>
<td>3/25/07 3:00 PM</td>
<td>3/25/07 3:20 PM</td>
<td>1</td>
</tr>
<tr>
<td>580</td>
<td>3/25/07 9:00 AM</td>
<td>3/25/07 9:01 AM</td>
<td>0</td>
</tr>
<tr>
<td>620</td>
<td>3/25/07 9:00 AM</td>
<td>3/25/07 9:01 AM</td>
<td>0</td>
</tr>
<tr>
<td>570</td>
<td>3/24/07 6:00 PM</td>
<td>3/24/07 6:20 PM</td>
<td>1</td>
</tr>
<tr>
<td>560</td>
<td>3/24/07 7:30 PM</td>
<td>3/24/07 7:34 PM</td>
<td>0</td>
</tr>
<tr>
<td>540</td>
<td>3/24/07 7:30 PM</td>
<td>3/24/07 7:40 PM</td>
<td>0</td>
</tr>
<tr>
<td>520</td>
<td>3/24/07 7:30 PM</td>
<td>3/24/07 8:00 PM</td>
<td>1</td>
</tr>
<tr>
<td>550</td>
<td>3/24/07 8:00 PM</td>
<td>3/24/07 8:04 PM</td>
<td>0</td>
</tr>
</tbody>
</table>

For the purposes of this simulation, a script was manually run to make some reminders as cleared. This tested the functionality of the synchronization to move reminders to the reminderLog table at the proper time.

In addition to functional verification of the proposed WARM software architecture, another purpose of this simulation was to gather performance statistics. To do this, each time a portion of the synchronization program was called, a record was written out to a file listing the name of the procedure and the start and end times. In order to create an environment that closely resembles what it would be in a real-life situation, rows of data were bulk copied into the tables before the simulation.

This simulation has been run multiple times throughout the verification stage of this thesis. In the first run, the performance of the all database access procedures, even those executed from the WARM web application, was unacceptable. As a result of this, the database schema was completely reworked. The use of integer keys to identify reminders was added and some columns were repeated in multiple tables to limit the
number of participants necessary in some join clauses (see Figure 5.2 for the final model). These changes help bring the performance data into an acceptable range, but as the simulation continued to run with the tables being of near-production size the performance began to degrade.

Batch jobs were put in place to run nightly to clean up the in progress tables. In the original design, data was copied into the reminderLog table, but never removed from the in progress tables. These batch jobs in addition to the database redesign corrected the performance issues. Even with a full database, all the WARM procedures run in sub-second time.
CHAPTER SEVEN
FUTURE WORK

Applications such as WARM have the potential to greatly change and enhance the lives of their users. WARM just scratches the surface of the functionality a wearable device can contain. The basic architecture of WARM was designed with scalability in mind. Because a large portion of the back-end processing is table driven, different front-end and networks can be “plugged in” as technology advances. Also, it lends itself to be able to include more functionality without reworking any of the low level components.

One useful addition to the WARM application would be the inclusion of a commercial web service to use in place of the WARMReference database. In WARM’s current design, the WARMReference database houses information about medications which is used to fill in data on the medical reminder entry screen and the actual reminder screen displayed on the wearable device. While this approach is sufficient, a web service would be far more robust and lead to more accurate information.

There are quite a few commercial web services that can provide medical information and pictures to applications. These take either a full or partial medication name and return the requested information. The application can consume the information in whichever way makes sense based on the technology. For WARM, the data would be saved to an object stored in the HTTP session and once the user confirms that the
reminder is complete, it would go to the WARMClient database in the medicalReminderDetail table.

Another area of WARM which can be improved upon is customization. Users can customize their reminders and give them detailed and meaningful descriptions. They can set the frequency, start time and what actions they’d like to occur on the two buttons. But, through developing and working with the simulation version of WARM, it became apparent that there are other details that a user may like to customize.

One approach to customization would be to create an administrative tool, possibly a MyWARM section, to allow for additional customization that can be applied to the reminders. These options, such as custom groups or icons, would be stored in a custom database and would be returned in the selection drop downs on the reminder setup pages.

Finally, even though this paper offers a software solution, the WARM wearable itself could offer some more options for ease of use. WARM is worn as a watch. This was because most people wear a watch everyday. But, in working with the emulator, the device may not be something everyone would want to wear. As discussed before, the WARM architecture lends itself to easily use a different wearable device. The device need only communicate with the user’s PC to get its updates and have a means for the alerts to be cleared. Possible, a device similar to a pager may be more desirable for some users. Others may like the watch-like version.

This goal of this thesis was to propose a solution for a device to assist its users in daily tasks and to provide their families with a level of confidence that their loved one is taking care of themselves. This architecture accomplishes that goal. It offers an easy to use and understandable interface for reminder entry, clearing and reporting. And, because
of the table driven approach, this architecture can easily be extended to include new features as they are developed. As research continues in this area, this architecture can be the basis for future wearable devices.
REFERENCES


