SYSTEM DESIGN OF AN ACTIVITY TRACKER TO ENCOURAGE BEHAVIORAL CHANGE AMONG THOSE AT RISK OF PRESSURE ULCERS

A Thesis
Presented to
The Academic Faculty

By

John J OBrien

In Partial Fulfillment
Of the Requirements for the Degree
Master of Science in Bioengineering

Georgia Institute of Technology

May, 2019

SYSTEM DESIGN OF AN ACTIVITY TRACKER TO ENCOURAGE BEHAVIORAL CHANGE AMONG THOSE AT RISK OF PRESSURE ULCERS

Approved By:

Dr. Stephen Sprigle, Advisor
School of Industrial Design
Georgia Institute of Technology

Dr. Sharon Sonenblum
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Thomas Ploetz
College of Computing
Georgia Institute of Technology

Date Approved: April 22, 2019

DEDICATION

Dedicated to my fiancée, Katelyn Sophia Ann Sturdivant

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Stephen Sprigle, for his guidance during the time that I spent working on my thesis in the Rehabilitation Engineering and Applied Research (REAR) lab. Under his guidance, I learned a great deal about design, engineering, and project management. Without him, I would not have been able to complete this thesis.

Second, I would like to thank Dr. Sharon Sonenblum for her ongoing work on WiSAT and the help she provided me along the way. She shared her immense knowledge of subjects related to WiSAT, especially those related to the hardware and algorithm subsystems. She was essential to my ability to complete Specific Aim 3.

I would also like to thank Dr. Thomas Ploetz for his time serving on my thesis committee, proofreading drafts of my thesis, and for the lessons he taught as part of his Mobile and Ubiquitous Computing course.

I owe a tremendous amount of thanks to my fellow students and colleagues in the REAR lab for their help with WiSAT and for keeping morale up during my time in the lab. In particular, I would like to thank members (past and present) of the WiSAT team: Chris Hanes, Katie Jordan, Yogesh Deshpande, and Ashley Andrews. Additionally, I owe a special thanks to Michael Aki for his work translating algorithms into Java and Swift, as well as Ji-Won Jung for her work as a tester, and Nauman Ahad for his tireless algorithm work and staying late to assist me with the work covered in Specific Aim 3. I would like to thank XiuXiu Yuan for her patience in teaching me Adobe XD and for her vital contribution to the design of the app. I would like to thank the students who assisted with the “real-time” app: Victor Zhu, Aditya Retnanto, Aelita Shen, and

Jordan Ray. I thank Christina Anderson for her assistance with the line drawings in the app. Also, I owe a special thanks to Jacob Misch for his general helpfulness around the lab.

Additionally, I would like to thank Quincy Williams of DataFinch Technologies and Enrich Trait for the advice and lessons he shared based on his years of experience as a software engineer.

Our industry and academic partners were especially valuable to the completion of this project. In particular: Mike Crowe and Alex Kooney of Gulf Coast Data Concepts, Ashok Amaran of App Chefs, Ankith Patel of Apeiro Technology, Yair Bar-On of TestFairy, and Dr. Bambang Parmanto and Dr. Andi Saptono of the HARI Lab at the University of Pittsburgh.

Many others helped in a variety of ways throughout the duration of this project: Agrim Chandra, Becky Yao, Chris Bartlett, Drew Keller, Jayanth Mohana Krishna (API Advice), Jun Kim, Laura Paige, Noah Posner, Shane Owens, Skyler Canute, and the staff of the REAR Lab.

Finally, I would like to thank my family, especially my parents, Ginny and John OBrien, for being supportive of me while I pursued my graduate degree. I would like to thank my Aunt Mar for letting me live with her for several months while I was at Georgia Tech. Last, I would like to thank my fiancée, Katelyn, for her support through a long-distance relationship while I was in school.

This work was supported by the Office of the Assistant Secretary of Defense for Health Affairs, through the Spinal Cord Injury Research Program under Award No. W81XWH-17-1-0221 and the National Institute of Aging of the National Institutes of Health under Award Number 1R01AG056255. Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the Department of Defense or National Institutes of Health.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

API: Application Program Interface

App: Smartphone Application

BCT: Behavioral Change Technique

BCTTv1: Behavioral Change Technique Taxonomy

HBCT: Health Behavioral Change Theory

ISO: International Organization for Standardization

JITAI: Just-In-Time Adaptive Intervention

mAh: milliamp hour

NIST: National Institute of Standards and Technology

SQL: A programming language used to retrieve, create, update, and delete information in a database

UI: User-Interface

WISAT: Wheelchair In-Seat Activity Tracker

SUMMARY

The Wheelchair In-Seat Activity Tracker (WiSAT) is a sensor-based activity tracker aimed at encouraging in-seat movement among wheelchair users who are at risk of pressure ulcers. Pressure ulcers tend to form in the buttocks or thighs of a wheelchair user due to a lack of pressure redistribution in that part of the body. Pressure ulcers are a serious risk to many wheelchair users due to a plethora of harmful side-effects, such as infection, hospitalization, and long recovery times. However, in-seat movements, such as weight shifts, have been linked with the occurrence of pressure ulcers.

WiSAT began as a research tool that enabled researchers to monitor the in-seat activity of wheelchair users during their daily lives through sensor-based reporting, as opposed to relying solely on the self-reporting of research participants. Through the efforts described in this thesis, WiSAT was transformed from a research tool into a consumer product. Specifically, this thesis describes the design, development, and integration of WiSAT's subsystem through four specific aims:

1. Design and evaluation of a user-interface based upon principles of Health Behavioral Change Theory.
2. Coupling of the Hardware and Smartphone App Subsystems
3. Preparation of the Algorithm Subsystem
4. Integration of the WiSAT Subsystems through Multi-layered Architecture for the WiSAT Smartphone App

CHAPTER 1. INTRODUCTION

Pressure ulcers pose a serious threat to the well-being of wheelchair users. The National Pressure Ulcer Advisory Panel defines pressure ulcers as “localized injury to the skin and/or underlying tissue usually over a bony prominence, as a result of pressure, or pressure in combination with shear”[1]. Because of their reduced mobility and, in many cases, impaired sensation, many wheelchair users remain susceptible to pressure ulcers throughout their lives. Although pressure ulcers range in severity, more serious pressure ulcers can take 6-18 months to heal [2], may be accompanied by infection, and may require the surgical removal of damaged tissue [1, 3]. Even in cases where surgery is not required, pressure ulcers may be treated with several weeks to months of bed rest, which often places a large financial and psychological burden on the individual due to treatment costs and lost productivity [4, 5]. One study found that within the first year of experiencing a spinal cord injury, 41% of individuals experienced a pressure ulcer [6]. Additionally, once a person develops their first pressure ulcer, they are at an increased risk of additional ulcers forming in the future[7].

Research indicates in-seat maneuvers known as pressure reliefs and weight shifts are effective in relieving pressure and increasing blood flow in the buttocks [8, 9]. Pressure reliefs are typically longer movements aimed at completely removing pressure from the ischial tuberosities. Weight shifts are any in-seat movement, including functional movements, which redistributes weight across the buttocks for several seconds or more. Since both pressure relief and blood flow are associated with pressure ulcer prevention [10], wheelchair users who are able to perform these maneuvers independently are encouraged to do so several times per hour [1, 10]. Unfortunately, evidence suggests that many wheelchair users fail to perform pressure reliefs and weight shifts consistently enough to lower their risk of developing a pressure ulcer [11].

Researchers and clinicians suggest that feedback on pressure ulcer prevention efforts encourages individuals to engage in behaviors that reduce their risk of pressure ulcers [12-16]. Previous studies, which relied upon self-reporting, indicated no link between pressure reliefs or weight shifts and pressure ulcers [17-19]. However, recent studies used sensor-based technology to monitor pressure relief and weight shift activity during study participants' daily lives. These studies indicated that (1) individuals with prior pressure ulcers perform fewer weight shifts than those with no prior history, (2) in general, none of the study participants performed an adequate number of pressure reliefs or weight shifts, and (3) that there is a link between weight shifts and the occurrence of pressure ulcers [9, 20]. The Wheelchair in-Seat Activity Tracker (WiSAT) was developed as an activity tracker system that builds upon the technology and findings of these previous studies to monitor and inform users about their in-seat activity.

The WiSAT system consists of four subsystems (Figure 1). First, a **hardware subsystem** contains a sensor mat and data logger. The sensor mat contains six force sensors and is positioned on the underside of the wheelchair user's cushion. The data logger module collects data from the pressure mat at 4 Hertz. Second, a companion **smartphone application (app)** receives the sensor data from the data logger via Bluetooth Low Energy technology. Third, **classification algorithms** in the app interpret the sensor data to determine when various in-seat activities occur, including weight shifts, and times when the user is out of their chair. The WiSAT app displays this information to the user in the context of user-defined goals. Goals are set for three variables: number of weight shifts, time between weight shifts, and an In-Seat Movement score. The in-seat movement score is a metric intended to measure "fidgeting", movements that are shorter in duration than weight shifts but still redistribute pressure off the buttocks and thighs. The last subsystem, **a remote server**, collects and archives aggregated WiSAT data through a series of RESTful API endpoints.

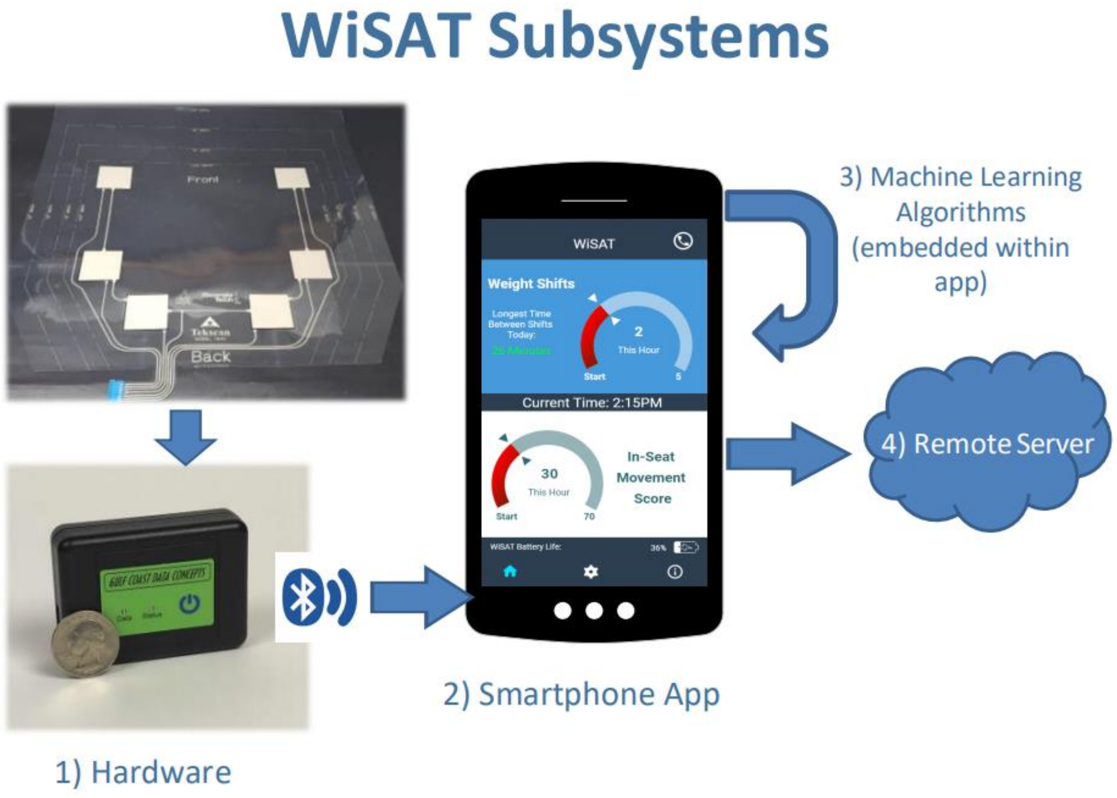


Figure 1: WiSAT Subsystems

Early iterations of WiSAT were designed for research purposes and were not yet ready for consumer use. The objective of this thesis was to create the next iteration of WiSAT, a consumer product built upon the technology and lessons learned from prior research. As a consumer product, WiSAT was designed to be used directly by the end-user, without any necessary intervention or guidance from healthcare experts. This objective was accomplished through four specific aims:

- **Specific Aim 1: Design and evaluation a user-interface based upon principles of Health Behavioral Change Theory.**
- **Specific Aim 2: Coupling of the Hardware and Smartphone App Subsystems**
- **Specific Aim 3: Preparation of Algorithm Subsystem**
- **Specific Aim 4: Integration of the WiSAT Subsystems through Multi-layered Architecture for the WiSAT Smartphone App**

CHAPTER 2. BACKGROUND

Decades before the introduction of mobile computing, Carr and Wilson [21] examined the effectiveness of electronic monitoring and notification systems upon wheelchair users' pressure ulcer prevention efforts. They suggested that wheelchair users often fail to comply with their prevention regimens because of the lack of feedback accompanying a single relief. They found that the use of their electronic notification system is associated with increased compliance and hypothesized that this is due to the frequent feedback that is provided to the user based on their compliance (or lack thereof) with their regimen.

With the proliferation of mobile computing, new technologies have arisen to encourage health and wellness by monitoring and communicating user behavior. For example, multiple systems utilize a mobile, sensor-based approach to monitor and promote physical fitness [22-26]. Likewise, many mobile applications exist to help patients manage chronic illnesses, such as diabetes [27, 28], cardiovascular[28], and respiratory disease [28]. Other applications seek to build compliance among patients with regards to a medication regimen [28-30]. As with Carr and Wilson's research, many of these systems utilize sensor data to provide regular feedback to the user and encourage specific behaviors.

In the domain of pressure ulcer prevention, past work highlights the use of sensor data to determine information regarding in-seat activity. With the help of machine learning algorithms, it is possible to determine long-term, in-seat activity with a pressure sensing mat placed under wheelchair cushions [31]. This thesis builds upon previous hardware developed to monitor in-seat activity in a research setting [9, 11] and combines it with a smartphone app.

At least one similar system has attempted to monitor in-seat-activity with pressure sensors and provide feedback to the user through a mobile application [32]. However, WiSAT differs from previous work in that it is built with a strong focus on Health Behavioral Change Theory (HBCT) as a means to encourage users to comply with their prevention regimens.

2.1 Health Behavioral Change Theory

In the context of this thesis, HBCT is an “umbrella term” which encompasses several theories that attempt to explain the psychological mechanisms responsible for promoting health and wellness through the reduction of harmful behaviors and the encouragement of beneficial ones. Although it has many applications, behavioral change is especially important to the management of chronic health conditions [33]. Social Cognitive Theory, the Health Belief Model, and the Transtheoretical Model of Change are among the leading theories in this field [34, 35].

Concepts from all three theories can be applied to pressure ulcer prevention. As noted by Prochaska and Velicer, “No single theory can account for all of the complexities of behavior change” [36]. Researchers summarize the commonalities between the three theories as: (1) goal setting, (2) social support, (3) rewarding beneficial behaviors, (4) maintenance, and (5) prevention of relapse [35]. Behavioral interventions based on these theories often involve many components including those related to the patient’s social, environmental, and community surroundings [37].

2.2 Health Behavioral Change Theory Frameworks

WiSAT was designed and evaluated based on principles that derive from the theories described above. While there is not yet a universal standard to evaluate mobile HBCT systems, there are some proposed frameworks in the literature. As seen in work by Wang, Fadhil, Lange, and Reiterer (2017), the best approach to design and evaluate an HBCT may be a combination of several frameworks [38].

Many of the UI features of the WiSAT app derive from Behavioral Change Techniques (BCTs), which are individual components of a behavior intervention [39]. At least three related taxonomies exist to classify BCTs in mobile activity trackers [40, 41]. The most recent of these taxonomies is known as Behavioral Change Technique Taxonomy v1 (BCTTv1) [39]. It is composed of 93 BCTs divided into 16 categories. A study by Lyons, Lewis, Mayrsohn, and Rowland (2014) examined the prevalence of these BCTs in “electronic activity monitors” (mobile applications that track the user’s physical activity) [42]. As acknowledged by the authors, not all BCTs in this taxonomy are relevant for all scenarios. Some BCTs may be more relevant to large-scale, public policy interventions but not as relevant to more individualized interventions and vice-versa [39].

When this taxonomy was used by Lyons, Lewis, Mayrsohn, and Rowland (2014), they identified fourteen BCTs from the BCTTv1 associated with successful interventions in the literature [42] :

Table 1: Lyons' Behavioral Change Techniques associated with Successful Interventions

Behavior Goal Setting	Problem Solving	Outcome Goal Setting
Action Planning	Review of Behavior Goals	Commitment
Feedback on Behavior	Self-monitoring of Behavior	Social Support
Instruction on how to Perform the Behavior	Information about Consequences	Social Comparison
Behavioral Practice/Rehearsal	Rewards	

Of these, they highlight behavior goal setting, review of behavioral goals, feedback on behavior, self-monitoring of behavior, and rewards as especially prevalent among physical activity trackers. Additionally, prior research indicates that users may prefer apps which include BCTs related to goal-setting and self-monitoring over those associated with social support [38, 43]. For a specific use case, relevant BCTs may be identified through the use of determinants, broader concepts such as “self-efficacy” and “self-regulation” that can be linked to specific BCTs such as “instruction on how to perform the behavior” and “goal setting” [44].

2.3 Just-In-Time Adaptive Intervention

Just-In-Time Adaptive Interventions (JITAI) is another framework used in the design the WiSAT app. JITAI encourages the use of context-awareness features of smartphones to dynamically control the behavioral intervention [38]. The SitCoach app is an example of a JITAI system [45]. This mobile application uses the phone’s accelerometer to measure the time an office worker is sedentary and sends reminders to take a break only when a user-specified time threshold of sedentary time has been exceeded [46]. Conversely, a non-JITAI application would be unaware of context and send reminders regardless of whether or not the user was sedentary. The JITAI framework is based upon four components: (1) decision points (when/how often an intervention is deployed), (2) intervention options (what type of intervention is deployed at a decision point), (3) tailoring variables (what interventions are options based on information about the user), and (4) decision rules (the algorithm that links the tailoring variables to the intervention options) [45].

2.4 Usability and Health Numeracy

The success of a mobile health application relies heavily upon its usability and design. Prior work has revealed that users of gamified apps may stop using them if the usability or design is poor [47, 48]. More generally, usability is important to any software system that contains a user-interface (UI) [49]. As defined by Nielsen (1994), the five usability attributes are: learnability, efficiency (the user’s ability to be productive with the system), memorability (the user not having to relearn the system with each use), low error rate, and satisfaction (pleasantness of use) [49]. Similarly, the ISO and NIST evaluate usability through effectiveness, efficiency, and satisfaction [50, 51].

Usability principles are especially important to components of WiSAT that provide the user with feedback regarding their behavioral change. As described by Ryan (2009), systems built upon HBCT should monitor the behavior “proximally” and “distally” [33]. In other words, individual behaviors should be measured (proximal), but the overall progress towards the goal should also be tracked (distal). Additionally, the JITAI framework takes advantage of distal/proximal goals [38]. A JITAI intervention may use various proximal goals to help the user to reach the distal goal, and these proximal goals may inform the creation of the four JITAI components described earlier [45]. In the context of WiSAT, individual goals are proximal measurements, and overall regimen compliance is the distal measurement.

Some creativity is required when addressing ways to communicate the proximal and distal information to the system user. As Nielsen points out, “the reality is that most people do not acquire comprehensive expertise in all parts of the system, no matter how much they use it.” [49] To this end, modern software designers stress the importance of simplicity in application design [52, 53]. This is especially important in data visualizations (graphs, charts, etc.), where it is imperative not to overwhelm the user with unnecessary data [54], as many users have poor “health numeracy” (ability interpret numerical health information) and “graphical literacy” (ability to interpret graphs) skills [55]. In WiSAT, the term “passive notification” is used to describe sections of the user-interface that display graphical feedback on behavior.

In the context of proximal monitoring, prior research describes the benefits of active notifications. As an example, myHealthPal is a popular iOS application that incorporates HBCT. If a user does not meet their goals, the application actively notifies them through a push notification [56]. In another HBCT mobile application, study participants became more active users of their system after introducing a feature to push live notifications to the user’s mobile phone whenever new information was available [57]. For wheelchair users, Apple has introduced new features to their Apple Health application specifically targeted at encouraging users to perform wheelchair-specific exercise and movement [58]. These features include push notifications to remind wheelchair users when it’s time to move [59].

2.5 Technical Design Philosophy

A major theme in the design philosophy of WiSAT could be described as “loosely-coupled”. Loose-coupling helps ensure that WiSAT’s subsystems are as autonomous as possible and that, whenever possible, changes in one subsystem or module would not require a corresponding change in another system. This is especially important when developing multiple subsystems in parallel and deploying updates or implementing features in any subsystem.

The microservices literature succinctly defines the concept of loose-coupling: “By interacting through clearly defined interfaces, or through published events, each microservice remains independent of the internal implementation of its collaborators” [60]. In WiSAT, this concept was materialized through a communication paradigm that utilized well-recognized, straightforward standards. For example, this thesis describes an application program interface (API) that facilitates communication between the app and the hardware subsystems. It was designed in such a way that any device with a Bluetooth module, regardless of hardware or operating system, would be able to request and retrieve data from the logger without requiring any knowledge of the internal state or workings of the logger. Likewise, the logger was designed to not require any knowledge of the implementation of the WiSAT app beyond the information provided to it through the API. This had the effect of creating modular systems that are as resilient as possible to changes, upgrades, and bugfixes in their counterparts.

CHAPTER 3. DEVELOPMENT AND DESIGN PROCESS

The development of WiSAT was an iterative, development and design project rather than a controlled experiment. This process was not perfectly sequential in all cases. Many of the specific aims were conducted in parallel to each other. This research design allowed for the transition of technology that was originally developed for research purposes into a consumer product.

3.1 Specific Aim 1: Design and evaluate WiSAT's user-interface

Table 2: Scope of Specific Aim 1

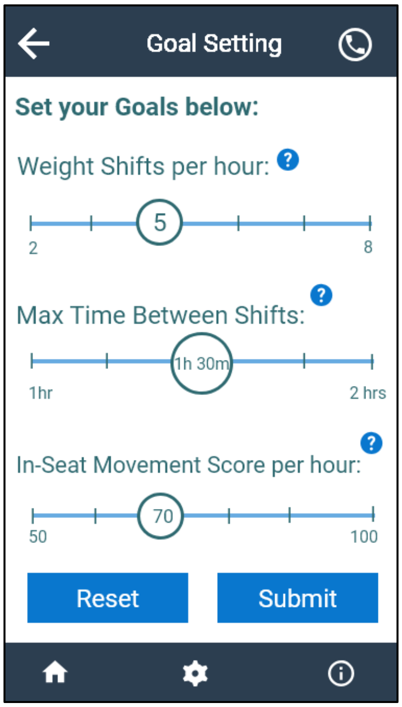
Subsystem	**Prior State of Subsystem**	**Thesis Contribution**	**Outside Scope of Thesis (Third-Party or Future Work)**
App	• Early iterations of UI design	• UI Evaluation • Final UI Design	• Implementation by external vendor

3.1.1 Overview of WiSAT's User-Interface Functionality

As with many other activity trackers, WiSAT was designed to be used without any outside intervention from medical professionals or other specially trained persons. As such, the user-interface design emphasized features which are approachable to a general audience and clearly display information to the end-user.

Upon installing WiSAT, the user first encounters a series of setup and initialization screens (Appendix A). These screens allow the app to capture information for later use in the algorithm subsystem. Specifically, the app prompts the user to perform a series of leans to capture baseline information on their particular sitting behavior. Additionally, the user must enter calibration parameters specific to their hardware system. During this setup, the user is prompted to connect the app to the WiSAT hardware via Bluetooth for the first time.

As part of initial setup, users are required to set their own goals. Consistent with HBCT, the Goal Settings screen was designed to empower users to define their own goals. However, their choices are bounded to a set of discrete values to ensure that goals are both useful and realistic. These bounds were chosen based on past research regarding wheelchair users' typical weight-shift behavior [9, 20]. Users may select between 2 and 8 Weight Shifts per hour (in increments of 1), 50 to 100 In-Seat Movement scores per hour (in increments of 10), and 1 hour to 2 hours maximum time between shifts (in increments of 15 minutes). Users select their goals using sliders. Sliders were chosen to allow the final design to be more accessible to users with limited dexterity.



Goal Setting

Set your Goals below:

Weight Shifts per hour: 5

Max Time Between Shifts: 1h 30m

In-Seat Movement Score per hour: 70

Figure 2: Goal Setting Screen

Following setup, the home screen is the first screen that the user encounters every subsequent time the WiSAT app is launched. The home screen displays information for the past hour related to the three WiSAT metrics (Weight Shift count, time between shifts, and in-seat movement score). Clicking on either the “Weight Shift” or “In-Seat Movement Score” sections allows the user to delve deeper into detail screens that display information on these metrics according to day, week, or month views. Collectively, the home screen and these detail screens constitute WiSAT’s passive notifications.

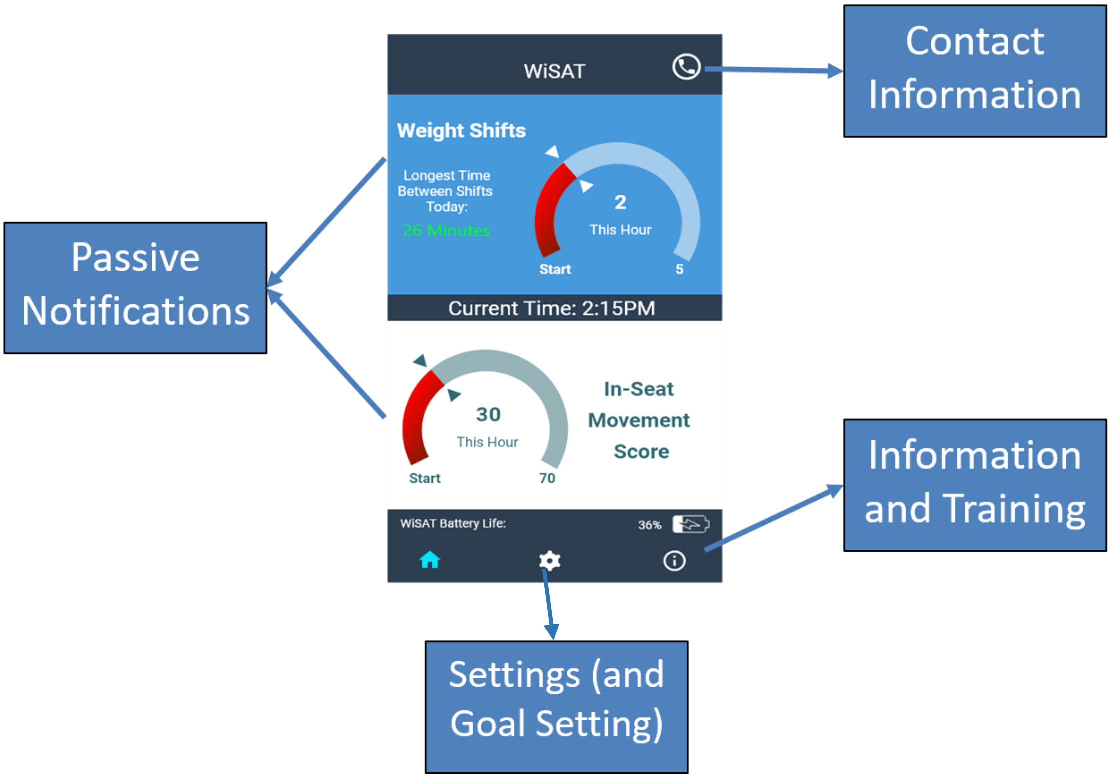
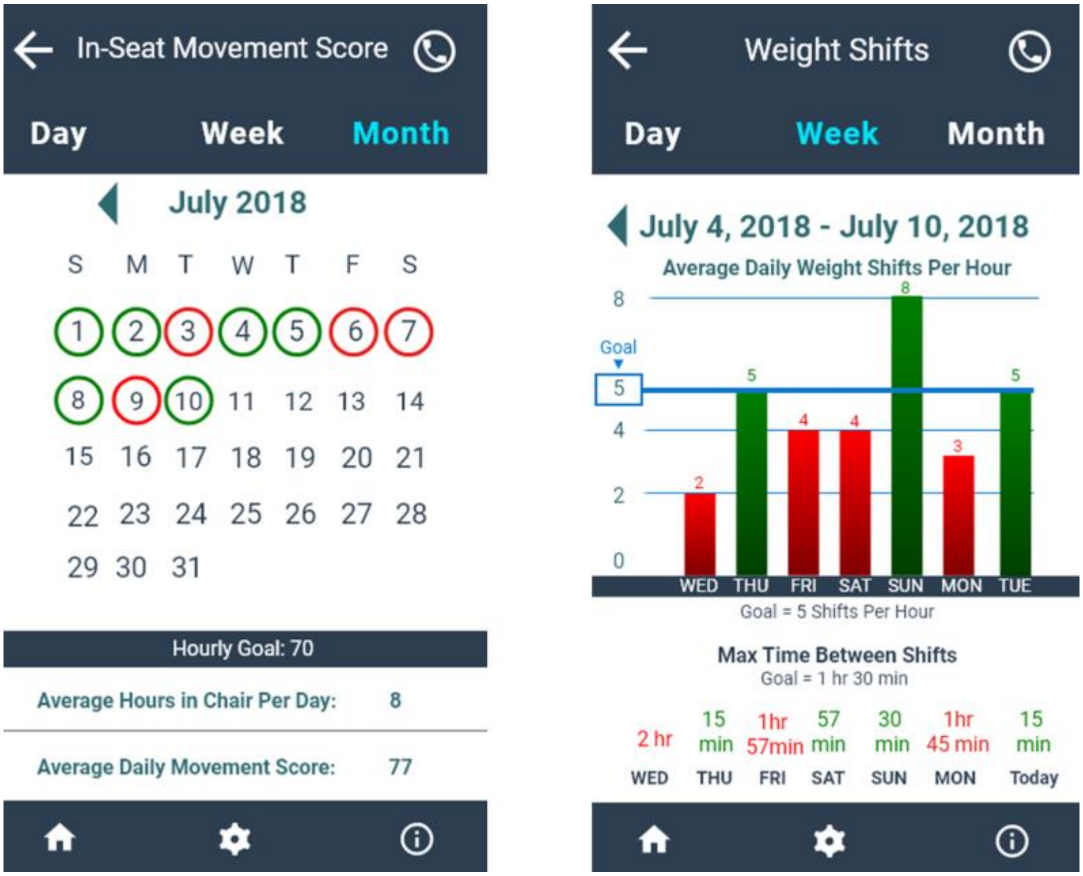


Figure 3: WiSAT Home Screen Sections



Hourly Goal: 70

Average Hours in Chair Per Day: 8

Average Daily Movement Score: 77

Max Time Between Shifts

Goal = 1 hr 30 min

WED	THU	FRI	SAT	SUN	MON	Today
2 hr	15 min	1hr 57min	57 min	30 min	1hr 45 min	15 min

Figure 4: Month and Week Screens

A “footer bar” on most WiSAT user-interface screens provides quick access to the Home, Settings, and Information/Training screens. “Settings” allows the user to change their goals, as well as other system settings. “Information and Training” provides information regarding pressure ulcers, weight shifts, in-seat movement score, and app functionality.

As with many activity tracker apps, WiSAT provides feedback on behavior through push notifications, brief messages displayed on a phone’s lock screen that are often accompanied by a tone or haptic feedback. These notifications constitute WiSAT’s “active notifications”. They provide users the option to review their behavior goals, and suggest future actions. The JITAI framework was useful in the creation of the push notification options for WiSAT. Two of WiSAT’s three variables, Weight Shift Count and In-Seat Movement score, were used as tailoring variables.

3.1.2 Prior Art

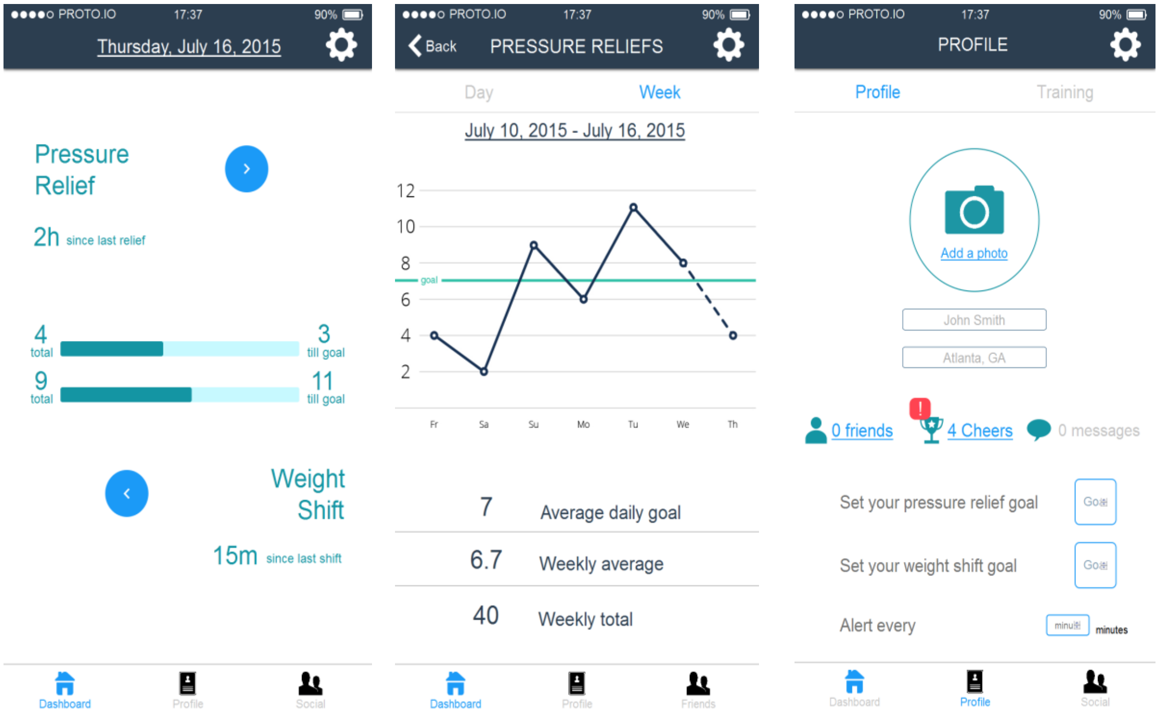


Figure 5: Prototype WiSAT Screens from Cheng, 2015

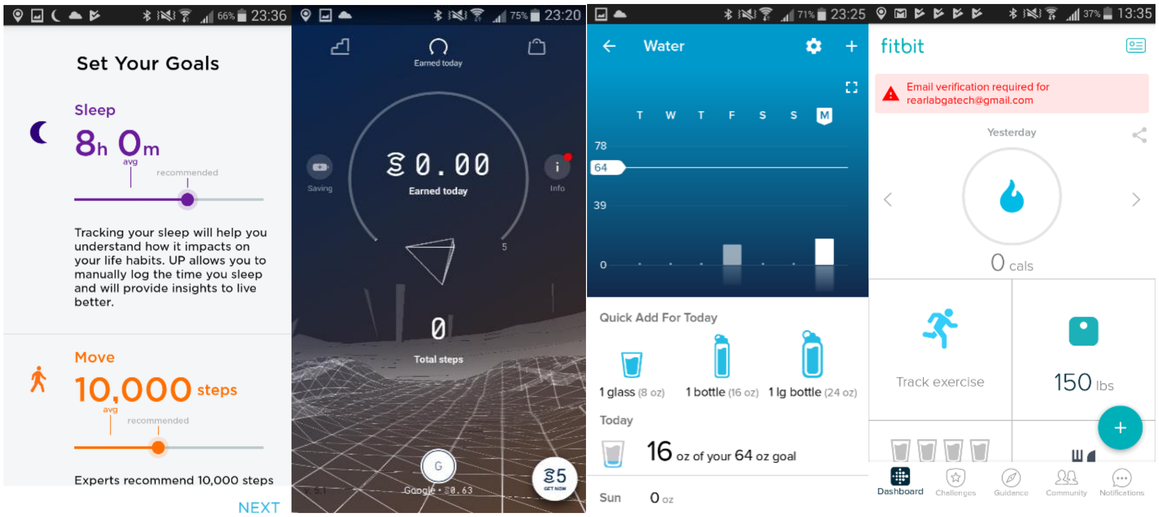
Specific Aim 1 built upon earlier design work, in which an initial WiSAT user-interface was designed for the mobile app. These designs were never developed into a full-fledged app, but existed as a series of prototype user-interface screens. In this prior work, an iterative design process was used, in which members of the target audience were engaged with surveys and targeted interviews [61].

Because several years had elapsed between the creation of the original WiSAT prototype and the current effort, several more recent activity trackers, health, and wellnesses smartphone apps were reviewed. Design elements that accomplished goals similar to those of

WiSAT were noted. For example, many apps included Goal Setting, Training information, Passive Notifications, and Active Notifications. Elements from these designs inspired design features of the new iteration, such as the circular, odometer style graphs and the slider used in the Goal Settings screen. A full list of reviewed apps is in Table 3.

Table 3: Apps Reviewed:

NikeFit	Fabulous	Google Fit
Apple Health	Jawbone Up	Sweatcoin
iRest	Fitbit	Six Pack in 30 Days
Sleep as Android	MyFitnessPal	DiabetesPal
Sensimat	VA Pressure Ulcer/Injury Resource (VAPUR)	Medisafe



Tracking your sleep will help you understand how it impacts on your life habits. UP allows you to manually log the time you sleep and will provide insights to live better.

Experts recommend 10,000 steps

Email verification required for rearlabgatech@gmail.com

Figure 6: Examples of Apps Reviewed: Jawbone Up (Far Left), Sweatcoin (midde left), Fitbit (right two)

Using the previous WiSAT prototype and other prior art as a basis, two alternate WiSAT prototypes (Figure 7) were created. These alternate designs featured differences in their Passive Notification screens (Appendix B). The other screens and design features were kept constant between the two alternates. Recent health and fitness apps were reviewed for further design inspiration. Special attention was paid to those apps with some basis in HBCT.

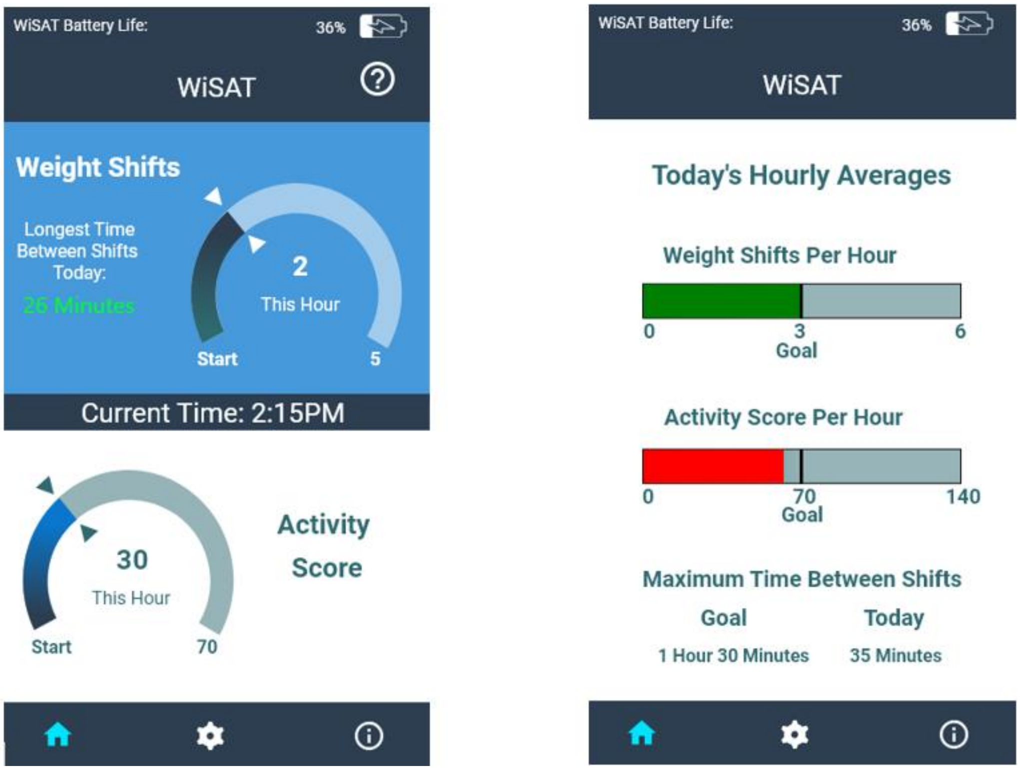
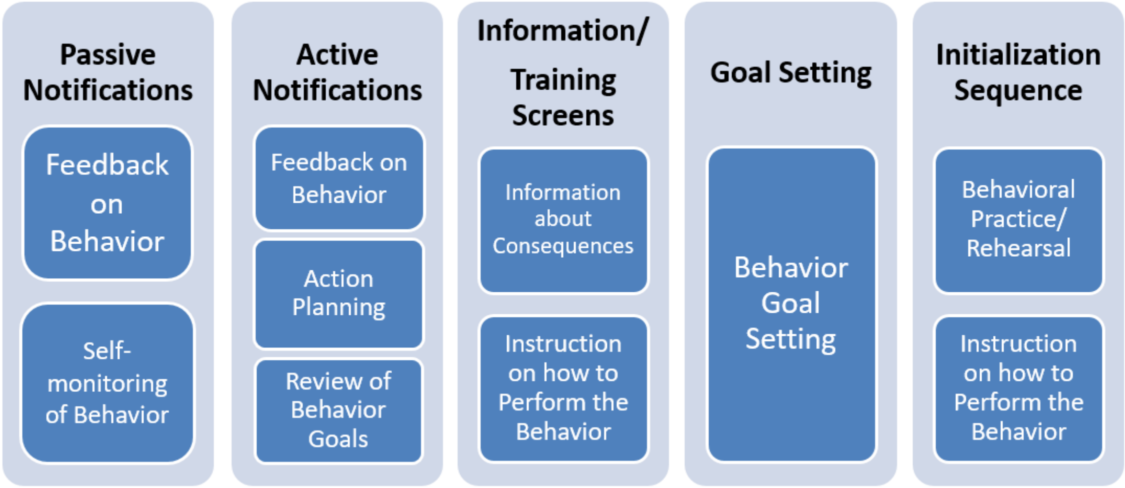


Figure 1: Home Screens of Design A (left) and Design B (right)

Of the original fourteen BCTs from Table 1, eight were incorporated into the final design of WiSAT, as outlined below:



Passive Notifications
- Feedback on Behavior
- Self-monitoring of Behavior

Active Notifications
- Feedback on Behavior
- Action Planning
- Review of Behavior Goals

Information/ Training Screens
- Information about Consequences
- Instruction on how to Perform the Behavior

Goal Setting
- Behavior Goal Setting

Initialization Sequence
- Behavioral Practice/ Rehearsal
- Instruction on how to Perform the Behavior

Figure 8: BCT Implementation in WiSAT

3.1.3 UI Evaluation

A UI evaluation was conducted to assess the app's design and overall usability. Members of the target audience were engaged with the goal of identifying design elements from both screen variants to be used in the final design. Ten participants were recruited, in total. Participants were full-time wheelchair users, aged 18 years or older who had some prior experience using mobile applications. This study excluded those who do not use a wheelchair as their primary means of mobility because they are not in the targeted user group of the WiSAT app. Additionally, this study excluded those under age 18 because the WiSAT app is targeted primarily at adults- the group most at risk of pressure ulcers. Last, subjects without any prior mobile application experience were excluded because smartphone ownership is a prerequisite of the WiSAT system.

Participants were asked to interact with the two prototype versions of the UI displayed on a smartphone through the Adobe XD app (Appendix B). They were asked to perform a series of tasks involving navigating through screens and retrieving information (Appendix C). During task execution, the researcher recorded the total time that the participant took to accomplish the task.

Data from the UI evaluation was used to inform the next iteration of the design of the WiSAT app. This evaluation was not structured to select a "winner" or "loser" between Design A and Design B. Rather, the evaluation results were used to select elements from both designs to include in the next design iteration.

Upon completion of the tasks, participants were shown the detail screens side-by-side and asked for their opinion on each. Participants were shown the home screen and day, week, and month screen from either the Weight Shift or In-Seat Movement score screens.

Additionally, participants were asked to interact with the prototype for a few minutes in order to provide open-ended feedback on the app, as a whole. The final six participants were asked for their opinion regarding active notifications and whether they would prefer to see notifications at a specified time of day or if they preferred to see them arrive on an “as-needed” basis.

Afterwards, participants were asked to complete a short survey to document their satisfaction with the interface. A modified version of the Software Usability Measurement Inventory (SUMI) was used as the survey instrument (Appendix D). The SUMI was developed in the early 1990s with psychometric methods as a software usability survey tool [62]. NIST lists the SUMI as one of several standardized survey tools suitable for measuring user satisfaction [50]. In additional to its use in industrial contexts [63], previous academic studies on mobile behavioral change apps have used the SUMI to measure user satisfaction [64, 65].

Results from navigational tasks were tabulated and compared across all ten participants. To measure effectiveness, the number of participants who failed to complete each task was noted. For efficiency, the time taken to complete each task was recorded and averaged for each task.

Responses to the side-by-side comparison questions were used to determine how many users preferred Design A or Design B’s interpretation of each passive notification screen. Further, open-ended feedback from this part of the study was used to help identify which features stood out to users as particularly desirable or undesirable.

Each user’s response to survey questions was coded as “satisfactory”, “unsatisfactory”, or “neutral”. Satisfactory responses were counted for each question to determine how many of the 10 users were satisfied with the specific element of the app which was addressed by that particular SUMI question.

Responses to the question about active notifications were counted as either favoring "as-needed" or "daily-reminder" active notifications. This information was used to tailor the design of WiSAT's active notification structure, including the selection of elements of the JITAI framework used in the deployment of WiSAT's active notifications.

3.2 Specific Aim 2: Coupling of the Hardware and Smartphone App Subsystems

Table 4: Scope of Specific Aim 2

Subsystem	**Prior State of Subsystem**	**Thesis Contribution**	**Outside Scope of Thesis (Third-Party or Future Work)**
Hardware	Version 1 Data Logger: • Pre-production Hardware and Firmware for a data logger designed to collect and store sensor data from a prototype sensor mat	• API to allow for communication between Data Logger and Smartphone App • Update to Data Storage and Transmission Format • Evaluation of Production Hardware and Firmware • Battery Evaluation	• Hardware and Firmware development performed by an external vendor • Sensor mat development
App	• Only UI designs. No prior implementation	• Integration plan for data communication between app and logger • Development of a standalone app to test API	• Implementation by external vendor

The prior iteration of the WiSAT data logger and sensor mat hardware, "Version 1" or "V1", was built primarily for research purposes. Originally, V1 provided researchers with a sensor-based solution to collect information pertaining to wheelchair user's in-seat activities. V1 was deployed primarily in a research setting and experienced very little user-interaction aside from its installation and removal by the research team. As such, usability was not a primary concern. Additionally, because V1 only featured a hardware subsystem, all data was stored locally on the logger, retrieved once at the end of each study via USB, and processed externally from the WiSAT system. While V1 was a valuable research tool, it was not viable as a consumer product.

Version 1 was converted into a consumer-product, “Version 2” or “V2”, during the duration of this thesis. While there were many components involved in the development of Version 2, Specific Aim 2 focused on the features of Version 2 that enabled communication between the app and the logger. The interface between the hardware and app subsystems was especially critical to WiSAT, as it involved transitioning a legacy hardware subsystem from a research tool with locally stored data to a consumer product capable of communication with the new smartphone app subsystem.

The original, Version 1, data logger was based upon the ADL16 logger by Gulf Coast Data Concepts (GCDC). The logger consisted of two modules, each with dimensions of 2.75” x 2.00” x 0.80”. No Bluetooth communication existed in the original logger. The sensor mat consisted of four Tekscan FlexiForce A502 sensors positioned on a flexible mat. In both versions of WiSAT, the sensor mat was placed underneath the wheelchair user’s cushion, and the data logger was placed in the side or rear of the cushion under the cushion cover. In both V1 and V2, raw, analog data is collected from the WiSAT sensor mat at a rate of 4 Hertz, converted to a digital signal, assigned a timestamp, and stored in an on-board SD card (Figure 9). Data stored on the SD card was formatted as comma-separated values with each sample separated by a new line.

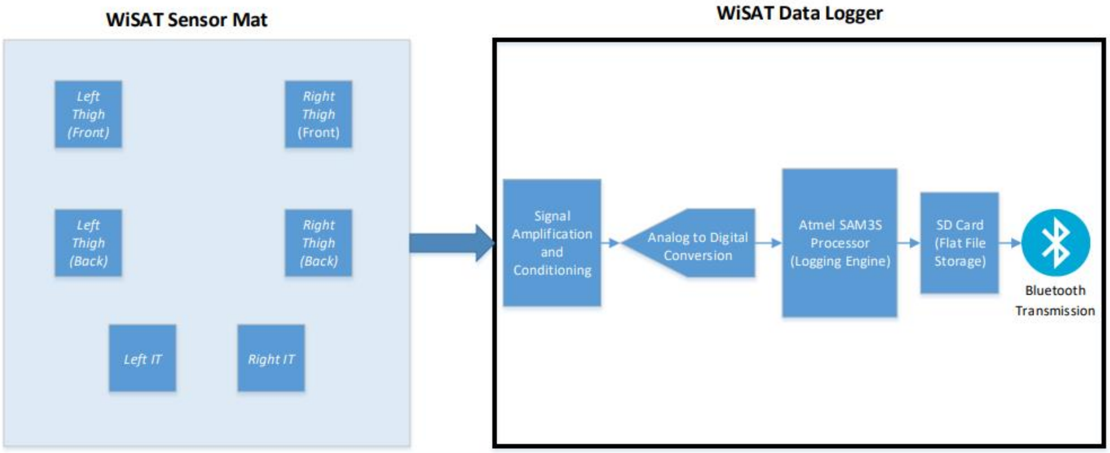


Figure 9: WiSAT Data Acquisition (Version 1 and 2)

As a consumer-product, the development of Version 2 emphasized an accessible, unobtrusive design, compatible with the other WiSAT subsystems. Through Specific Aim 2, the data storage scheme was redefined, data formatting was optimized, an API was specified that allowed for two modes of data transmission from the logger to app, and a battery solution was assessed. Quality assurance was conducted on all firmware and hardware changes. This included a capacity test to simulate the logger's behavior after several months of operation (see Appendix E for quality assurance and capacity test protocol).

Initially, plans were created to upgrade the logger's processor to allow for faster, more complex data processing. Additionally, data was to be stored in a SQLite database to utilize a trusted, third-party data storage solution. However, due to external complications encountered by the hardware vendor, this solution was abandoned in favor of one that continued to use the original Version 1 processor and flat-file data storage.

3.2.1 Data Storage and Batching

The logger's data storage scheme was redesigned in Specific Aim 2 to allow for reliable communication between the app and logger. As a research device, the Version 1 logger stored data with the assumption that it would be retrieved by researchers after data collection had completed. When connected to a computer via a USB port, the logger would behave as a portable flash drive and allow the researchers to retrieve data files directly from the logger's file system.

In Version 2, emphasis shifted to preparing data for transmission to the smartphone app. It was assumed that retrieval directly from the data logger was now only necessary for debugging purposes. Additionally, collected data no longer had clearly defined start and top times, as it had in research conducted with the Version 1 logger. Instead, data was assumed to be collected continuously.

For the work performed within the scope of this thesis, an assumption was made that the fully-integrated system should be able to function with minimal support for a period of at least six months. After this period, data could be manually cleared from the data logger and app to avoid malfunction or an excessive consumption of hard drive space. This assumption prepared the logger for long-term deployment in future research studies by preserving as much data as possible in the WiSAT subsystems, in case it was needed for debugging or analysis purposes.

3.2.2 Data Formatting

To help ensure consistent, reliable data transmission, Version 2 restructured the original data storage format of the logger from comma-separated values (CSV) to Javascript Object Notation (JSON). In Version 1, retrieving data through USB presented a much lower risk of lost or corrupted data than the wireless transmission of Version 2. This introduced an added challenge of accurately maintaining accuracy and consistency across subsystems, which the CSV format was not able to adequately address. Therefore, both the format and content of the individual batch files were assessed to minimize the risk of lost or corrupted data in the hand-off between the logger and app. As part of this effort, timestamp formatting was revisited, resulting in the decision to use UNIX timestamps throughout the WiSAT system.

3.2.3 API for Communication between Logger and App

WiSAT uses two modes of operation to manage communication between the logger and the app. The first mode is the default behavior of the WiSAT system in which data is transmitted regularly throughout the day. While both data integrity and transmission speed are important to WiSAT, it was decided that data reliability was to be favored over processing speed in this mode of operation. For the end-user, data is never displayed in true “real time”. Some of the app’s UI screens display weight shifts or in-seat movement score for a specific hour, but no smaller unit of time is visualized. Therefore, it was decided that the accuracy of the information reported was more important than “up to the second” reporting of in-seat activity. However, this did not negate the need for efficiency in WiSAT. It merely highlights that, in this mode of operation, accuracy will be chosen over speed when a decision must be made between the two.

The second mode addresses a specific need within the app’s “Setup and Initialization” procedure. Prior to their first use of WiSAT, the app prompts users to capture some baseline data for later use in the algorithm subsystem. Because the app prompts users to perform specific movements in real-time, it is crucial for the logger to transmit live data to the app as efficiently as possible. In this scenario, speed is favored over data integrity.

3.2.4 Battery Evaluation

As a side-effect of Bluetooth integration and longer usage times, the data logger system consumes more power than in Version 1. For Version 2, batteries of varying capacities and form factors were considered. Each battery was tested through installation in the data logger, charging it to a maximum capacity, and allowing it to discharge naturally while the logger was turned on and connected to a loaded sensor mat. Voltages were recorded every two minutes in the headers of the WiSAT batch files. These voltage readings were graphed, as shown in Figure 10.

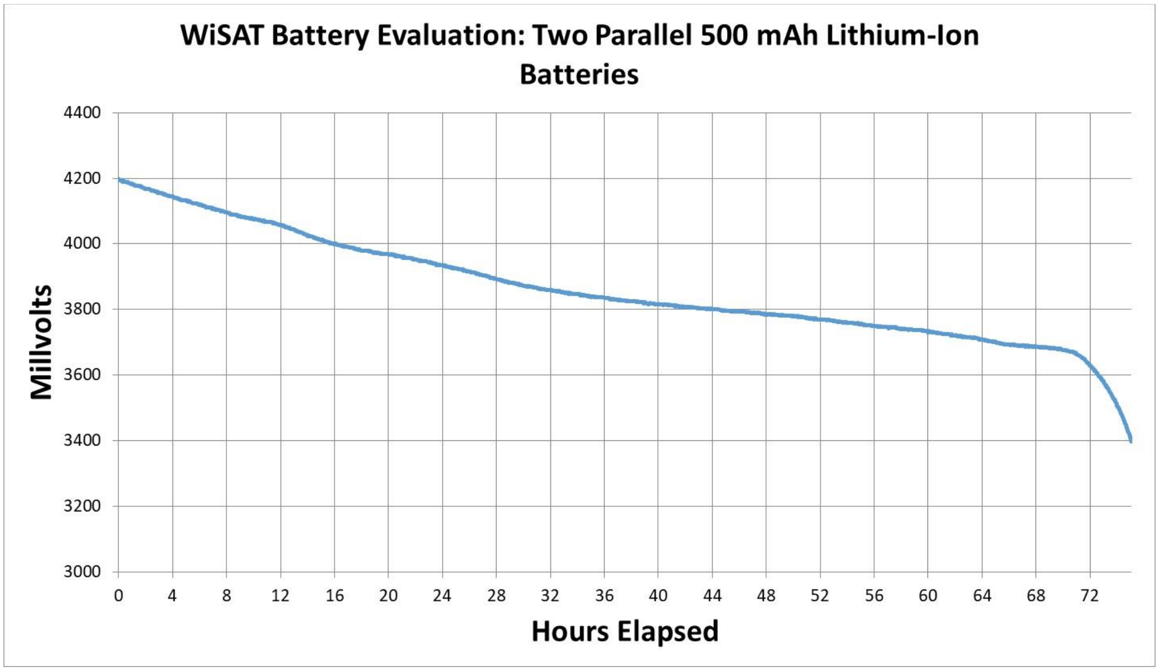


Figure 10: Example Battery Plot

3.2.5 Quality Assurance

Appendix E outlines the hardware quality assurance (QA) process that was developed during the course of this thesis. This process helped ensure that all incoming data loggers were functional and met specifications. Additionally, it facilitated the early identification of firmware bugs.

QA performed as part of Specific Aim 2 tested both isolated components, as well as integrated parts. In evaluating the functionality and resiliency of individual components, it was necessary to eliminate as many confounding variables as possible. For example, to evaluate the Bluetooth API, a standalone Android app was written to issue commands to the logger. This allowed the researcher to test the implementation of the API on the logger in isolation from the WiSAT app. When a communication issue was encountered in an integration test between the logger and the app, the standalone app helped narrow down the source of the issue by eliminating variables during debugging.

3.2.6 Work Outside Scope of Thesis

Outside the scope of this thesis, a form factor change by the vendor allowed the entire Version 2 logger to fit inside a single 2.75” x 2.00” x 0.80” module. Additionally, a Version 2 sensor mat was developed alongside the Version 2 data logger. The Version 2 sensor mat was professionally manufactured and included an updated connector and six force sensors.

Figure 11: Version 2 WiSAT Sensor Mat

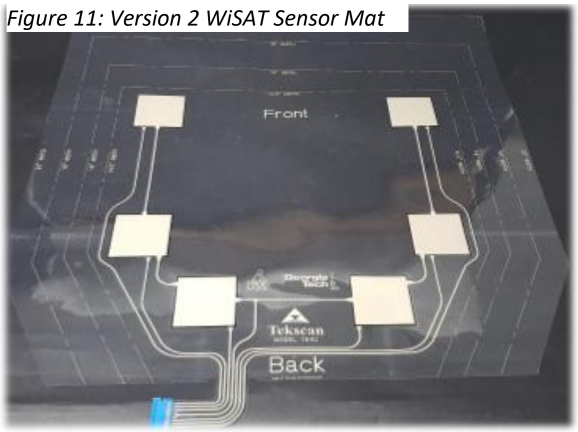


Figure 12: Version 2 Data Logger

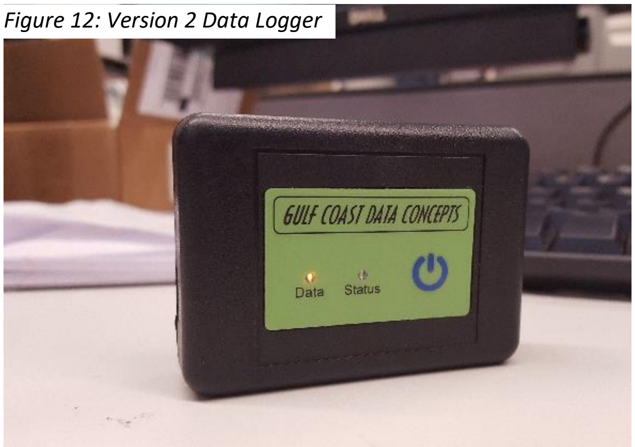
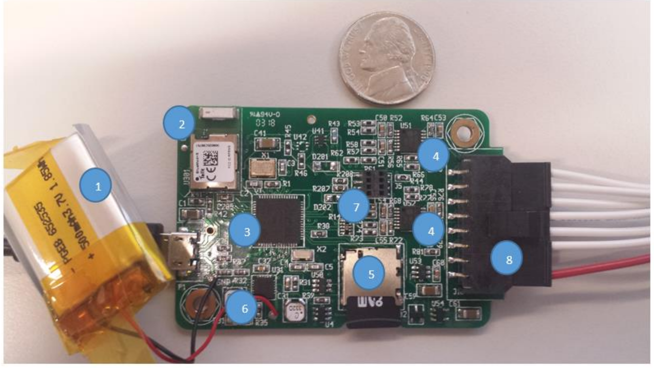
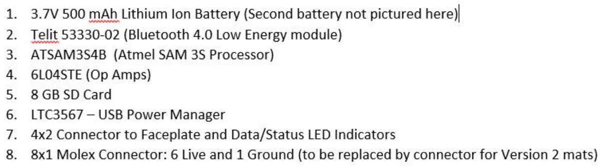


Figure 13: Version 2 Data Logger with Case Removed





1. 3.7V 500 mAh Lithium Ion Battery (Second battery not pictured here)
2. Telit 53330-02 (Bluetooth 4.0 Low Energy module)
3. ATSAM3S4B (Atmel SAM 3S Processor)
4. 6L04STE (Op Amps)
5. 8 GB SD Card
6. LTC3567 – USB Power Manager
7. 4x2 Connector to Faceplate and Data/Status LED Indicators
8. 8x1 Molex Connector: 6 Live and 1 Ground (to be replaced by connector for Version 2 mats)

3.3 Specific Aim 3: Preparation of Algorithm Subsystem

Table 5: Scope of Specific Aim 3

Subsystem	**Prior State of Subsystem**	**Thesis Contribution**	**Outside Scope of Thesis (Third-Party or Future Work)**
Algorithm Library	• Classification algorithms for Occupancy, Weight Shift, and In-Seat Movement Score detection written in Matlab • Calibration function written in Matlab • Initialization function to collect baseline information written in Matlab	• Translation of algorithms from Matlab to libraries compatible with the WiSAT app • Added features to algorithms to allow for continuous data processing and resiliency to smartphone power cycles.	• Development and testing of the algorithms in Matlab • Implementation of algorithm coding in Java and Swift was conducted with the help of the app vendor and two undergraduate developers

The WiSAT classification algorithms were originally developed to work on discrete data sets within a pre-configured Matlab development environment. The smartphone adaptation was conducted in two parts. First, the algorithms were translated from a procedural coding style to object-oriented programming languages. Since WiSAT supports both Android and iOS operating systems, this meant translating the algorithms into Java and Swift, respectively. This translation effort emphasized “loose-coupling” between the algorithms and the rest of the app. This meant simplifying and standardizing the algorithms entry points and outputs across revisions of the algorithm code. Keeping the algorithms modular allowed the app developer to seamlessly exchange older versions of the algorithms with updated ones without having to rework any of the surrounding application code. Conversely, changes in the mobile app did not affect the internal workings of the algorithm code.

Second, the algorithms were modified to function continuously instead of only on discrete data sets. While in the laboratory environment, the algorithms processed data sets in discrete batches that were produced in controlled environments. However, in the smartphone app, data was received continuously. Thus, by definition, there was no predetermined end to the incoming data. Additionally, in a smartphone app, it is impossible to predict interruptions to the incoming data. For example, when one hour of sensor mat data is processed by the Matlab algorithms, the entire hour’s data is passed into the algorithms prior to their execution, and it is known beforehand that the start and end time of any occupancy events, weight shifts, and activity occur within the duration of this hour.

The smartphone algorithms must process data one datapoint at a time and must account for external events that may lead to corrupt or incomplete data. For example, power failure on the data logger could lead to gaps in the incoming data, and Bluetooth issues could lead to delays in data being received by the app. In Specific Aim 3, features were added to the algorithm libraries to build resiliency against these external events without sacrificing any of the functionality of the original algorithms. Specifically, a strategy was introduced which provided the app with context regarding occupancy and weight shift events that occurred prior to the app restarting. Further, this strategy managed data interruptions by basing all algorithm calculations on the timestamps of the datapoints and ignoring the specific times in which data is received.

A standalone testing tool was developed to test the algorithms in isolation from the app subsystem. This allowed for rapid testing without the additional setup time involved with deploying the algorithms in a mobile app. This also aided in pinpointing the source of bugs as they arose. Algorithms were evaluated by comparing their performance against that of the original Matlab algorithms, as well as against data collected in the field. Testing was also conducted through the app after the algorithms subsystem had been implemented by the app developer.

3.4 Specific Aim 4: Integration of the WiSAT Subsystems through Multi-layered Architecture for the WiSAT Smartphone App

Table 6: Scope of Specific Aim 4

Subsystem	**Prior State of Subsystem**	**Thesis Contribution**	**Outside Scope of Thesis (Third-Party or Future Work)**
App	• Early iteration of UI design	• Database Schema • Integration plan for interfaces between data logger and app, app to remote API, and algorithm library to app. • Selection and management of App development contractor	• Implementation of App (development and coding)

Remote API	*No prior version*	• API Specification	• Implementation of the API on a third-party server • Hosting of the API Endpoints

Due to its centrality to the system, the smartphone app is the only subsystem to directly integrate with all three additional subsystems. Additionally, the app experiences the vast majority of interaction with the end-user. Specific Aim 4 was responsible for the supporting framework that enabled the app to fulfill these requirements, and therefore, tied the individual components of WiSAT into one fully-integrated system.

Prior to the onset of this thesis, it was decided that the system would target the Android and iPhone platforms and display information pertaining to WiSAT's three main metrics: number of weight shifts, time between weight shifts, and in-seat movement score points. System requirements related to data collection and flow is consolidated in the graphic displayed in Figure 14. Additional system requirements were related to the user-interface design discussed in Specific Aim 1. Some additional technical requirements were also defined, such as the use of a Realm database in the mobile app. This conceptualization of the WiSAT data flow and system requirements informed the development of the three layers of the WiSAT app: UI layer, application layer, and database layer. A full list of system requirements are outlined in Appendix F.

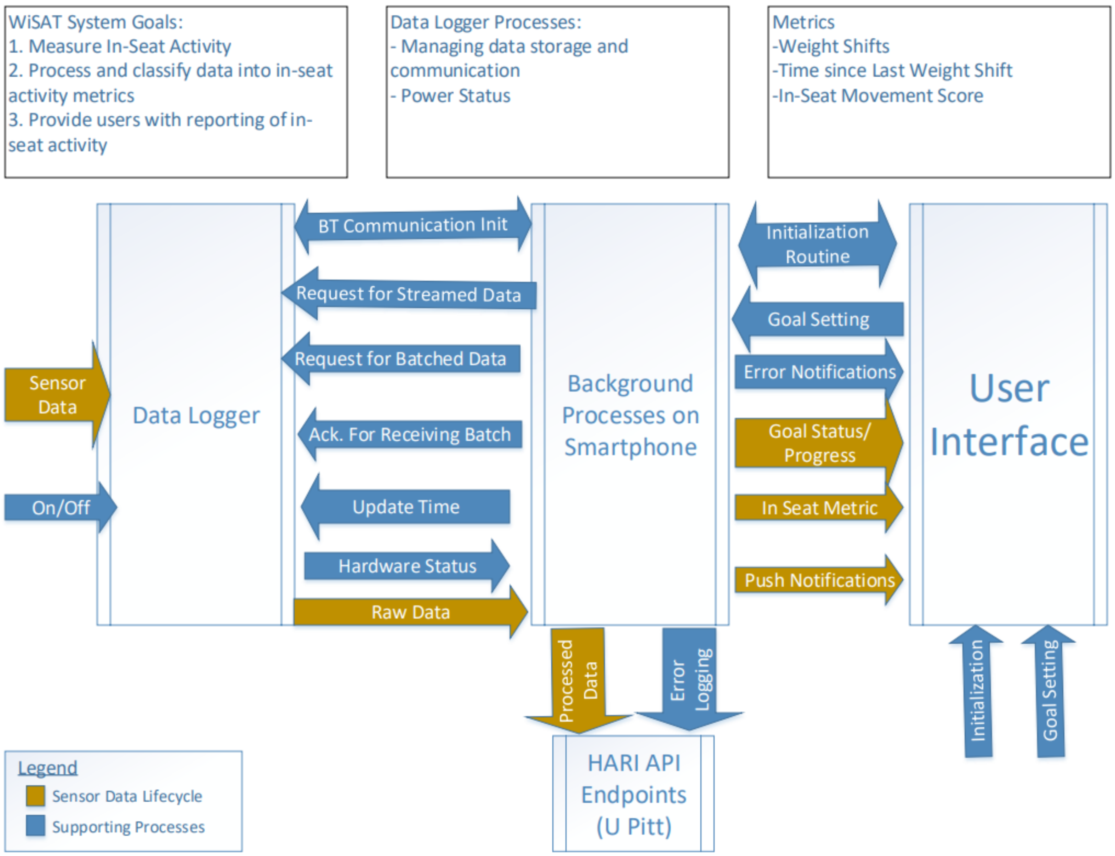


Figure 14: WiSAT System Requirements and Data Flow

An experienced software development firm, Apeiro Technologies, was contracted to write the smartphone application code according to the research team's specifications. Principles of the Agile Methodology, as well as the Iterative and Incremental Development model were employed during the development lifecycle of the app. The system was developed iteratively to identify potential flaws early in the development lifecycle. This enabled the team to spot technical and process-related flaws early, improve overall product quality, work on multiple systems in parallel, and accommodate changes as they arose [66-68]. This methodology was especially relevant to the goals of Specific Aim 4, as it helped ensure that a change in one subsystem could be quickly accounted for in the other subsystems.

Specific Aim 4 also included a specification of the API used by the app to communicate with the remote server subsystem. The app's database schema was used to construct this specification, and individual endpoints mirror their counterpart tables in the schema. Early in the development process, it was decided that WiSAT would be a primarily offline system. Ultimately, the requirement to upload information to a remote server was done for research purposes and is the artifact of WiSAT's research grant. WiSAT was designed so that the remote server subsystem could be removed, if desired, once WiSAT is deployed as a consumer product.

Since it is never assumed that the user has a guaranteed internet connection, all data processing is done in the app subsystem, and the remote server functions merely as a place to collect and aggregate user data for future research and analysis purposes. No data is received by the app from the server, and no web portal or data analysis tool is currently planned to interface with this remote server. Research partners at the University of Pittsburgh's Health and Rehabilitation Informatics lab were engaged to implement and host this API.

Quality Assurance of WiSAT was conducted at the task and system level. Functional test cases were written around individual tasks, where appropriate, to ensure that they were bug-free and working as expected (Appendix G). Smoke testing was conducted according to expected use-cases, and exploratory testing was conducted to identify potential edge cases.

CHAPTER 4. IMPLEMENTATION, DELIVERABLES, AND RESULTS

4.1 Specific Aim 1: Design and evaluation of a user-interface

4.1.1 Passive Notifications

Several key pieces of information were obtained from the UI Evaluation that contributed towards the final design of WiSAT. Efficiency and Effectiveness results are listed in Table 7 and Table 8. None of the average times to complete specific tasks exceeded 30 seconds. Considering this was the first time any of the participants had interacted with the app, none of these task completion times were especially high and did not lead to any changes in the final app design.

Table 7: Efficiency and Effectiveness Results of UI Evaluation – Alternate Designs

	2.1 Go to the screen for weight shifts. In which hours did the user meet the weight shift goal?	2.2 Navigate to a screen that shows weight shift statistics for the month. How many times did this user miss their goals in the past month?	3.1 Go to the In-Seat Movement (ISM) Score screen. Check the ISM Score status for the day. Comment on the ISM Score with respect to the goal	3.2 2. Navigate to the Weekly information. In how many days was the ISM goal met for the week?
Design A				
Average	21	7	7	4
Highest	58	15	16	9
Lowest	9	4	4	1
Standard Dev	17	4	4	3
Misses	2	1	3	1
Design B				
Average	19	10	9	5
Highest	52	34	19	8
Lowest	7	4	5	1
Standard Dev	19	10	5	2
Misses	5	1	2	2

Table 8: Efficiency and Effectiveness of UI Evaluation - Information and Settings Screens

	Sequence 1: 1.1 Go to the Information page. Go to the information about battery charging	Sequence 1: 1.2 Read information and comment: was it clear? Do you understand how you would charge the battery?	Sequence 2: 1.1 B Go to the Information page. Go to the information about Pressure Ulcers.	Sequence 1: 4.1 Go to settings page. Go to the setting a goal page	Sequence 1: 4.2 Set an ISM Score goal of 80 per hour
Average	12	21	5	4	10
Highest	36	40	12	10	25
Lowest	3	2	2	1	2
Standard Dev	11	13	3	3	9
Misses	2	0	1	0	1

In terms of effectiveness, five participants had trouble with task 2.1 in Design B: “Go to the screen for Weight Shifts. In which hours did the user meet the weight shift goal?”. Three of these five failures were due to difficulties interpreting the graph. The other two were navigation issues. Based on this, the “Day” and “Week” screens were revised for the final design. Feedback received in the side-by-side comparison (Table 9) and open-ended feedback (Table 10) revealed a compromise between the two designs. Bar Charts from Design A replaced the Line Graph of Design B, but colors and numbers from Design B were utilized in the final design (Figure 15).

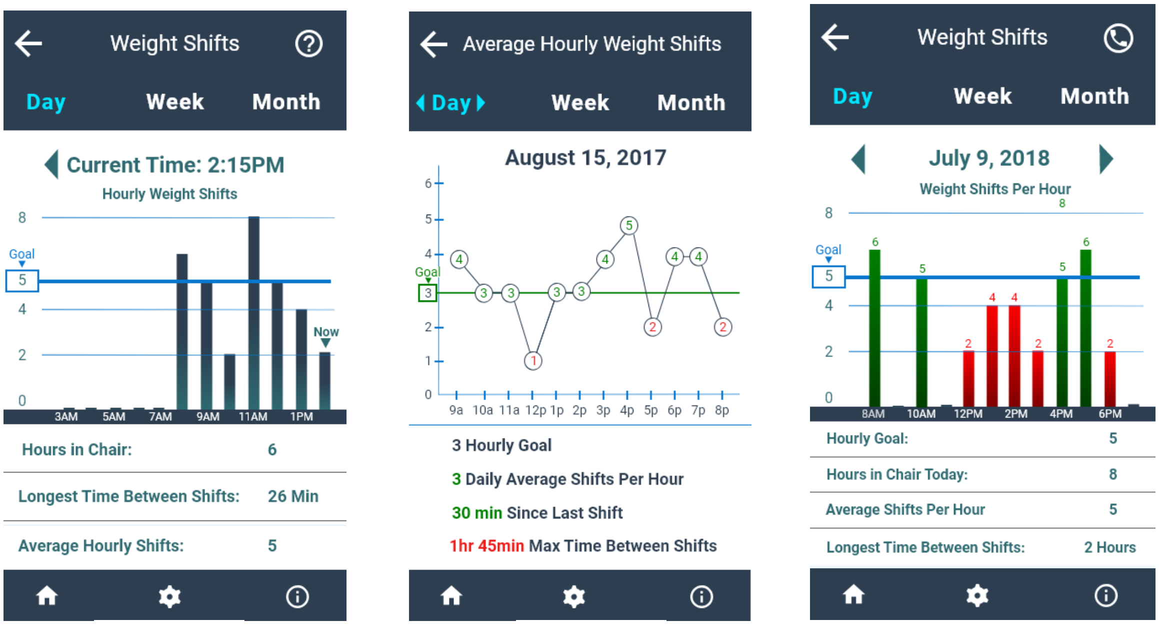
Table 9: Side-By-Side Evaluation of Detail Screens

	Home	**Day**	**Week**	**Month**
Prefers Design A	8	5	4	5
Prefers Design B	1	4	5	3

Table 10: Open-Ended Responses to Day and Week Side-By-Side Comparisons

#	**Day**	**Week**
1	B: Likes to be able to read off data points. Likes that there are numbers in the circles in the graphs B: Likes the color-coded data points	B: Likes having more data
2	B: “really easy” – likes numbers. He would rather not look on X and Y axis to figure out the time and score “Bar graphs are easier to read” “One is easier to read. One is more convenient”	B: Likes “days of the week at bottom and times and things” B: seems like more information
3	Likes bar graph style of A but likes numbers in B Wants a combination where we have a bar graph with numbers Likes the way that screen A displays information (“title”: “number”) Likes how B also includes the Goal on screen Likes the wording on screen A	B: Likes bar graph with numbers Wants to have more summarized information (as in A) by default, but wants to see more information by clicking on a day Likes screen variants about the same Likes bar graph
4	A: Prefers bar graphs Line straight across lets me pick out when goals are met Likes having the current time displayed B: Looks like an Excel graph He said it was a close decision between the two	A: more info on bottom
5	Not sure which he liked better, at first A: Aesthetics better; function the same B: likes numbers	B: likes having more information (especially with regards to max time) “Definitely B”
6	A: Clearer; Can read chart right away	A: Seems more clear

	B: Has to go from left to right and count with his finger	
7	Doesn’t like PM/AM B: Likes that the circle contains numbers; More information at a glance “Brain resets” when having to shift focus: He compared this to how he keeps his file folders all in one line	B: Crisper, more direct Harder for him to process more information Would like B because of the breakdown of max weight shift per day: doesn’t have to think about it as much
8	“I prefer bar charts” “To me it’s easier to read bar charts”	“Better organized” “More detailed”
9	Prefers line graphs versus bar graphs “Information is the same” Likes the actual number being displayed in the line graph	A: “I like this one a little bit better” A: “it gives you a little bit more information”. B: “Some of it more relevant than this one”
10	A: easier to read “Less things going places” - referring to line graph “B confuses me for a minute” “Times in A match up easier for me” – referring to matching X axis to the goals	“About the same to me” B: “I may like this one a little more” – referring to the max time between shifts at the bottom of the screen Liked max time in bottom of screen “A doesn’t show goal”



Hours in Chair: 6

Longest Time Between Shifts: 26 Min

Average Hourly Shifts: 5

Design A

3 Hourly Goal

3 Daily Average Shifts Per Hour

30 min Since Last Shift

1hr 45min Max Time Between Shifts

Design B

Hourly Goal: 5

Hours in Chair Today: 8

Average Shifts Per Hour 5

Longest Time Between Shifts: 2 Hours

Final Design

Figure 15: WiSAT Design Alternatives

A similar process was used to implement other changes in the final design. For example, the final home screen used green and red to differentiate between passing and failing, as in Design B, but it inherited its odometer style graphs from Design A. Likewise, the final month screen design used Design B's style of calendar, but "Daily Average Hours in Chair" and "Average Hourly Shifts" below the calendar as in Design A.

As seen in Table 11, 26 of 29 SUMI questions received satisfied responses from at least 5 (50%) of users. Because this indicated that most participants appeared satisfied with the WiSAT prototype, the results of the SUMI survey did not directly lead to any changes in the final design. However, these results may be a useful point of comparison if the SUMI is deployed in future usability studies of WiSAT.

Table 11: Summary of SUMI Results

#	Question	Satisfied Respondents	Unsatisfied Respondents	Neutral Respondents
Subscale A: Affect (i.e. likeability)				
1	Working with this software is satisfying.	9	0	1
2	The way that system information is presented is clear and understandable	9	0	1
3	Working with this software is mentally stimulating	7	1	2
4	There is never enough information on the screen when it's needed	7	3	0
5	I feel in command of this software when I am using it.	9	0	1
6	I prefer to stick to the facilities that I know best.	4	3	3
Subscale B: Helpfulness				
1	I think this software is inconsistent (This question accidentally left out of survey for Participant 1)	8	0	1
2	I would not like to use this software every day	5	2	3
3	I can understand and act on the information provided by this software.	10	0	0
4	This software is awkward when I want to do something which is not standard.	5	1	4
5	There is too much to read before you can use the software.	7	1	2
6	Tasks can be performed in a straightforward manner using this software	10	0	0

7	Using this software is frustrating	10	0	0
8	The software has helped me overcome any problems I have had in using it.	4	0	6
9	I keep having to go back to look at the guides.	7	1	2
Subscale C: Control				
1	It is obvious that user needs have been fully taken into consideration.	9	0	1
2	There have been times in using this software when I have felt quite tense.	8	0	2
3	The organization of the menus or information lists seems quite logical	10	0	0
4	The software allows the user to be economic of keystrokes.	8	1	1
5	There are too many steps required to get something to work	10	0	0
6	It is easy to make the software do exactly what you want	9	0	0
7	I will never learn to use all that is offered in this software.	8	0	2
Subscale D: Learnability				
1	The software has a very attractive presentation.	8	0	2
2	Either the amount or quality of the help information varies across the system.	1	5	4
3	It is relatively easy to move from one part of a task to another.	10	0	0
4	It is easy to forget how to do things with this software	9	0	1
5	This software is really very awkward.	10	0	0

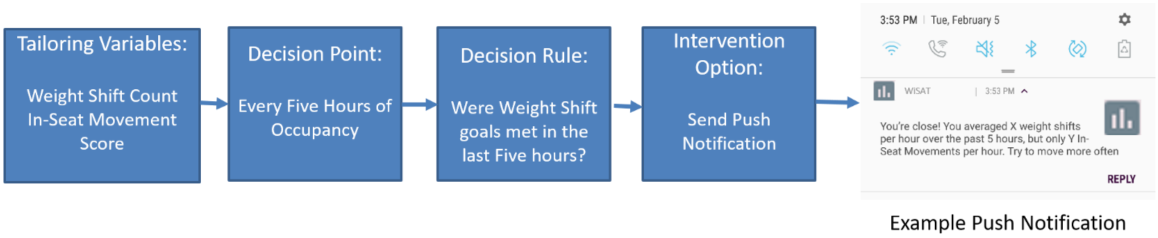
6	It is easy to see at a glance what the options are at each stage	9	0	1
7	I have to look for assistance most times when I use this software	9	0	1

Elsewhere in the app, user feedback informed a few additional changes. For example, the pressure ulcer information found in the Information and Training screen was updated to include a few additional stages of pressure ulcers that were not in the original screen. Additionally, feedback revealed that the logger's battery life indicator on the home screen was easily confused with the battery life indicator of the smartphone itself. This was remedied by moving the icon lower in the home screen.

4.1.2 Active Notifications

Originally, two decision point options were considered. The first option would send push notifications on a predetermined schedule regardless of external events. The second option allowed for push notifications to arrive based on some action on the part of the user. Six participants from the UI Evaluation were asked about their preferences between the two options. Four of the six preferred push notifications arriving based on some user action (with one participant undecided). From this, it was decided that the decision point should be reached once every time the user reached a certain amount of hours in their chair (as opposed to a certain time of day). A five hour occupancy time was chosen following a review of push notifications in other, similar apps. One of the key lessons learned from other apps was that it was crucial to strike a balance between annoying the user with too many notifications and not reminding them enough. Additionally, using a five hour window minimized the effect that any delayed data processing might have upon shorter time periods. For example, a 30 minute delay is 25% of a 2 hour window, but only 10% of a five hour window.

WiSAT’s Decision Rule simply asks whether or not the Weight Shift or In-Seat Movement score goals were met, on average, for the past five hours of occupancy. The user may see one of four Intervention options based on whether one, both, or neither of these goals were met. All four intervention options involve sending a push notification to the user to remind them of their performance. Users are congratulated if their goal(s) were met and encouraged to meet the goal(s) that they didn’t meet. Messages were kept to under 150 characters to ensure that they fit on all targeted smartphone devices.



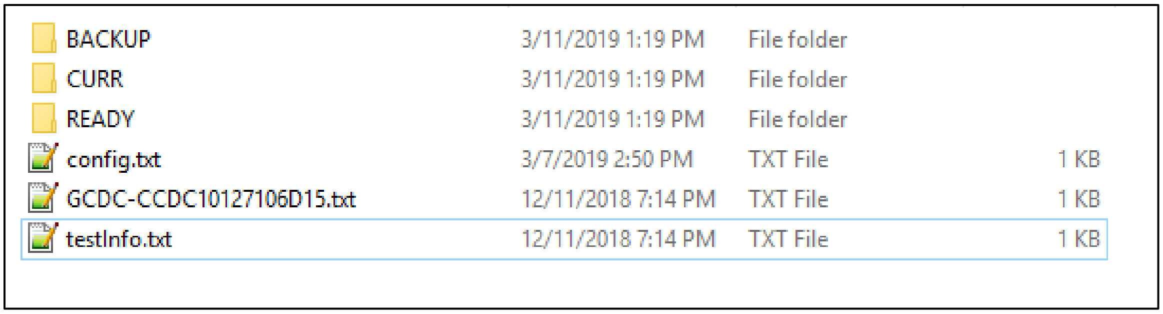
You’re close! You averaged X weight shifts per hour over the past 5 hours, but only Y In-Seat Movements per hour. Try to move more often

Figure 16: Example of JITAI in WiSAT

4.2 Specific Aim 2: Coupling of the Hardware and Smartphone App Subsystems

4.2.1 Data Storage and Batching

Version 2 uses flat-file data storage, but files are divided into two-minute batches. The logger’s file system is divided into three sub-directories: CURR, BACKUP, and READY (Figure 17). As data is collected by the logger, it is temporarily stored in a file in the CURR directory. Following the completion of the batch, the file is moved to READY, where it remains until requested by the smartphone app. Once transfer to the app is completed, the batch is stored in BACKUP until manually removed from the logger for debugging or research purposes.



```
BACKUP                      3/11/2019 1:19 PM    File folder
CURR                        3/11/2019 1:19 PM    File folder
READY                       3/11/2019 1:19 PM    File folder
config.txt                  3/7/2019 2:50 PM     TXT File       1 KB
GCDC-CCDC10127106D15.txt    12/11/2018 7:14 PM   TXT File       1 KB
testInfo.txt                12/11/2018 7:14 PM   TXT File       1 KB
```

Figure 17: Data Logger's Root Directory as Viewed by a Windows PC

Two-minute batches were chosen following capacity testing, as they balance the concerns of small and large batch sizes. Smaller batch sizes are preferable for two reasons. First, batch files are not sent to the app until they were completed. Therefore, there is an inherent time delay between the app and the logger. This delay increases as the batch size increases. As an extreme example, a very large batch size, such as a one-hour batch, could potentially affect user-experience. Any weight shifts or in-seat movements that occur during that batch would not be received by the app for an hour or more. Second, smaller batch sizes minimize the total data lost in corrupt batches. As with any such system, an occasional lost or corrupted file may occur over the lifetime of the system. Minimizing the amount of data contained in each batch also minimizes the amount of data that is lost when this occurs.

On the other extreme, reducing the batch file to an extremely small size bring its own risks. Due to the presence of a header in each batch file, there is additional overhead associated with each batch file transfer. If the batches were reduced to an extremely small size, for example, one sample per file, the logger would effectively be streaming data with the added overhead of header information. Further, as the size of each batch decreases, the number of batches required to capture the same amount of data increases.

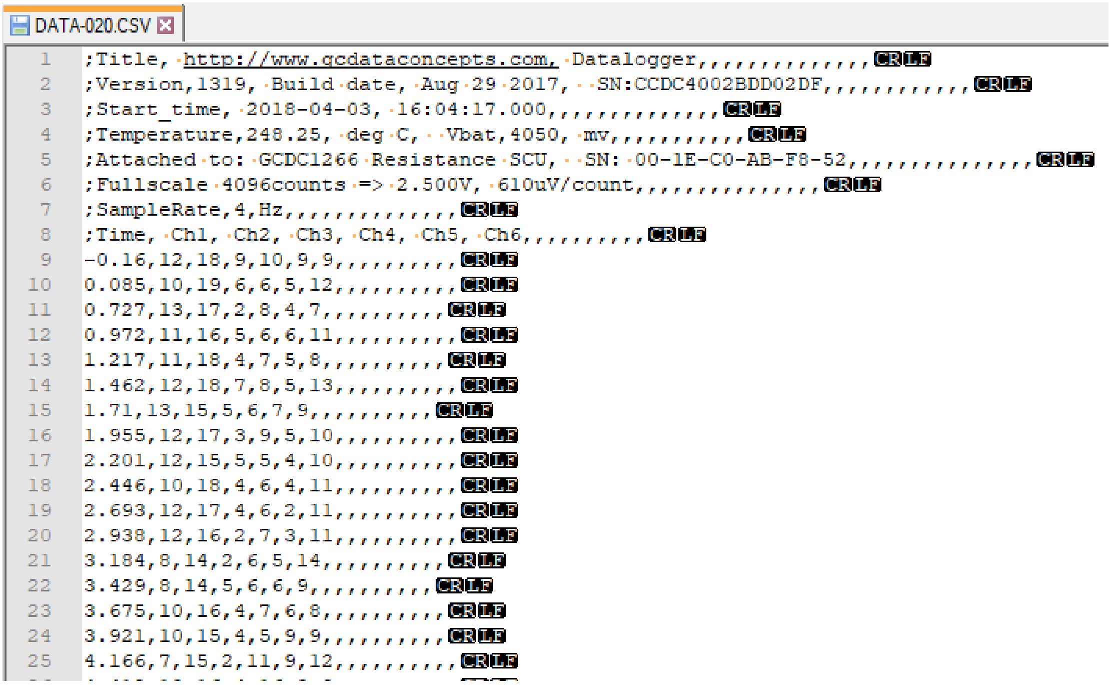
Capacity tests revealed that the logger's file system experienced two problems as the number of files increased to the tens of thousands. First, each new batch file would take an increasingly longer time to save to the file system as the total number of files in the logger's hardware increased. During the time that the file was being saved to the logger's file system, data collection and Bluetooth communication would suspend. The suspension of data collection was corrected through a firmware update. The Bluetooth communication issue was not as easily corrected, but since it merely delays data transfers and does not result in data loss, this did not present an extreme risk to the system.

Second, testing revealed that each directory in the logger was limited to a number of files less than the capacity of the 16-bit integer which was used for file indexing. In other words, no directory could exceed 65,535 files. Therefore, if an extremely small batch size was chosen (~1 second), this limit could be exceeded in the READY or BACKUP directory with a single day's worth of data. However, with a two-minute batch, over 90 days of data could be stored in a single directory before this limit would be exceeded. To further protect against this issue, the BACKUP directory was divided into sub-directories. Each sub-directory was designed to hold batch files for a specific day, which greatly diminished the chances of an individual folder reaching the 65,535 file capacity limit, even over the course of many months.

4.2.2 Data Formatting

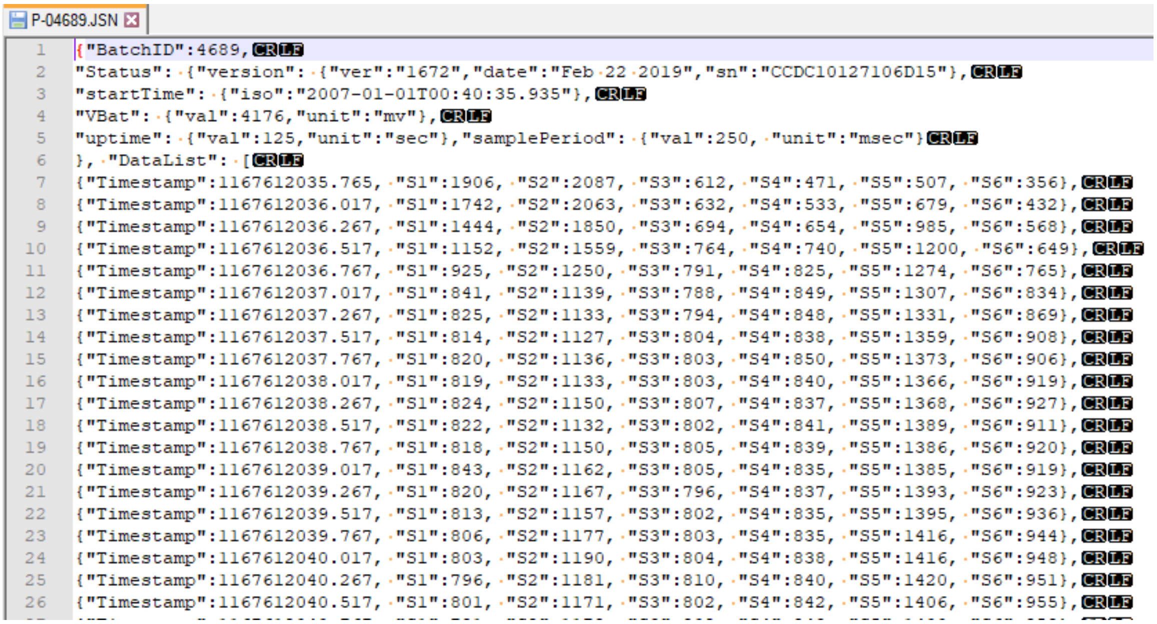
As seen in Figure 19, the Version 2 JSON structure conveys information via nested objects consisting of key-value pairs. For example, the key, "Status", contains a nested object as its value. Within this object, three key-value pairs provide diagnostic information about the current state of the logger. The "DataList" key contains an array which stores the data samples from the logger. Each object within array represents a single sample and consists of a Timestamp, Sensor1, Sensor2, Sensor3, Sensor4, Sensor5, and Sensor6.

Three benefits were realized through this format. First, Version 1's CSV format relied on the assumption that the six sensors were arranged in the correct order for every sample. Therefore, a missing comma, data value, or line return (as may occur during Bluetooth transmission) could potentially obfuscate the data. Since each piece of data was enveloped in a key-value pair in Version 2, it became more explicit which sensors belonged to which values. Second, because of the prevalence of JSON among smartphone apps, the format kept deserialization on the smartphone app relatively simple. Third-party software libraries are readily available for both Java and Swift that aid in JSON deserialization. Further, because JSON is object-based, it provided the basis for which to form the basic WiSAT data objects in the smartphone. Last, JSON provides a more human readable format than other solutions, as it does not require any special software or tools to read and interpret. This was important during development and testing of all subsystems.



```
;Title, http://www.gcdataconcepts.com, Datalogger,,,,,,,,,,,,,
;Version,1319, Build date, Aug 29 2017,  SN:CCDC4002BDD02DF,,,,,,,,,,,
;Start_time, 2018-04-03, 16:04:17.000,,,,,,,,,,,,,
;Temperature,248.25, deg C,  Vbat,4050, mv,,,,,,,,,,
;Attached to: GCDC1266 Resistance SCU,  SN: 00-1E-C0-AB-F8-52,,,,,,,,,,,,,,
;Fullscale 4096counts => 2.500V, 610uV/count,,,,,,,,,,,,,,
;SampleRate,4,Hz,,,,,,,,,,,,,
;Time, Ch1, Ch2, Ch3, Ch4, Ch5, Ch6,,,,,,,,,
-0.16,12,18,9,10,9,9,,,,,,,,,
0.085,10,19,6,6,5,12,,,,,,,,,
0.727,13,17,2,8,4,7,,,,,,,,,
0.972,11,16,5,6,6,11,,,,,,,,,
1.217,11,18,4,7,5,8,,,,,,,,,
1.462,12,18,7,8,5,13,,,,,,,,,
1.71,13,15,5,6,7,9,,,,,,,,,
1.955,12,17,3,9,5,10,,,,,,,,,
2.201,12,15,5,5,4,10,,,,,,,,,
2.446,10,18,4,6,4,11,,,,,,,,,
2.693,12,17,4,6,2,11,,,,,,,,,
2.938,12,16,2,7,3,11,,,,,,,,,
3.184,8,14,2,6,5,14,,,,,,,,,
3.429,8,14,5,6,6,9,,,,,,,,,
3.675,10,16,4,7,6,8,,,,,,,,,
3.921,10,15,4,5,9,9,,,,,,,,,
4.166,7,15,2,11,9,12,,,,,,,,,
```

Figure 18: Version 1 CSV Data (As Viewed in Notepad++)



```
{"BatchID":4689,
"Status": {"version": {"ver":"1672","date":"Feb 22 2019","sn":"CCDC10127106D15"},
"startTime": {"iso":"2007-01-01T00:40:35.935"},
"VBat": {"val":4176,"unit":"mv"},
"uptime": {"val":125,"unit":"sec"},"samplePeriod": {"val":250, "unit":"msec"}
}, "DataList": [
{"Timestamp":1167612035.765, "S1":1906, "S2":2087, "S3":612, "S4":471, "S5":507, "S6":356},
{"Timestamp":1167612036.017, "S1":1742, "S2":2063, "S3":632, "S4":533, "S5":679, "S6":432},
{"Timestamp":1167612036.267, "S1":1444, "S2":1850, "S3":694, "S4":654, "S5":985, "S6":568},
{"Timestamp":1167612036.517, "S1":1152, "S2":1559, "S3":764, "S4":740, "S5":1200, "S6":649},
{"Timestamp":1167612036.767, "S1":925, "S2":1250, "S3":791, "S4":825, "S5":1274, "S6":765},
{"Timestamp":1167612037.017, "S1":841, "S2":1139, "S3":788, "S4":849, "S5":1307, "S6":834},
{"Timestamp":1167612037.267, "S1":825, "S2":1133, "S3":794, "S4":848, "S5":1331, "S6":869},
{"Timestamp":1167612037.517, "S1":814, "S2":1127, "S3":804, "S4":838, "S5":1359, "S6":908},
{"Timestamp":1167612037.767, "S1":820, "S2":1136, "S3":803, "S4":850, "S5":1373, "S6":906},
{"Timestamp":1167612038.017, "S1":819, "S2":1133, "S3":803, "S4":840, "S5":1366, "S6":919},
{"Timestamp":1167612038.267, "S1":824, "S2":1150, "S3":807, "S4":837, "S5":1368, "S6":927},
{"Timestamp":1167612038.517, "S1":822, "S2":1132, "S3":802, "S4":841, "S5":1389, "S6":911},
{"Timestamp":1167612038.767, "S1":818, "S2":1150, "S3":805, "S4":839, "S5":1386, "S6":920},
{"Timestamp":1167612039.017, "S1":843, "S2":1162, "S3":805, "S4":835, "S5":1385, "S6":919},
{"Timestamp":1167612039.267, "S1":820, "S2":1167, "S3":796, "S4":837, "S5":1393, "S6":923},
{"Timestamp":1167612039.517, "S1":813, "S2":1157, "S3":802, "S4":835, "S5":1395, "S6":936},
{"Timestamp":1167612039.767, "S1":806, "S2":1177, "S3":803, "S4":835, "S5":1416, "S6":944},
{"Timestamp":1167612040.017, "S1":803, "S2":1190, "S3":804, "S4":838, "S5":1416, "S6":948},
{"Timestamp":1167612040.267, "S1":796, "S2":1181, "S3":810, "S4":840, "S5":1420, "S6":951},
{"Timestamp":1167612040.517, "S1":801, "S2":1171, "S3":802, "S4":842, "S5":1406, "S6":955},
```

Figure 19: Version 2 JSON Data (Viewed in Notepad++)

Version 2 also faced time reporting constraints that were not present in the research environment. Version 1 reported its timestamps as values relative to the time that the logger began recording data. As seen in Figure 18, a single, human-readable timestamp was recorded at the beginning of the CSV file, and each subsequent data sample included a “Time” value that counted up from the start of the file. However, with Version 2, consistent timestamps needed to be maintained across the four subsystems. Additionally, as a consumer product, it was necessary to maintain accurate time across timezones, daylight-savings-time, and other nuances of time reporting.

Unix seconds were chosen as the standard by which to measure time across the entirety of the WiSAT system, including the data logger. Unix seconds allow WiSAT to express time as a single floating-point number while avoiding the complexities of parsing human-readable dates. Because Unix seconds represent time according to the Coordinated Universal Time (UTC) standard, timezone and daylight saving time issues were also avoided. In the smartphone app, Unix time is translated to human-readable time only when it needs to be displayed to the end-user.

4.2.3 Bluetooth and Logger to App API

A communication scheme was established to facilitate the transmission of data between the Version 2 logger and the WiSAT app. In Bluetooth Low-Energy communication, “Master” refers to those devices which initiate and manage connections, and “Slaves” are the devices that accept the connection from the Master [69]. The Master device typically has more resources than slave devices[69]. For this reason, WiSAT was designed such that the smartphone always serves as master, and, therefore, initiates the connection with the logger, which serves as the slave device.

Two modes of communication were conceived. First, a “batch” mode was created to address WiSAT’s default use-case, in which data is transmitted regularly during the normal operation of the system. In this mode, the logger will attempt to send the oldest JSON batch file in the READY directory to the smartphone whenever it receives a “getnext\n” string from the app. Upon receipt of the complete batch file, the smartphone deserializes the file and saves it to the app’s database. If no errors are encountered in this process, the phone issues a command to the logger: “received?batch=[batchID]\n”, where the “batchID” argument is substituted with the specific Batch ID number from the header information of that JSON batch. This command lets the logger know to move that file to the BACKUP directory. Afterwards, the app requests the next batch in the queue by issuing the “getnext\n” command again. The cycle repeats until the logger has transmitted all files in the READY directory. While there is an inherent time cost to this approach, this approach was chosen to minimize data loss as much as possible.

WiSAT’s second communication mode allows the app to receive data in real-time as the user is guided through setup and initialization. Known as “streaming” mode, this mode streams data continuously from the logger to the app, similar to the way that a Bluetooth headset system would stream music or other audio. WiSAT’s streaming mode issues the following command to the logger to tell the logger to begin streaming live data to the app: “stream?=true\n". Once all initialization data has been captured and streaming is no longer required, the app issues “stream?=false\n” to turn streaming mode off.

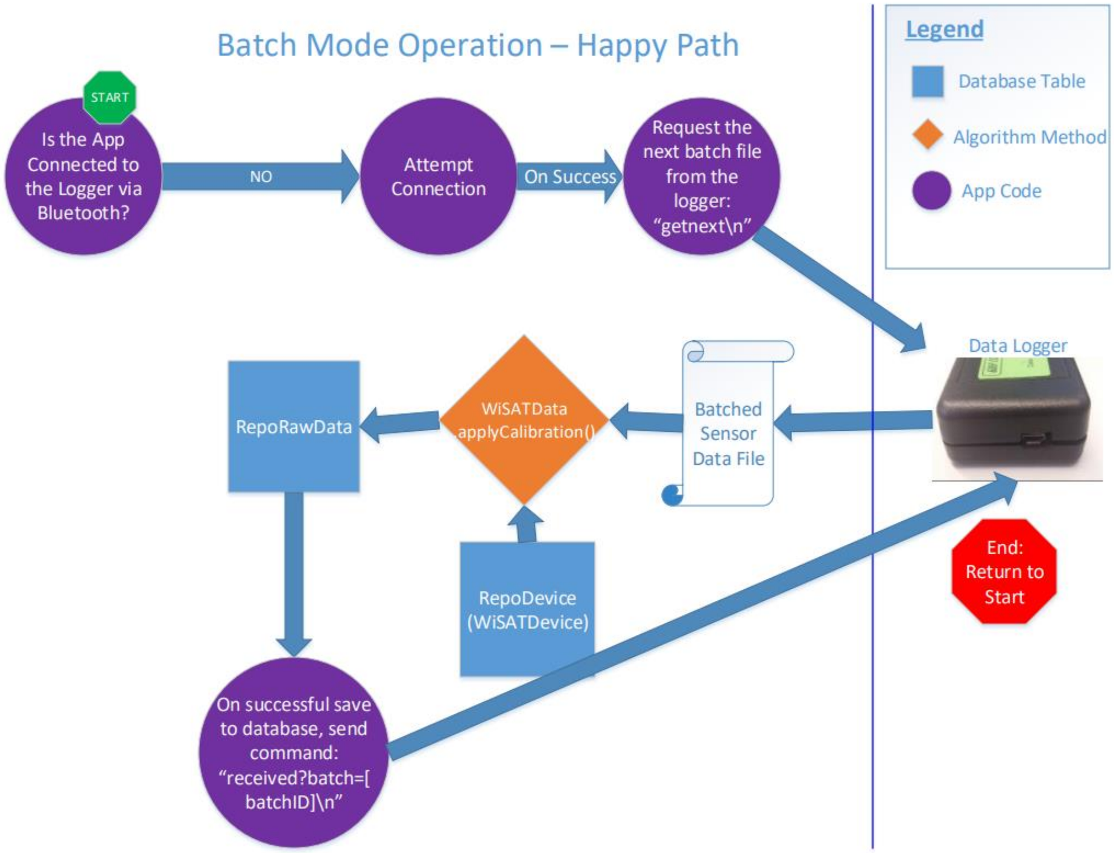

Batch Mode Operation – Happy Path
Legend
Database Table
Algorithm Method
App Code
START
Is the App Connected to the Logger via Bluetooth?
NO
Attempt Connection
On Success
Request the next batch file from the logger: “getnext\n”
Data Logger
Batched Sensor Data File
WiSATData applyCalibration()
RepoRawData
RepoDevice (WiSATDevice)
On successful save to database, send command: “received?batch=[batchID]\n”
End: Return to Start

Figure 20: Batch Mode Operation

A third command issued by the app resets the internal time on the logger. The data logger is subject to occasional clock resets due to firmware issues or external events. When a reset occurs, the time is set back to the system's default time (usually January 1, 2007). In Version 1, this error could be manually corrected by uploading a configuration file to the datalogger. Alternatively, it was sometimes possible to retroactively correct inaccurate timestamps programmatically during data anlaysis. In Version 2, this process was simplified by providing a command by which to set the logger's internal time: "time?set=[time]\n".The time variable is replaced with the current Unix timestamp. As a safeguard against occasional time resets, the app was programmed to issue this command upon connecting to a logger and once every hour afterwards.

4.2.4 Battery Evaluation

Two decisions were reached from battery evaluation. First, it was decided to place two 500 mAh lithium ion batteries wired in parallel inside the logger (Figure 21). This provided WiSAT with an estimated 75 hour maximum operation time before having to be recharged. This was the greatest time that could be expected without modifying the logger's form factor or PCB layout to fit higher capacity batteries. Second, these graphs informed the decision to switch the logger into "low-power mode" upon detecting a charge of 3500mv or lower. Low-power mode forces the logger to stop logging and all non-essential processes in order to maintain the time on the logger's internal clock. In Version 1, this mode activated when the logger dropped to 3200mv or lower, but Version 2 changed this, because the cost of a slightly shorter battery life was deemed worth the added safeguard of maintaining the logger's internal clock as long as possible.

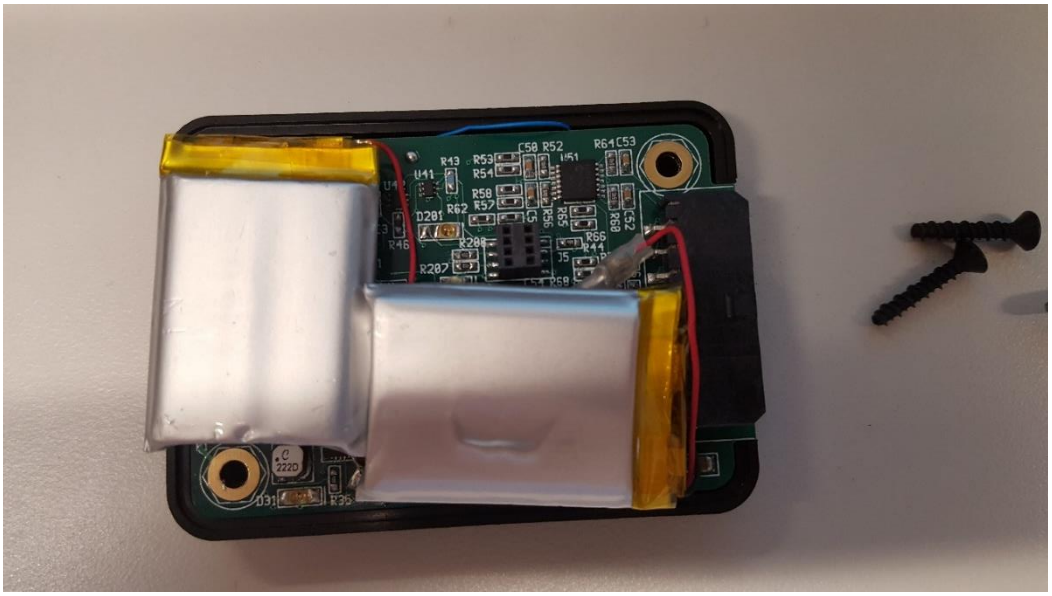


Figure 21: Battery Configuration

4.3 Specific Aim 3: Preparation of Algorithm Subsystem

4.3.1 Goal 1 – Algorithm Translation

The first goal of Specific Aim 3 was the translation of the algorithm subsystem from Matlab to object-oriented languages. The algorithm subsystem consisted of five main features:

Table 12: Algorithm Features

Component	**Function**
Calibration	Calibrates WiSAT Data according to specific hardware
Initialization	Creates Scaling Features based on the user's sitting behavior. Establishes reference points for upright posture, front leans, right leans, and left leans. These reference points are used in other classifiers in the algorithm library
Weight Shift Classifier	Identifies Weight Shifts from calibrated raw data.
In-Seat Movement Classifier	Identifies In-Seat Movement "points" from calibrated, raw data.
Occupancy Classifier	Identifies segments of data in which the user is seated.

This effort began with an assessment of the state of the algorithms prior to translation. Appendix J was created to document the milestones through which WiSAT data passes on its way to the Weight Shift Classifier. As seen in the visualization, Calibration and the Occupancy Classifier are included as milestones in this process. Additionally, the Scaling Features created by the Initialization function are utilized within several of the milestones. The In-Seat Movement Classifier follows a similar workflow with fewer milestones, so it was developed last.

As with the rest of WiSAT's development, this translation effort followed a theme of loose-coupling between the algorithm and app subsystem. To this end, it was decided that the algorithm subsystem should only have access to information and data that was explicitly passed to it by the app subsystem. This means the algorithms do not directly create, modify, or delete any data in the app subsystem, including the database layer. Although this constrained the algorithm subsystem, it also allowed this specific aim to satisfy its goal of keeping the algorithms modular and allowed the app and algorithms to be developed independently of each other.

4.3.2 – Algorithm Resilience

The second goal of Specific Aim 3 introduced constraints related to processing data continuously in a volatile, smartphone environment. Specifically, the algorithms needed to be resilient to both the app and logger being powered off. For example, when the logger is turned off, no data is logged. The next time that logger is powered on, the app would receive new data. In this scenario, a large time jump would exist between the last processed data and the new incoming data points. The algorithm would need to gracefully handle the time gap and recognize that a gap of several hours, for example, should not be reported by the occupancy classifier as a continuous in-seat session.

A second, very common scenario would occur whenever the app was closed by the user, crashed, or was terminated by some other means. Unlike the Matlab algorithms, the smartphone algorithms operate under the assumption that any information stored in volatile memory could be lost unpredictably at any point during the operation of the algorithm. Any data not returned to the rest of the app for storage in the database could be lost without warning.

In summary, constraints faced by the algorithm are as follows:

- The algorithms cannot directly read, write, or modify any data or information in the database or the surrounding app without it first being passed in to the algorithm library by the app
- Data coming into the algorithms is not guaranteed to arrive at a consistent rate or on a specific schedule. This is due to the unpredictability of the hardware and app subsystems being turned on and off, as well as Bluetooth connectivity issues.
- Gaps in the data may occur due to the hardware subsystem being turned off.
- Volatile memory may be lost at any time due to the app being closed.
- The algorithms should be able to handle sampling errors if the data logger produces an error where the sample rate exceeds or drops below four hertz.

Based on these constraints and the two goals of Specific Aim 3, the inputs, outputs, and potential error conditions were defined for the algorithm library. Three entry points (“methods” in object-oriented terminology) were defined for access to the five features described by Table 12.

Prior to saving the data to the mobile database, the calibration method adjusts the data. It is the only method which directly receives the actual “raw” data from the logger. This was done to minimize the risk of uncalibrated data or twice-calibrated data entering the rest of the algorithm library, and ensured that only calibrated data was to be stored in the database. Therefore, the rest of the system could assume that any data retrieved from the database was calibrated and ready for use. Additionally, a database table was added to the schema to store the calibration parameters entered by the user for their specific hardware. A “WiSATDevice” object was defined to convey data from this table to the Calibration method.

Second, initialization was separated from the rest of the algorithms to allow it run in tandem with the logger’s “streaming” mode. A single entrypoint, the “processIntialization” method, accepted four lists of WiSATData objects representing the upright, front lean, left lean, and right lean data. An AllScalingFactors object was created to store the outputs of the initialization algorithm, which was then stored in a database table, “ScalingFactors”. Figure 22 illustrates this workflow.

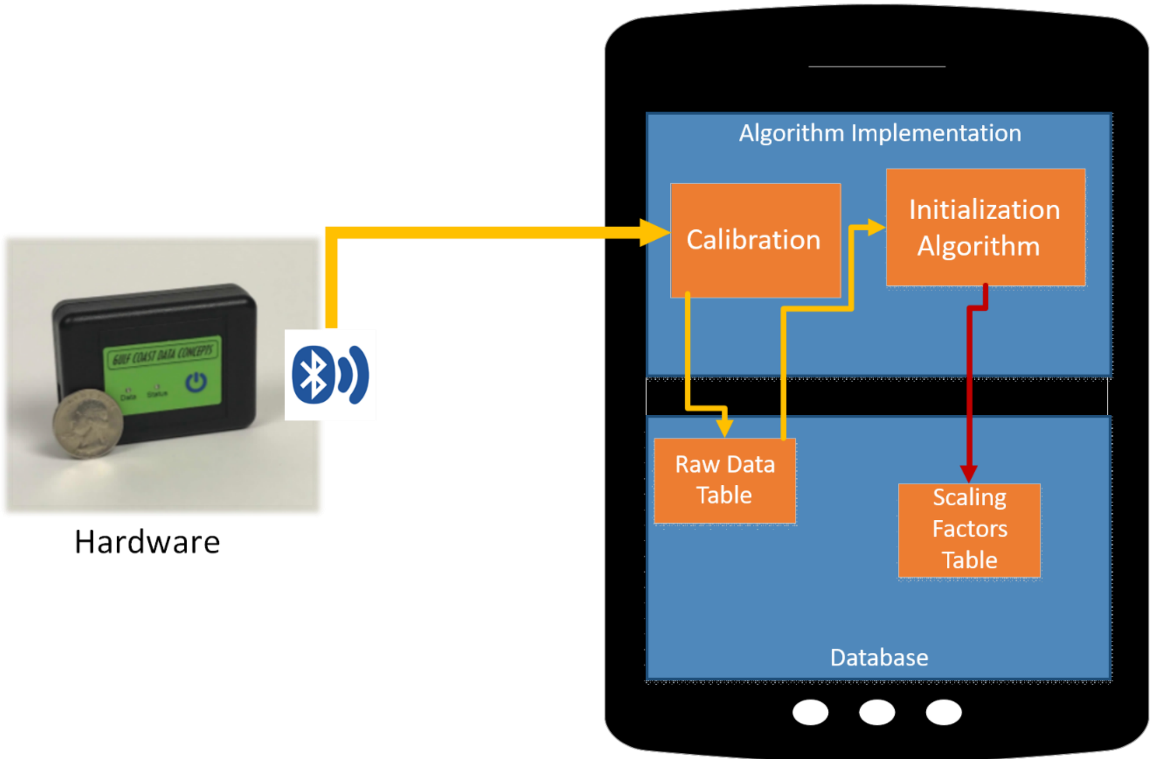


Figure 22: Initialization Algorithm Data Flow

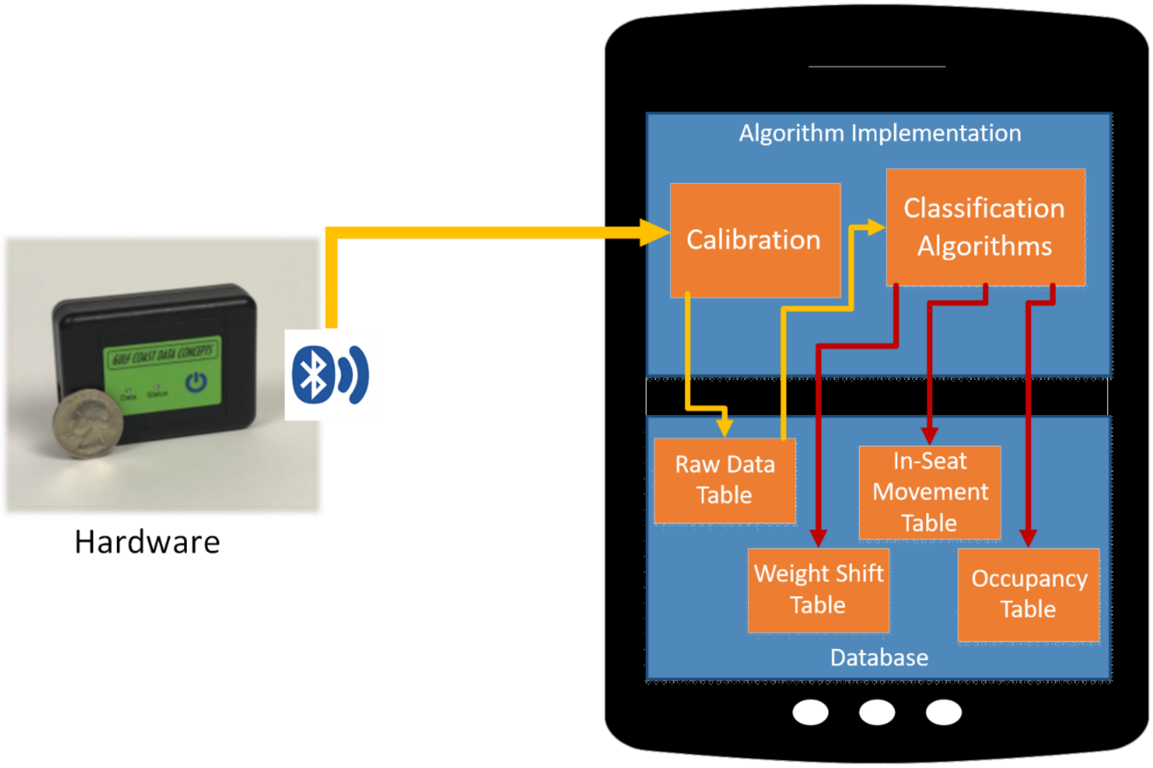


Figure 23: Batch Mode Algorithm Data Flow

Last, the Occupancy, Weight Shift, and In-Seat Movement classifiers were combined in a single method. Unlike the initialization algorithm, this method was intended to receive and process data collected during the logger's "batch" mode. Named "addOccupancyDataPoint", this method receives a single WiSATData point at a time. The app vendor was instructed to pass in unprocessed data from the RepoRawData table from oldest to newest. During processing, this method would check for Occupancy, Weight Shifts, and In-Seat Movements utilizing the milestones outlined in Appendix J.

Because these three events occur over longer time spans than the amount of time represented by a single datapoint, a "Singleton" object was leveraged to track the algorithm's state over time. In the Singleton architectural pattern, exactly one instance of a given object, known as the Singleton, is instantiated for the entirety of the application's runtime. In WiSAT, this meant that this Singleton object was able to store information across multiple calls to addOccupancyDataPoint. As each WiSATData entered addOccupancyDataPoint, the singleton tracked whether the user was in the chair, or if the data was part of a Weight Shift or In-Seat Movement.

When enough data was collected in the Singleton to determine that a Weight Shift or In-Seat Movement had completed, the start and end time was saved in a Weight Shift or In-Seat Movement object and added to a list in the Singleton. When the algorithm determined that the user had left the seat, the start and end time of that occupancy session was recorded in an occupancy object. At this point, the lists of Weight Shifts and In-Seat Movements, and the occupancy object were packaged into DataPointHistory and returned to the rest of the app. Additionally, a list of identifiers (the primary key from RepoRawData) of all WiSATData processed during that occupancy session were bundled in DatapointHistory.

The app developer was instructed to retrieve the information from the returned DatapointHistory object and save to the appropriate database tables. The WiSATData identifiers were used to mark the datapoints as processed in the RepoRawData table.

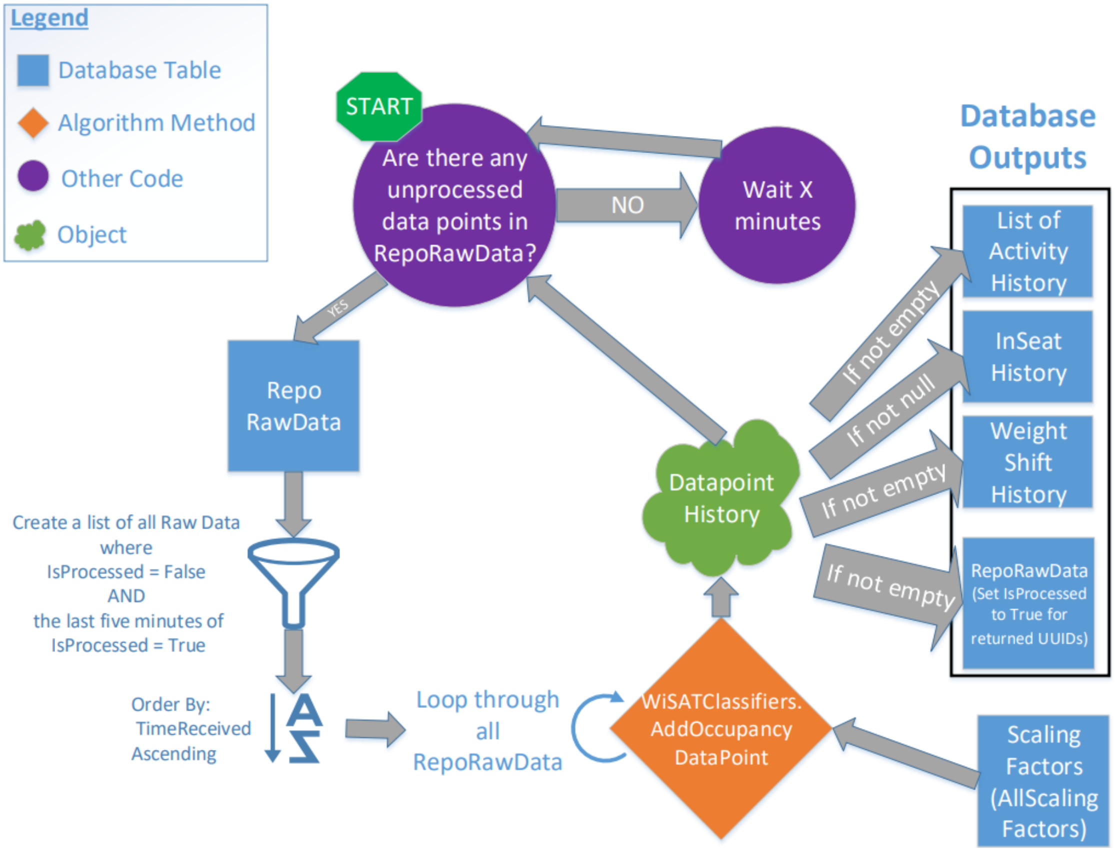
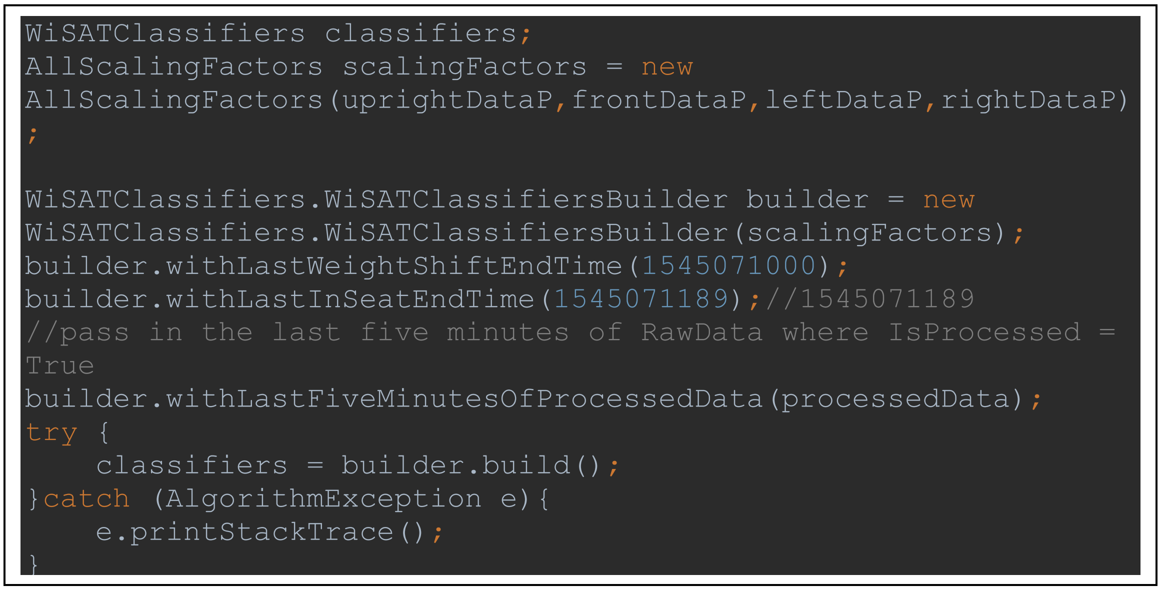

Legend
Database Table
Algorithm Method
Other Code
Object
START
Are there any unprocessed data points in RepoRawData?
NO
Wait X minutes
YES
Repo RawData
Create a list of all Raw Data where IsProcessed = False AND the last five minutes of IsProcessed = True
Order By: TimeReceived Ascending
Loop through all RepoRawData
WiSATClassifiers. AddOccupancy DataPoint
Datapoint History
Scaling Factors (AllScaling Factors)
Database Outputs
If not empty
If not null
If not empty
If not empty
List of Activity History
InSeat History
Weight Shift History
RepoRawData (Set IsProcessed to True for returned UUIDs)

Figure 24: Classification Algorithm Workflow

Because the three metrics would only be returned at the end of an occupancy session, a 15-minute cap was added to prevent long occupancy sessions from delaying the return of data to the app. When the 15-minute limit was reached, the algorithm would automatically return DataPointHistory. When the next datapoint was passed to the method, the next occupancy session would begin, and the algorithm lifecycle would repeat.

This workflow allowed the algorithm library to maintain information in volatile memory across calls to addOccupancyDataPoint without the algorithm subsystem requiring access to the app's database. Additionally, data could be passed into the algorithm continuously without needing to be divided into segmented chunks. However, this workflow did not address the need for the algorithm to maintain resiliency across the unpredictable restarts of the app. Further, there was a need to pass some information, such as the Scaling Features from the Initialization algorithm, into the classification algorithm only one time at startup and never again until the next time the app was launched.

To address both these concerns, an architectural pattern known as the Builder Pattern was used. According to this pattern, a WiSATClassifiersBuilder class was defined which could be instantiated once at startup. Scaling Features and other parameters could be passed into the builder object once. Afterwards, a special build() method in the builder object setup the Singleton object and returned the WiSATClassifiers object, which contained the addOccupancyDataPoint() method.



```
WiSATClassifiers classifiers;
AllScalingFactors scalingFactors = new
AllScalingFactors(uprightDataP,frontDataP,leftDataP,rightDataP)
;

WiSATClassifiers.WiSATClassifiersBuilder builder = new
WiSATClassifiers.WiSATClassifiersBuilder(scalingFactors);
builder.withLastWeightShiftEndTime(1545071000);
builder.withLastInSeatEndTime(1545071189);//1545071189
//pass in the last five minutes of RawData where IsProcessed =
True
builder.withLastFiveMinutesOfProcessedData(processedData);
try {
    classifiers = builder.build();
}catch (AlgorithmException e){
    e.printStackTrace();
}
```

Figure 25: Builder Pattern Used to Create WiSAT Algorithms

Because the builder object was run at startup, it was used to provide historical context to the classification algorithm without the need to pass the historical information in each time the method was executed. Besides Scaling Features, the app vendor was instructed to pass in three parameters to the builder object: the last five minutes of processed data from

RepoRawData, the last weight shift end time, and the end time of the last time the user was in the chair. Each time the app restarted, these parameters ensured that the classification algorithm was “smart” enough to capture weight shifts and in-seat movements that started prior to the first unprocessed datapoint, as well as whether or not the previously processed data segment captured the user in or out of their chair. This was important because some classification rules applied to data captured within the first two minutes of occupancy. For example, the Weight Shifts occurring in these first two minute would not be counted.

No WiSATData was permanently lost when the app restarted. Any information stored in the Singleton was lost when the app was closed, so the algorithm would restart from the last unprocessed datapoint when the app started again. WiSATData was only marked as processed when the DataPointHistory object was returned. Therefore, a natural checkpoint was built into the algorithms. Since DataPointHistory was returned at least once every fifteen minutes, less than fifteen minutes worth of data would be lost from the Singleton each time the app closed. Upon restarting, the app algorithm would start from the oldest datapoint that had not yet been processed.

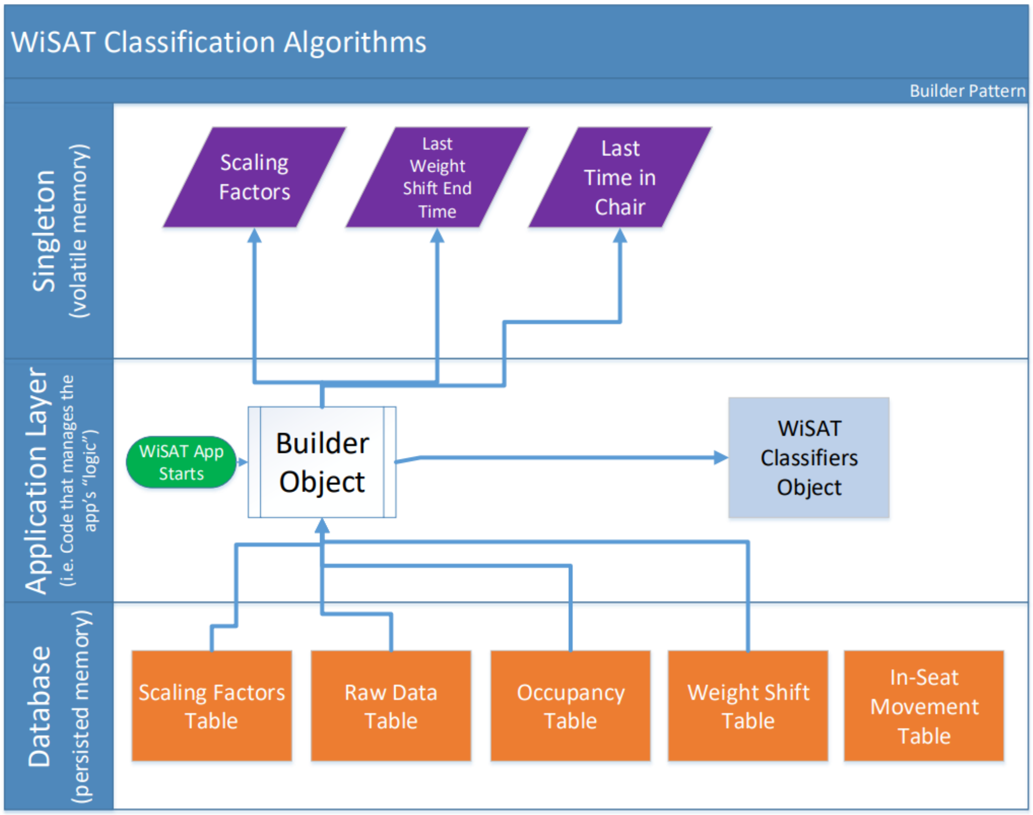


Figure 26: Builder Pattern Visualization

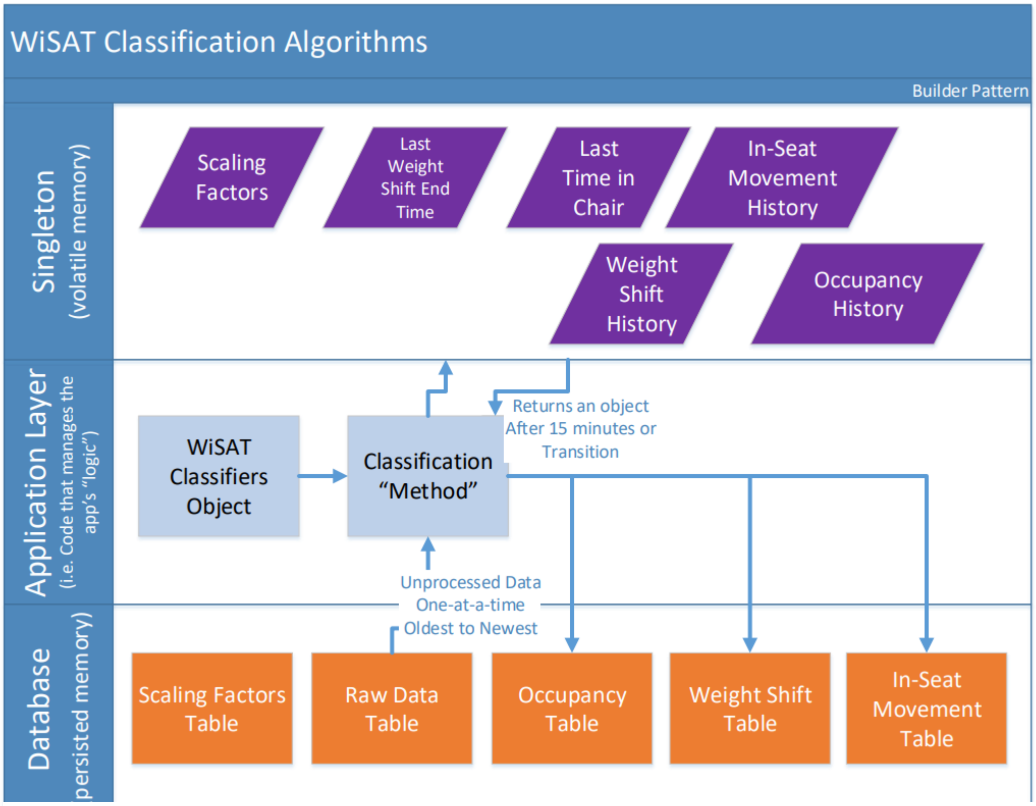


Figure 27: Singleton to Database Visualization

4.4 Specific Aim 4: Integration of the WiSAT Subsystems through Multi-layered Architecture for the WiSAT Smartphone App

For the purposes of this specific aim, the app's architecture is conceptualized as a basic, three-layered system (see Figure 28). First, the UI layer is responsible for capturing user input and rendering the app's screens and design features. The database layer stores all persisted data in the application, including raw data, processed data (occupancy, weight shift data, etc), error logs, and more. Last, the application layer serves as a middleman between the two. It is responsible for most of the internal application logic and the interface between the algorithms and the rest of the app.

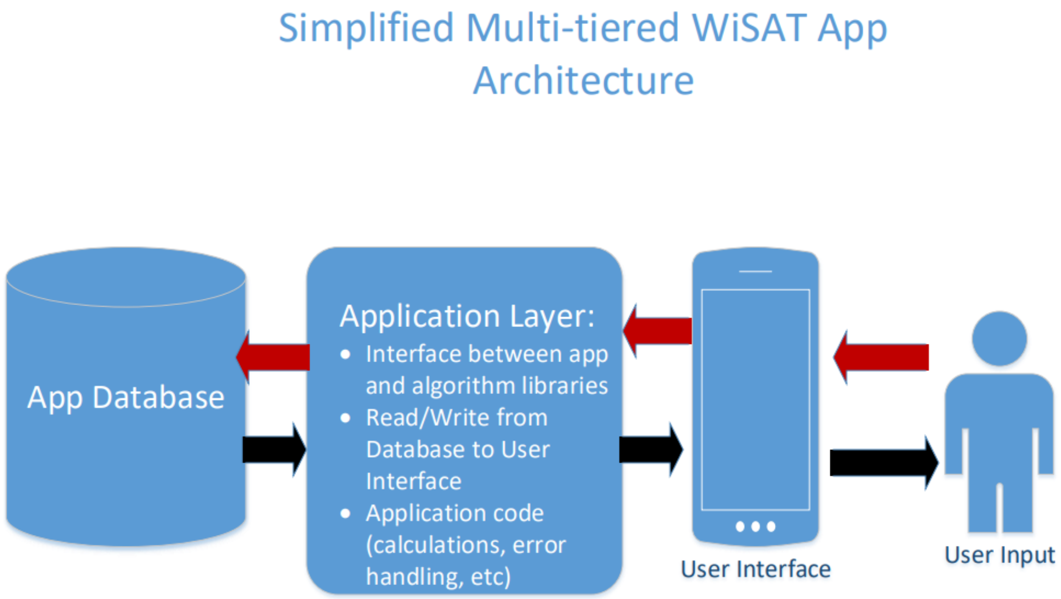

Simplified Multi-tiered WiSAT App Architecture
App Database
Application Layer:
Interface between app and algorithm libraries
Read/Write from Database to User Interface
Application code (calculations, error handling, etc)
User Interface
User Input

Figure 28: WiSAT App Architecture

As a whole, WiSAT's subsystems integrate with the smartphone app as visualized in Figure 29. The hardware subsystem communicates with the app using Bluetooth Low-Energy and the API defined in Specific Aim 2. Within the app, the application layer, sends API calls to the datalogger and accepts incoming data. It is also responsible for deserializing data from JSON and saving it to the database. However, before saving to the database, the application layer passes deserialized data to the algorithm library's calibration feature. The application layer is also responsible for pulling saved, calibrated data back out of the database and passing it into the algorithm subsystem, as described in Specific Aim 3. Outputs of the algorithm subsystem are received in the application layer and returned to the database. Periodically, the application layer retrieves unsynced data from the database, transmits it to the remote server subsystem, and marks it as synced. Finally, all user input is passed from the UI layer into the application layer to the database, and all data displayed in the UI layer is passed from the database to the application layer to the UI layer, where it is viewed by the user.

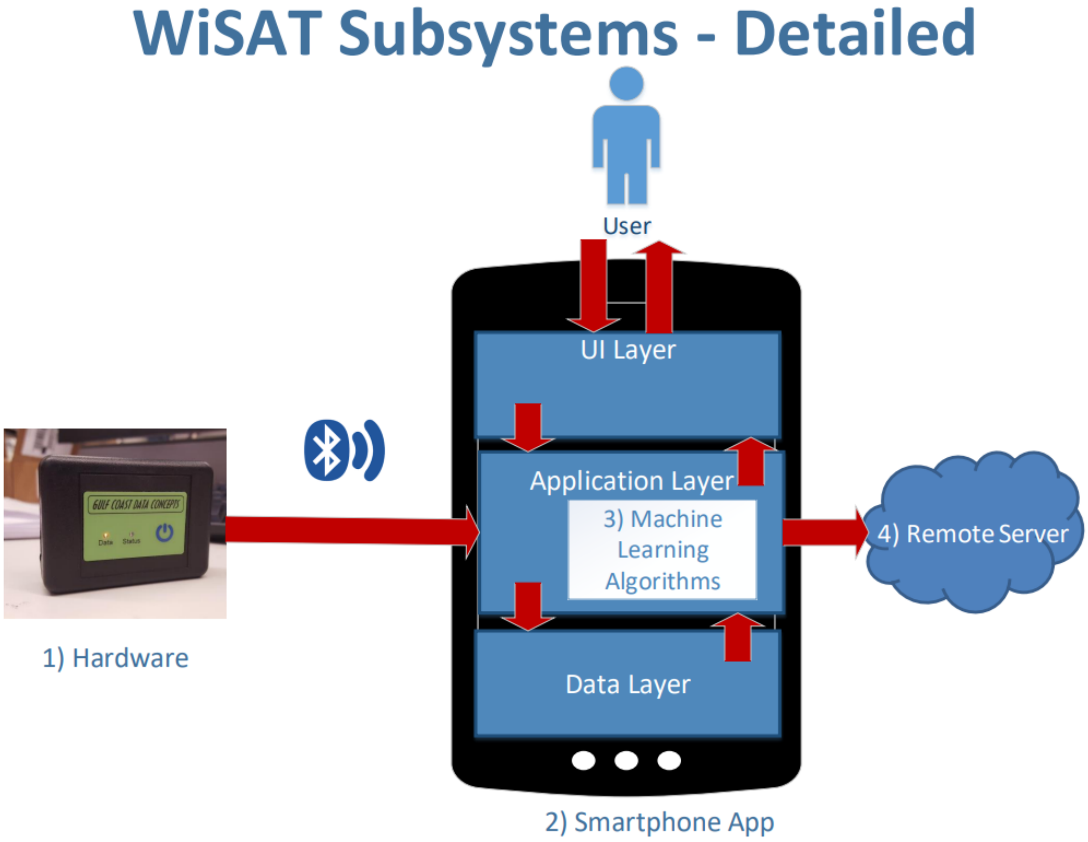


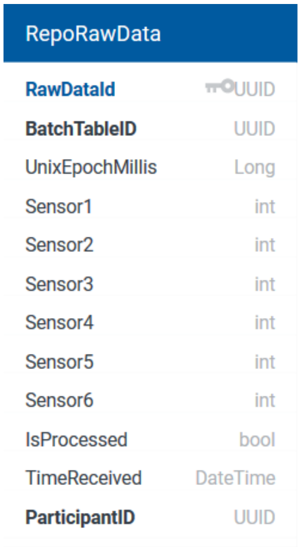
Figure 29: WiSAT Subsystems - Detailed

In the UI layer, the system requirements were defined a set of features that were rendered as a UI prototype in Specific Aim 1. The prototype screens were used to create a UI flow diagram to illustrate how users would interact with the system. This drove much of the functionality that was implemented by the other two layers.

4.4.1 Database Layer

A database schema was created for the app's database layer with the goal of creating a location to "persist" WiSAT data. Here, the term "persist" refers to the ability of the database to maintain data after the app is closed, as opposed to data in "volatile" memory, which is lost when the app is closed. The schema was designed and visualized as a traditional, relational database. However, it was *implemented* using a third-party, object-oriented database library, "Realm Database". Several benefits were realized from this approach. First, the formal schema provided a familiar, common basis of communication across the various teams responsible for the implementation of WiSAT. For example, the app developer and API developer implemented their respective subsystems in isolation from each other using different technology stacks. Having a common schema helped convey data requirements to both teams without needing to delve into implementation details in each team's respective domains.

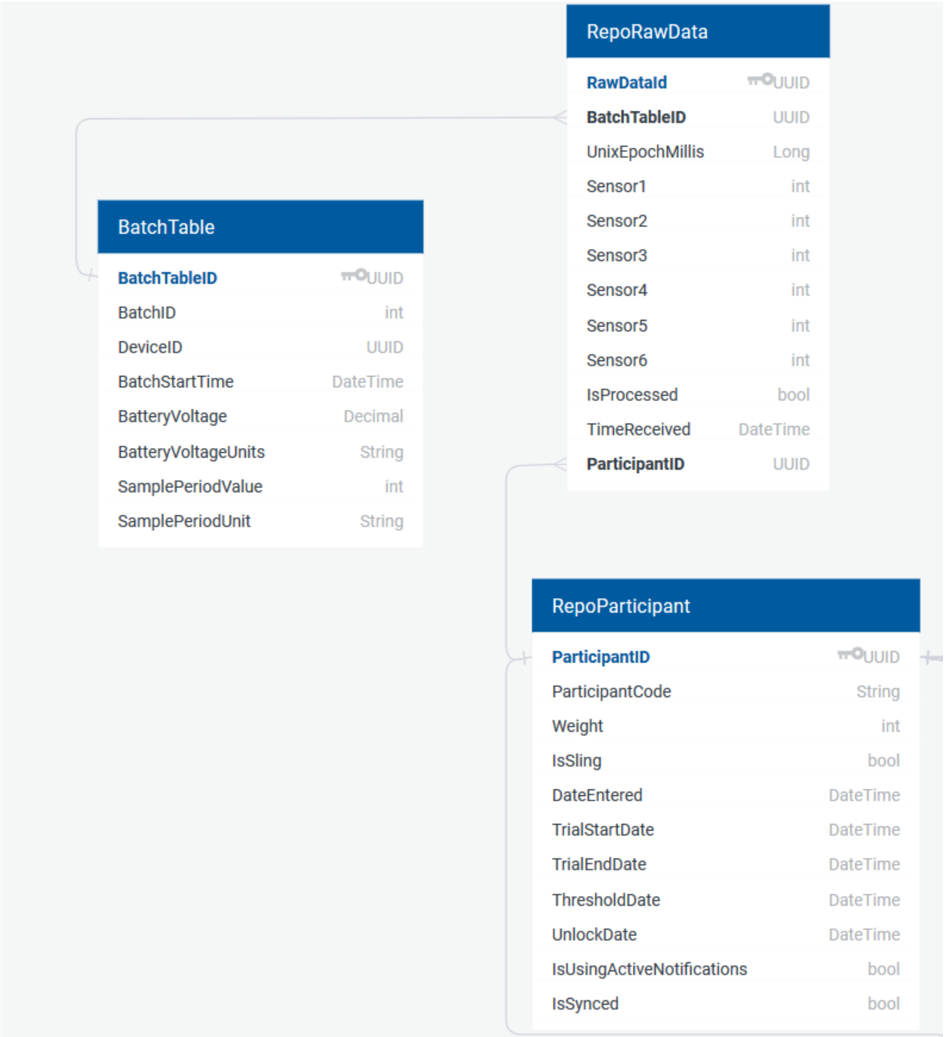
Similarly, the schema helped ensure consistency across the data objects in each subsystem. For example, the logger, algorithm, and app subsystems each contained a WiSATData object. This object represented a single sample of pressure mat data. WiSATData consists of a timestamp and six force sensor readings (one for each of the sensors in the sensor mat). In each of these subsystems, WiSATData was stored differently. In the data logger, it was stored as an object inside a JSON batch file (see Specific Aim 2). In the app, it was stored via Realm Database. The algorithms did not permanently store this object, but they used this object as an argument in several of their methods. The schema provided a reference with which to ensure that all important information about WiSATData was captured by the app, as seen by the "RepoRawData" table visualized in Figure 30.



RepoRawData	
RawDataId	🔑UUID
BatchTableID	UUID
UnixEpochMillis	Long
Sensor1	int
Sensor2	int
Sensor3	int
Sensor4	int
Sensor5	int
Sensor6	int
IsProcessed	bool
TimeReceived	DateTime
ParticipantID	UUID

Figure 30: Raw Data Table from Database Schema.

Representing data in a relational model provided a structure which satisfied WiSAT's requirement for consistent, accurate data. Foreign key relationships helped ensure that each data could be tied to relevant metadata. For example, any given datapoint in RepoRawData could be linked to its batch file of origin through the BatchTableID foreign key (Figure 31). Primary keys were utilized to ensure that every piece of database data was uniquely identifiable – even across subsystems. For example, the RawDataID used in the database could be used to identify the same data sample in both the algorithm libraries and the smartphone app. Although not a relational database, Realm provides features which allowed for the enforcement of these constraints.



RepoRawData	
RawDataId	UUID
BatchTableID	UUID
UnixEpochMillis	Long
Sensor1	int
Sensor2	int
Sensor3	int
Sensor4	int
Sensor5	int
Sensor6	int
IsProcessed	bool
TimeReceived	DateTime
ParticipantID	UUID

BatchTable	
BatchTableID	UUID
BatchID	int
DeviceID	UUID
BatchStartTime	DateTime
BatteryVoltage	Decimal
BatteryVoltageUnits	String
SamplePeriodValue	int
SamplePeriodUnit	String

RepoParticipant	
ParticipantID	UUID
ParticipantCode	String
Weight	int
IsSling	bool
DateEntered	DateTime
TrialStartDate	DateTime
TrialEndDate	DateTime
ThresholdDate	DateTime
UnlockDate	DateTime
IsUsingActiveNotifications	bool
IsSynced	bool

Figure 31: Foreign Key Relationships of RepoRawData

Implementing the database through Realm Database provided benefits of its own. As an object-oriented database, the app developer never needed to “translate” from Java/Swift objects to SQL statements, nor did they need to manually build and maintain a separate database. Instead, they utilized the Realm API to perform all database interactions through their platform’s native development language. Object-Relational Mappers (ORMs) also allow developers to avoid this translation between SQL and object-oriented languages. However, ORMs operate as an additional layer between the database layer and the application and, therefore, require additional processing time and system resources. As an object-oriented database, Realm bypasses this constraint by storing the data as objects and minimizing data conversion [70].

4.4.2 Application Layer

Many specifics of the application layer's implementation were left to the app developer. However, two important exceptions to this exist. First, is the implementation language which the developer was instructed to use. Although WiSAT is designed to be a cross-platform app, the decision was made to develop both the Android and IPhone apps in Java and Swift, respectively. Alternative, cross-platform solutions, such as Xamarin, were considered. However, due to the app's dependence on Bluetooth, it was decided to remove the extra layer of abstraction between the smartphone hardware and the app that would be added by the cross-platform development languages.

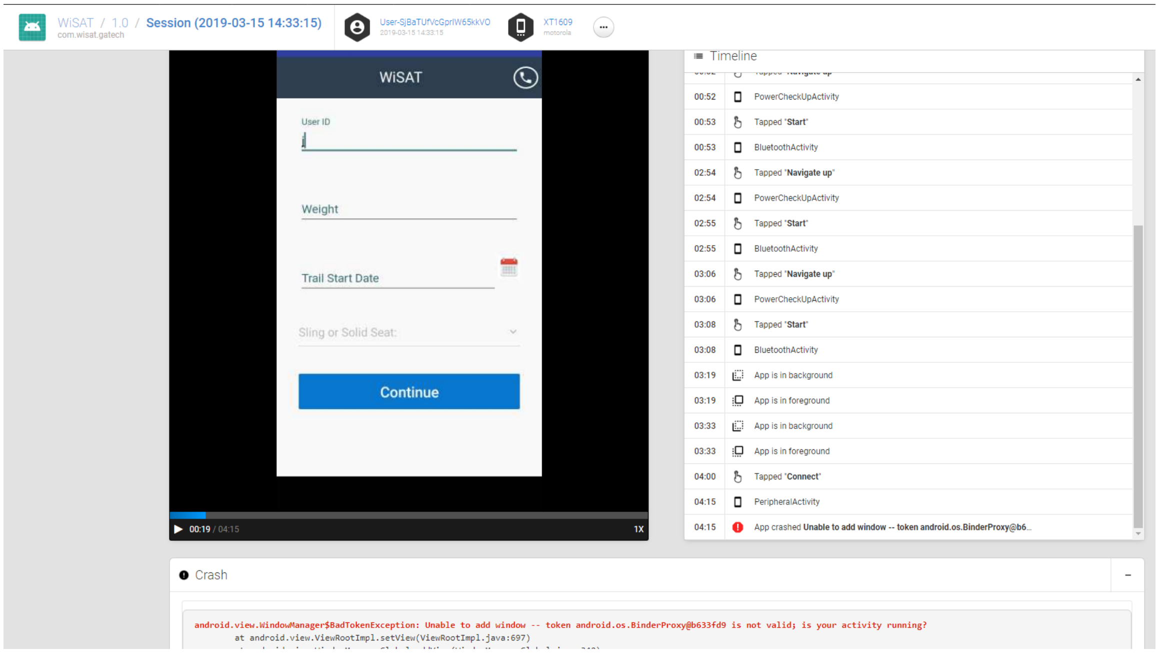
Background processing was another area of the application layer which was dictated by the surrounding system requirements. Here, background processing refers to code run "behind-the-scenes" regardless of whether or not the user is actively engaged with the app. As an activity tracker, WiSAT is designed to collect and process data from the logger whenever the logger is active regardless of whether or not the app is open. Further, active notifications are not be constrained to the lifecycle of the app. As such, the developer was instructed to use background processing to ensure that the Bluetooth communication, algorithm execution, and push notifications would not depend on the user leaving the app open on their smartphone.

4.4.3 Quality Assurance

As in Specific Aim 2, evaluation of the app's architecture was conducted against both individual components and integrated parts. This is captured in the test cases outlined in Appendix G. Some test cases were tested in isolation. For example, "The Weight Shift Screen shall show daily data" was evaluated using test data to isolate the testing of the UI layer from those tests aimed at the algorithms responsible for generating that data.

To facilitate this style of testing in the database layer, the app developer added a feature to the app to export the database to the smartphone's file system. This provided the means to validate the database structure and pinpoint the source of errors. For example, during one evaluation, the UI layer was not displaying Weight Shift or In-Seat Movement score information. An exported database was examined, and the source of the error was quickly identified as stemming from an absence of data in the RepoRawData table. As the issue was investigated further, it became apparent that the issue was due to a Bluetooth communication issue and had nothing to do with either the Weight Shift algorithm or the UI layer.

Issues in the application layer were primarily diagnosed through TestFairy, a mobile testing service which captures user interactions with the app in real-time, collects diagnostic information regarding the host smartphone, and records application logs (Figure 32). As tests were conducted against the app, the corresponding TestFairy session was monitored. Application logs were used to diagnose crashes. Screen interactions were utilized to quickly communicate bugs, crashes, and feature requests to the app developer.



00:52	PowerCheckUpActivity
00:53	Tapped "**Start**"
00:53	BluetoothActivity
02:54	Tapped "**Navigate up**"
02:54	PowerCheckUpActivity
02:55	Tapped "**Start**"
02:55	BluetoothActivity
03:06	Tapped "**Navigate up**"
03:06	PowerCheckUpActivity
03:08	Tapped "**Start**"
03:08	BluetoothActivity
03:19	App is in background
03:19	App is in foreground
03:33	App is in background
03:33	App is in foreground
04:00	Tapped "**Connect**"
04:15	PeripheralActivity
04:15	App crashed **Unable to add window -- token android.os.BinderProxy@b6...**

```
android.view.WindowManager$BadTokenException: Unable to add window -- token android.os.BinderProxy@b633fd9 is not valid; is your activity running?
        at android.view.ViewRootImpl.setView(ViewRootImpl.java:697)
```

Figure 32: Example TestFairy Session

The fully integrated system was evaluated by collecting data from a loaded sensor mat through a logger and pairing the logger to the WiSAT app.Results of this testing were visible in the app's UI, the mobile database, and the remote server.

CHAPTER 5. CONCLUSION AND FUTURE WORK

Through the product development effort described in this thesis, the WiSAT system is prepared for the next steps in its journey to becoming a complete, consumer product. To further evaluate the system, as a whole, a second stage usability study is planned to complement the evaluation already conducted in this thesis. During this study, wheelchair users will be recruited to use the WiSAT system during their daily routines for a period of one week. Results of this study will be used to improve usability and uncover any outstanding issues with the system.

Following this study, WiSAT will be deployed in a pre-clinical trial with third-party research teams from the Hines Veterans Administration Hospital and the University of Pittsburgh. Participants in this trial will use WiSAT in their daily routines over the course of approximately six months. This trial will focus on the clinical effectiveness of WiSAT, as measured by its acceptability, usability, and whether or not it impacts behavioral change. Specific attention will be paid to the passive and active feedback of the WiSAT system during this trial. Results will be used to further improve WiSAT.

Finally, the end-goal for WiSAT is to be deployed as a commercial product. Recruitment of firms to license and manufacture WiSAT is ongoing. Once an interested party is identified, the system may be tailored to their specific needs easily thanks to the modularity and loose-coupling employed throughout its design.

Based upon the iteration of WiSAT produced by this thesis, a third, production-ready version of WiSAT, “V3”, may be prepared. V3 will be produced in the context of information learned from the second-stage usability study, the pre-clinical trial, and the specific requirements of the commercial licensee. However, several steps may be taken to further prepare WiSAT for deployment in a production environment. Additionally, optional features and enhancements exist for each of the four subsystems.

5.1 Hardware Subsytem

Table 13 compares the current state of WiSAT's hardware system with features that may be added for V3. Many of the recommendations for V3 are based on lessons learned during the course of this thesis. For example, the recommendation to use an established, real-time operating system (RTOS) is based on past bugs and firmware issues uncovered during Specific Aim 1. Introducing a real-time operating system will reduce the risk, complexity, and expense associated with developing and maintaining custom, proprietary firmware to accomplish tasks that are handled automatically by the RTOS. Further, using an established solution increases the likelihood that solutions exist to any problems that may be encountered. If an appropriate RTOS is selected, WiSAT will see improvements in multitasking, file system implementation, and hardware communication.

In order to host a more computationally expensive features, such as an embedded database and more advanced data pre-processing, a more powerful microcontroller is desirable. One option would be to upgrade the current microcontroller from the Atmel SAM3S (Cortex-M3) processer to a slightly more advanced one, such as the Atmel SAM4S (Cortex-M4). In this scenario, additional RAM could be added to the board to support the minimum requirements for the database and pre-processing algorithms. However, most, if not all, of the electronic components on the existing board could be maintained.

A more extreme option would involve an overhaul of the existing hardware in favor of implementing an inclusive system-on-module, such as the Qualcomm DART-SD410. This option would utilize a more powerful microprocessor instead of a microcontroller and opens the possibility of deploying a more advanced operating system, such as Android, Linux, or even Windows.

An inductive charging module is currently planned for V3. This module will allow the user to recharge the data logger wirelessly through their wheelchair cushion cover. This is intended to increase WiSAT's usability. In V2, the user is required to remove the logger from their cushion entirely before charging via a micro-USB port. In V3, wireless charging will allow the user to leave their hardware system in-tact while charging.

Table 13: WiSAT Hardware Technical Specifications

	V2 Specifications	**V3 Design requirements and/or considerations**
Logger Enclosure		
Form factor	Polycase KT-40 enclosure: 2 ¾ x 1 7/8 x ¾"	An attachment will be added to the case to allow for the logger to be paired with an inductive charger. Specific Form Factor choices may vary based on the desires of the licensee. However, smaller form factor is, generally, more desirable.
Case Connectors	Seat sensor mat and data logger use Clincher 67516-208LF and Clincher 95736-108LF connectors. An optional extension cable ("pig-tail") allows increased wire length between the sensor mat and the datalogger.	No Change
Electronic Components		

Processor	Microprocessor: Atmel SAM3S4B (Cortex M3 family) Random Access Memory (RAM): On-Chip RAM provided with SAM3S4B	A more powerful processor opens up the possibility of improved data storage options and more on-board processing Considerations: Atmel Sam4S, Qualcomm DART-SD410
Battery life	Two 3.7V 500 mAh Lithium-Ion batteries in parallel (~75 hours) Make/Model: Powerstream’s GMB652535-PCB	An additional battery should be added to power the real-time clock separately.
Indicators and on-off switch	V2 & V3 WiSAT includes on-off switch & power indicator and data indicator	No Change
Pressure Sensors	Custom sensor mat made by Tekscan. Sensor Mat contains six force sensors on a flexible mat. The mat includes cut-lines to allow for users with cushions ranging from 14.5” to 20” to use WiSAT.	No Change
Inductive Charger	Not yet implemented	An inductive charging module will allow the user to charge the data logger wirelessly through their wheelchair cushion.
Data and Communication Protocols		
Communication	Bluetooth 4.0 Low-Energy (Telit 53330-02)	No Change
Data storage	8 GB SD Card to accommodate situation when phone and user are separated.	Flash Memory

Data Access/Transmission	Data stored in flat files and formatted as JavaScript Object Notation (JSON)	Data stored in a local, embedded database Considerations: SQLite
On-board processing	Currently does not calculate occupancy on logger. This is calculated in the app subsystem.	Occupancy and sensor drift on-board
User management	Instructions will be for user to charge daily. Mobile app can report battery level of device	No Change
System Status	System status (error states, battery life, etc) are appended to each batched JSON data file.	No Change
Firmware		
Firmware	Proprietary Gulf Coast Data Concepts firmware	Established, real-time operating system Considerations: Zephyr Project, FreeRTOS

5.2 App Subsystem

After WiSAT completes all usability and pre-clinical studies, additional features may be added to prepare the app for production. Depending on what information the licensee wishes to add, additional security features may become necessary. In Version 2, no personally-identifiable data or medical information is collected. However, if the licensee wishes to add more personalization to the app or integrate it with systems which collect medical information, WiSAT will need protection from outside attackers. Specifically, the licensee may choose to encrypt the mobile database to prevent unauthorized access or tampering. The trade-off associated with this is that it will make the data much more difficult to steal, but it will also make it very difficult to recover if separated from the app for debugging purposes.

Second, study findings may further refine WiSAT's design and features. With the introduction of new participants and many more hours of usage, new information may be learned which affects the design used in V2. Because of the loose-coupling of the app's subsystems, changes may be made to the app without impacting the other subsystems. For example, if icons or design elements on the UI are changed or re-arranged, the logger, algorithms subsystem, and remote server will remain unaffected.

As mentioned in Specific Aim 4, the app developer was instructed to process data and manage Bluetooth connections and push notifications through background processes. This introduced challenges due to background process restrictions on newer releases of Android. Starting with Android 8.0, Android restricts certain kinds of background processing while the app is idle [71]. This is done in an attempt to restrict developers from consuming excessive smartphone resources without the user's knowledge. Common workarounds include the use of "foreground services", which essentially use an overlay on the phone's lock screen to make the user aware of apps that utilize background processes. WiSAT collects and processes data at regular intervals throughout the day, regardless of whether the user is actively using the app or not. Therefore, the app developer was required to implement processes which relied upon background processing through a combination of foreground services and other techniques to maintain background processing throughout the time that the smartphone is turned on. As rules and limitations regarding background services continue to evolve, WiSAT's future licensee may have to deploy updates or changes in this area of the app code.

Last, WiSAT's future licensee will face the choice of keeping WiSAT a primarily offline app or introducing more remote server integration. As in V2, keeping WiSAT a primarily offline system means that users may continue to use WiSAT regardless of their cellular data plan or location. However, this comes with the added burden of processing all data locally within the app subsystem. If more remote server integration were introduced, the algorithm subsystem could be moved to the remote server, and the local database's size could be greatly reduced. For the user, this means that fewer system resources would be consumed (battery life, memory, disk space), but the app would no longer be as up-to-date when they were without internet access. As more consumers adopt data plans and coverage increases, however, this may become less and less of a concern as time passes. Ultimately, the licensee must decide whether this is a trade-off they are willing to make.

5.3 Algorithm Subsystem

In the algorithm subsystem, future work could include expanded data validation. Currently, the algorithm subsystem includes an "Algorithm Exception" Class. This class is intended as a parent class to the various error states that the algorithms may detect. For example, the algorithm subsystem throws an exception if it receives raw data out of chronological order (i.e. the most recent data point is older than the previous one). However, many other opportunities exist for automated error-detection from within the algorithm subsystem. For example, it may be possible to implement a feature which automatically detects when the user should be prompted to redo their setup and initialization if their system ever falls out of calibration. A sub-class could be created that inherits from AlgorithmException. Since the app already catches all exceptions of type AlgorithmException, this would automatically be picked up by the application layer and would be logged to the database. Additional updates to the application code could introduce more custom behavior, such as custom prompts to encourage the user to redo their initialization.

A second area of improvement for the algorithm subsystem lies with introducing new algorithm updates based on findings from data collected from research study participants and early adopters of the WiSAT system. As more data is collected, the opportunity exists to further tune the classification algorithms to further improve their accuracy. Because the algorithm subsystem was built as a separate module from the rest of the app, any changes made to the algorithms can be deployed without altering other areas of the application code.

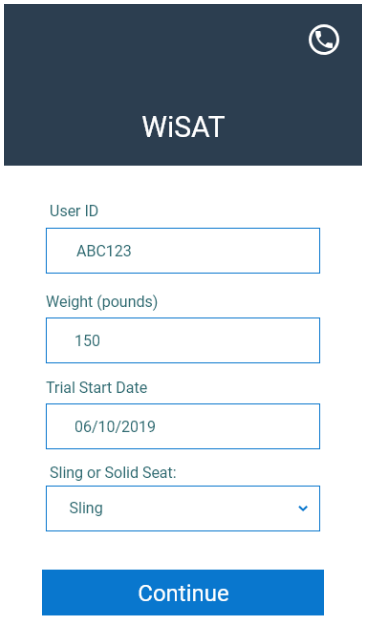
5.4 Remote Subsystem

If the future licensee wishes to use some version of the remote server subsystem, WiSAT's V2 API will need updated security features before it is ready for production. While WiSAT is still in pre-clinical studies, the risk of unauthorized individuals posting or retrieving data is minimal. Compared to a production-ready, commercial system, a system which is deployed among fewer than one-hundred users for a period of less than a year is unlikely to be infiltrated. Additionally, in the unlikely event that this did happen, no personally-identifiable information, healthcare data, or other sensitive information is collected by the system. However, for a production-ready product, implementing this added security is a best-practice. As the number of users increase, the risk of unauthorized access increases, and the difficulty and expense of resolving a security breach also increases.

Currently, the remote server subsystem relies upon a single access key to authenticate incoming data and requests. All instances of the WiSAT app utilize this same access key. If this access key were to be leaked, fake data could be sent on behalf of any user in the system. Further, if this access key needed to be changed, all users would be impacted. Several trusted and reliable authorization strategies exist in the industry, but the specific choice for WiSAT will depend upon a variety of factors dependent upon the specific use-case of the remote server which the licensee desires. Regardless of which specific approach is selected, the goal of the selected authorization scheme is two-fold . First, it must to minimize the risk of unauthorized users sending or receiving data from the remote server subsystem. Second it should reduce the ability for a data breach of one user’s account to impact another user.

Through the preparation of WiSAT V2, many lessons were learned about transitioning a research tool into a consumer product. In Specific Aim 1, this thesis built upon prior work and a controlled UI evaluation to design an effective, usable interface. Specific Aim 2 coupled the algorithm and app subsystem through an API with two modes of operation for specific scenarios within the lifecycle of WiSAT. The algorithm library was translated into object-oriented languages and prepared for use in a smartphone app through Specific Aim 3. Last, Specific Aim 4 tied the four subsystems together around the smartphone app subsystem. Going forward, WiSAT is prepared for its next steps towards becoming a production-ready consumer product.

APPENDIX A: WISAT INITIALIZATION SCREENS



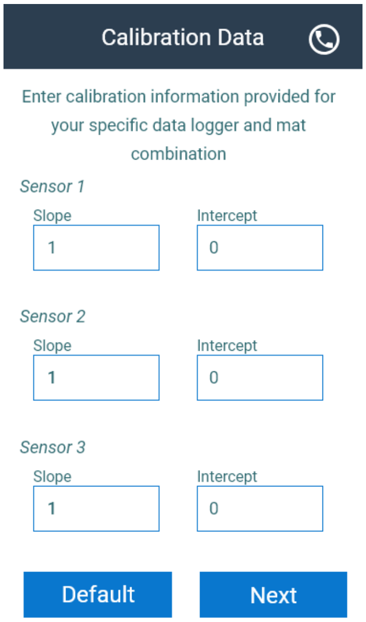
WiSAT

User ID: ABC123

Weight (pounds): 150

Trial Start Date: 06/10/2019

Sling or Solid Seat: Sling



Calibration Data

Enter calibration information provided for your specific data logger and mat combination

Sensor 1

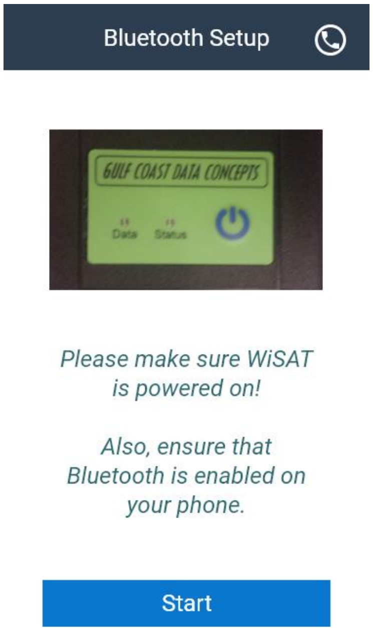
Slope: 1 Intercept: 0

Sensor 2

Slope: 1 Intercept: 0

Sensor 3

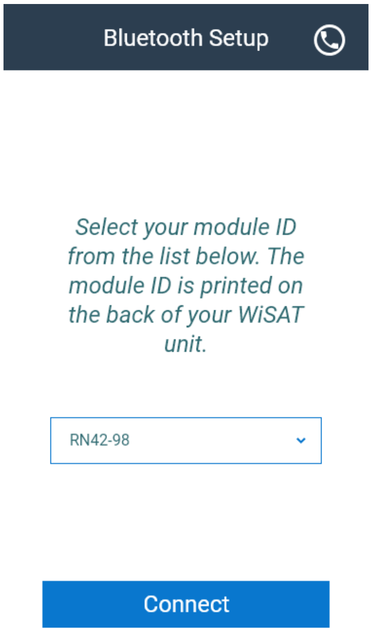
Slope: 1 Intercept: 0



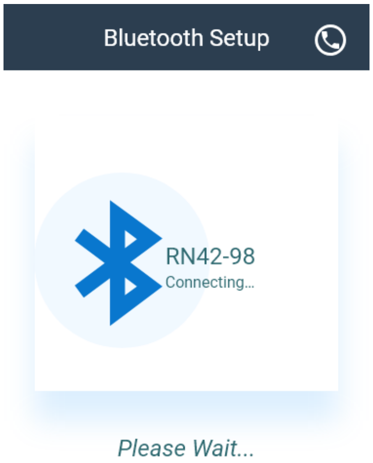
Bluetooth Setup

Please make sure WiSAT is powered on!

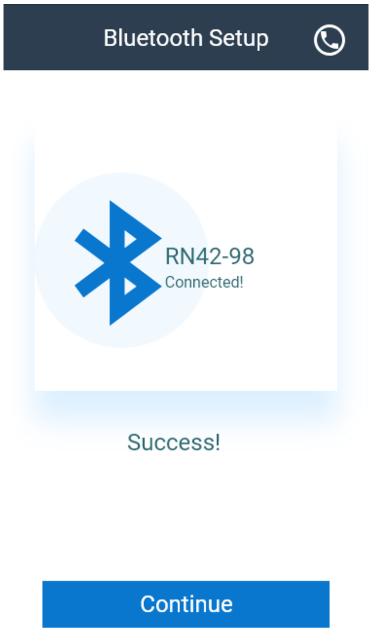
Also, ensure that Bluetooth is enabled on your phone.



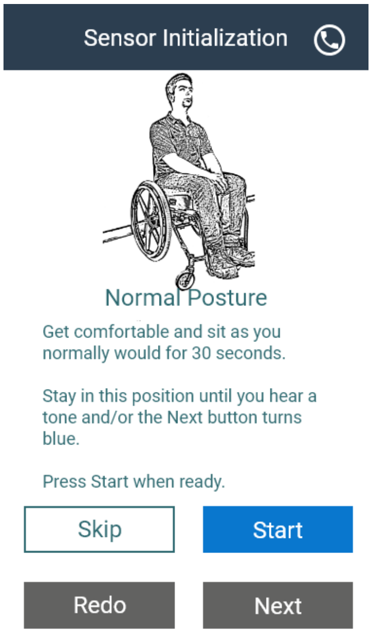
Select your module ID from the list below. The module ID is printed on the back of your WiSAT unit.



Please Wait...



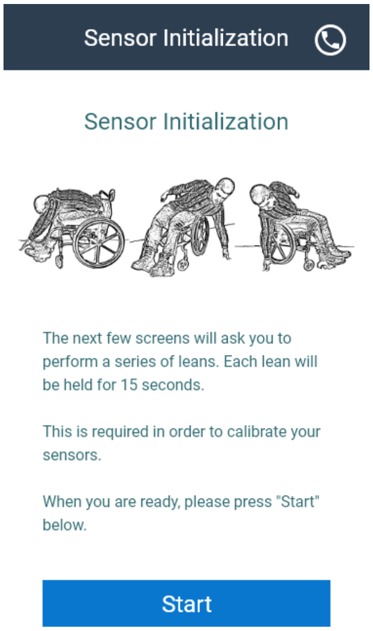
Success!



Get comfortable and sit as you normally would for 30 seconds.

Stay in this position until you hear a tone and/or the Next button turns blue.

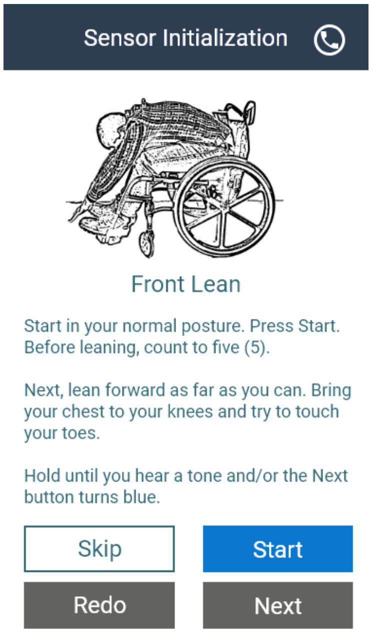
Press Start when ready.



The next few screens will ask you to perform a series of leans. Each lean will be held for 15 seconds.

This is required in order to calibrate your sensors.

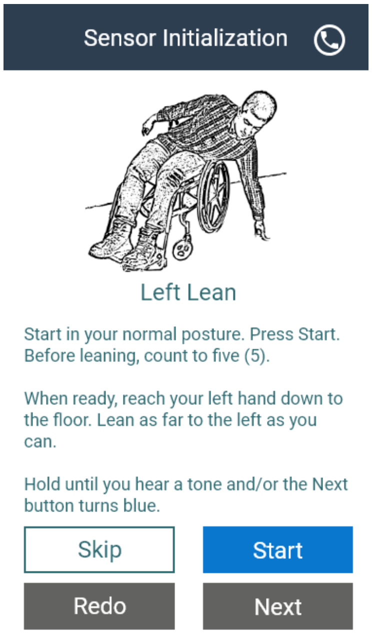
When you are ready, please press "Start" below.



Start in your normal posture. Press Start. Before leaning, count to five (5).

Next, lean forward as far as you can. Bring your chest to your knees and try to touch your toes.

Hold until you hear a tone and/or the Next button turns blue.



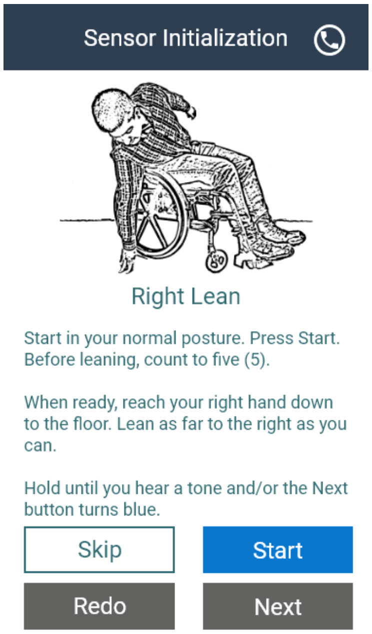
Sensor Initialization ☎

Left Lean

Start in your normal posture. Press Start. Before leaning, count to five (5).

When ready, reach your left hand down to the floor. Lean as far to the left as you can.

Hold until you hear a tone and/or the Next button turns blue.



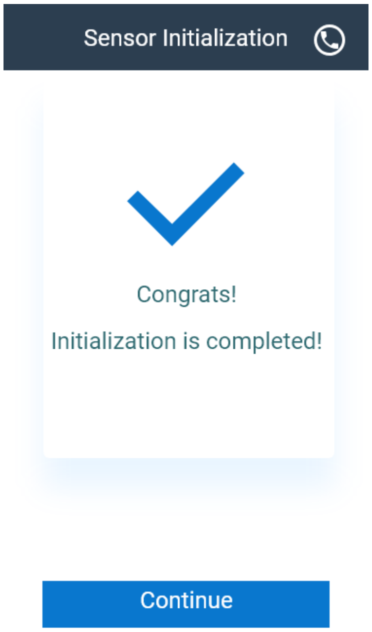
Sensor Initialization ☎

Right Lean

Start in your normal posture. Press Start. Before leaning, count to five (5).

When ready, reach your right hand down to the floor. Lean as far to the right as you can.

Hold until you hear a tone and/or the Next button turns blue.



Sensor Initialization ☎

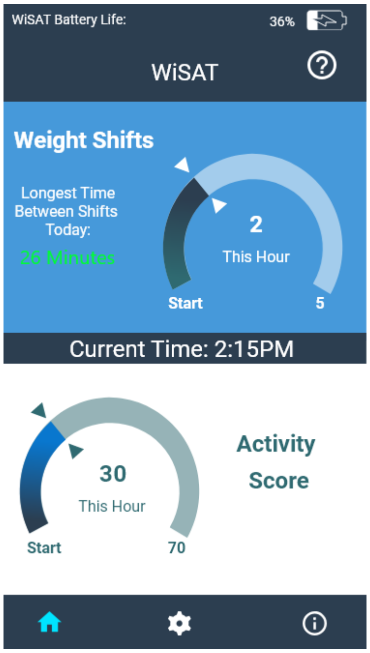
✓

Congrats!

Initialization is completed!

APPENDIX B: ALTERNATE WISAT DESIGNS

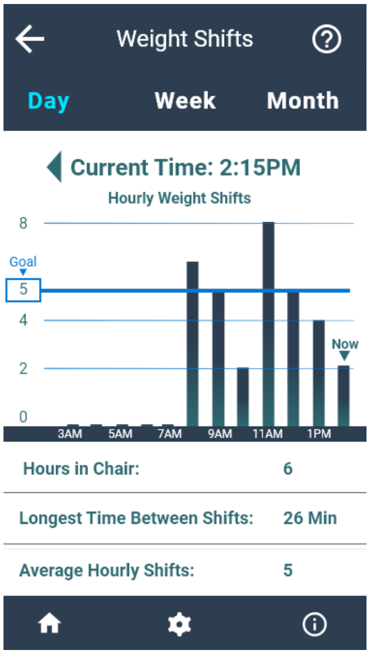
B.1: Design A



Longest Time Between Shifts Today: 26 Minutes

Current Time: 2:15PM

Home

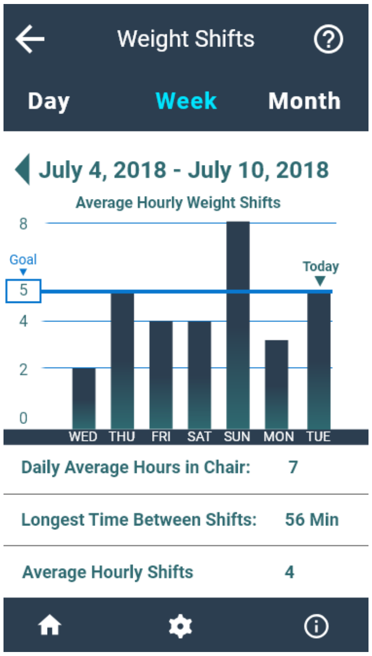


Hours in Chair: 6

Longest Time Between Shifts: 26 Min

Average Hourly Shifts: 5

Day

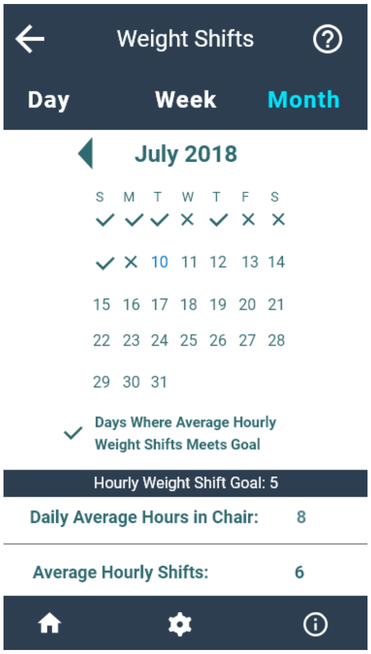


Daily Average Hours in Chair: 7

Longest Time Between Shifts: 56 Min

Average Hourly Shifts 4

Week



July 2018

S	M	T	W	T	F	S
✓	✓	✓	✗	✓	✗	✗
✓	✗	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

✓ Days Where Average Hourly Weight Shifts Meets Goal

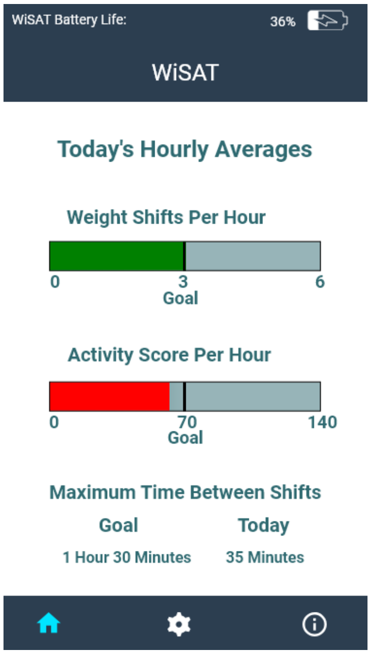
Hourly Weight Shift Goal: 5

Daily Average Hours in Chair: 8

Average Hourly Shifts: 6

Month

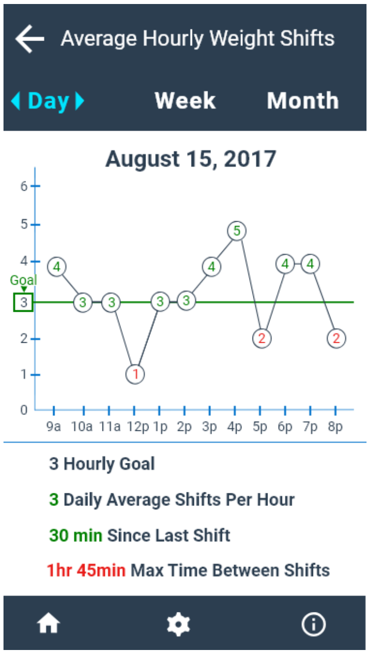
B.2: Design B



Maximum Time Between Shifts

Goal	Today
1 Hour 30 Minutes	35 Minutes

Home



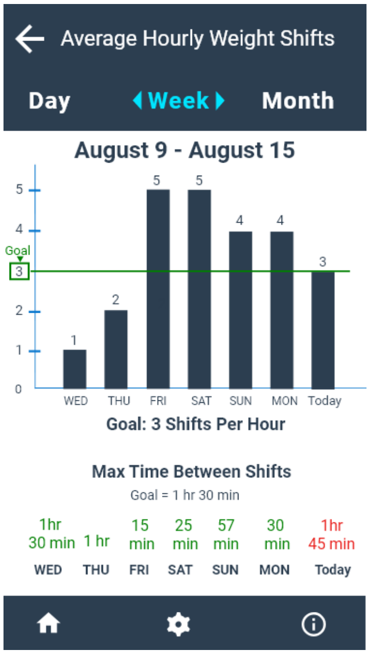
3 Hourly Goal

3 Daily Average Shifts Per Hour

30 min Since Last Shift

1hr 45min Max Time Between Shifts

Day

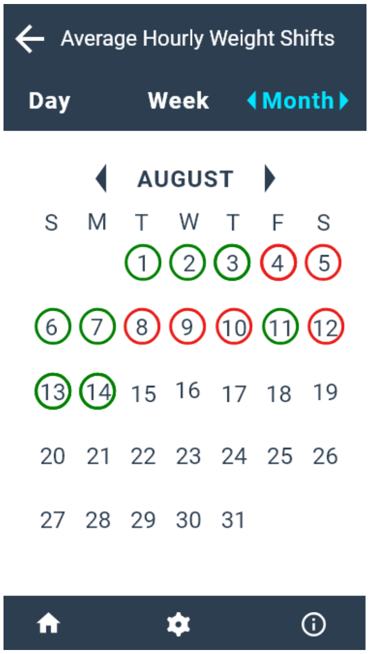


Max Time Between Shifts

Goal = 1 hr 30 min

WED	THU	FRI	SAT	SUN	MON	Today
1hr 30 min	1 hr	15 min	25 min	57 min	30 min	1hr 45 min

Week



S	M	T	W	T	F	S
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

Month

APPENDIX C: UI EVALUATION

C.1 Participant Engagement Plan

	Task	Approximate Duration (minutes)
1.	Introduction: The participant is provided a brief background on the mobile application and asked to sign a consent form if they agree to participate in the study. If they do not consent, the other steps in this list are not taken.	5
2.	The participant will be asked to perform a series of tasks representing three areas of app functionality: settings, information and activity goal status. The task series is listed in Appendix B. Two UI versions are being evaluated that differ in data presentation. The order of versions presented for evaluation will be randomized	45
3.	Survey: The participant will be asked to complete a survey (Appendix C) regarding their experience with the prototype app.	25
4.	Wrap-up: The participant will be asked if they have any further comments or questions and thanked for their time. A $30 gift card will be printed or emailed to the participant as a thank-you for their time.	5

C.2 Participant Engagement Plan

Randomly select whether Version A or Version B is presented first

Version A and B differ in Home Screen and Detail screens only

1. From the Home screen
    1. Go to the Information page. Go to the information about battery charging
    2. Read information and comment: was it clear? Do you understand how you would charge the battery?
    3. Return to the Home Screen
2. From the home screen:
    1. Go to the screen for weight shifts. In which hours did the user meet the weight shift goal?
    2. Navigate to a screen that shows weight shift statistics for the month. How many times did this user miss their goals in the past month?
    3. Return to the Home screen
3. From the Home screen
    1. Go to the In-Seat Movement (ISM) Score screen. Check the ISM Score status for the day. Comment on the ISM Score with respect to the goal
    2. Navigate to the Weekly information. In how many days was the ISM goal met for the week?
    3. Return to the home screen
4. From the home screen:
    1. Go to settings page. Go to the setting a goal page
    2. Set an ISM Score goal of 80 per hour
    3. Return to home screen

Repeat for 2nd version

1. From the Home screen
    1. Go to the Information page. Go to the information about Pressure Ulcers.
    2. Read information on screen 1 and comment. Was it clear? Do you understand the information on pressure ulcers? How could this be made clearer?
    3. Return to the Home Screen
2. From the home screen:
    1. Go to the screen for weight shifts. In which hours did the user meet the weight shift goal?
    2. Navigate to a screen that shows weight shift statistics for the month. How many times did this user miss their goals in the past month?
    3. Return to the Home screen
3. From the Home screen
    1. Go to the ISM Score screen. Check the ISM Score status for the day. Comment on the ISM Score with respect to the goal.
    2. Navigate to the Weekly information. In how many days was the ISM goal met for the week?
    3. Return to the home screen
4. From the home screen:
    1. Go to settings page. Go to the setting a goal page

2. Set a goal to perform 3 weight shifts per hour
3. Return to home screen

Side by side comparison: Home Screen A and Home Screen B

APPENDIX D: SUMI QUESTIONNAIRE – MODIFIED FOR WISAT

	Question	Answer Choices
Subscale A: Affect		
1	Working with this software is satisfying.	Agree, Undecided, Disagree
2	The way that system information is presented is clear and understandable	Agree, Undecided, Disagree
3	Working with this software is mentally stimulating	Agree, Undecided, Disagree
4	There is never enough information on the screen when it's needed	Agree, Undecided, Disagree
5	I feel in command of this software when I am using it.	Agree, Undecided, Disagree
6	I prefer to stick to the facilities that I know best.	Agree, Undecided, Disagree
Subscale B: Helpfulness		
1	I think this software is inconsistent	Agree, Undecided, Disagree
2	I would not like to use this software every day	Agree, Undecided, Disagree
3	I can understand and act on the information provided by this software.	Agree, Undecided, Disagree
4	This software is awkward when I want to do something which is not standard.	Agree, Undecided, Disagree
5	There is too much to read before you can use the software.	Agree, Undecided, Disagree
6	Tasks can be performed in a straightforward manner using this software	Agree, Undecided, Disagree
7	Using this software is frustrating	Agree, Undecided, Disagree
8	The software has helped me overcome any problems I have had in using it.	Agree, Undecided, Disagree
9	I keep having to go back to look at the guides.	Agree, Undecided, Disagree
Subscale C: Control		
1	It is obvious that user needs have been fully taken into consideration.	Agree, Undecided, Disagree
2	There have been times in using this software when I have felt quite tense.	Agree, Undecided, Disagree
3	The organization of the menus or information lists seems quite logical	Agree, Undecided, Disagree
4	The software allows the user to be economic of keystrokes.	Agree, Undecided, Disagree
5	There are too many steps required to get something to work	Agree, Undecided, Disagree

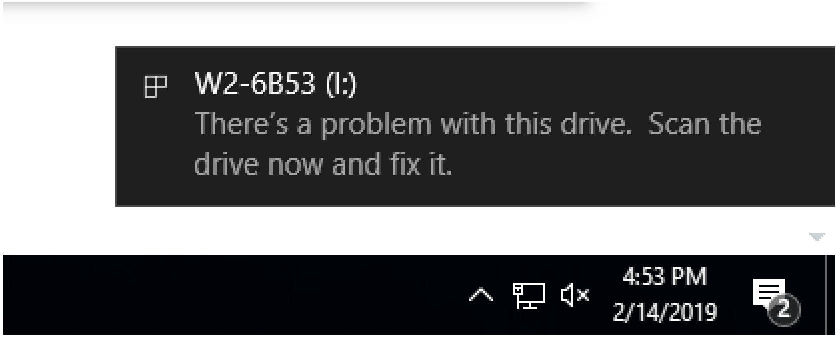
6	It is easy to make the software do exactly what you want	Agree, Undecided, Disagree
7	I will never learn to use all that is offered in this software.	Agree, Undecided, Disagree
Subscale D: Learnability		
1	The software has a very attractive presentation.	Agree, Undecided, Disagree
2	Either the amount or quality of the help information varies across the system.	Agree, Undecided, Disagree
3	It is relatively easy to move from one part of a task to another.	Agree, Undecided, Disagree
4	It is easy to forget how to do things with this software	Agree, Undecided, Disagree
5	This software is really very awkward.	Agree, Undecided, Disagree
6	It is easy to see at a glance what the options are at each stage	Agree, Undecided, Disagree
7	I have to look for assistance most times when I use this software	Agree, Undecided, Disagree
Addendum		
1	How important for you is the kind of software you have just been rating?	Extremely important Important Not very important Not important at all
2	How would you rate your software skills and knowledge?	Very experienced and technical I'm experienced but not technical I can cope with most software I find most software difficult to use
	What do you think is the best aspect of this software and why?	[Open-Ended]
	What do you think needs the most improvement and why?	[Open-Ended]

APPENDIX E: QUALITY ASSURANCE

E.1 Hardware Quality Assurance Protocol

To be repeated for each logger that is shipped to the REAR Lab. Values in red may need to be changed in the future.

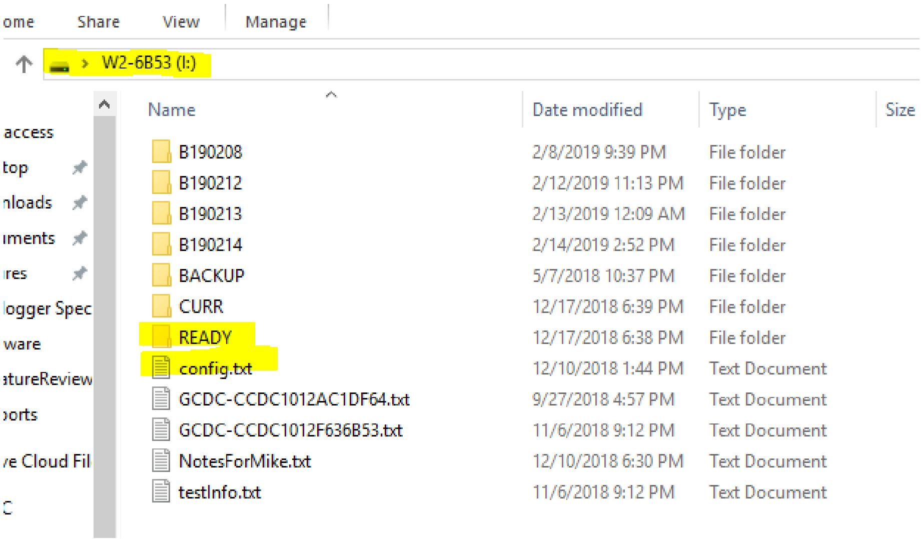
1. Physical Inspection
    - The logger should respond to the power button on the faceplate. Faceplate lights should blink as the power button is held.
    - Batteries: Two GMB652535 batteries should be wired in parallel
    - Faceplate Connector: Please replace faceplate connector if it's loose or broken free from the surface mounts
2. Resistor Inspection (Enclosure must be opened)
    - R79 should be 105kOhms.
3. Firmware Evaluation
    - Plug the datalogger into a computer. Wait for it to mount (i.e. for the computer's operating system to recognize that it has been attached). It should take less than one minute for this to happen. If this takes longer, contact the hardware vendor. On a Windows computer, you will see a message like this when the logger mounts:



W2-6B53 (I:)

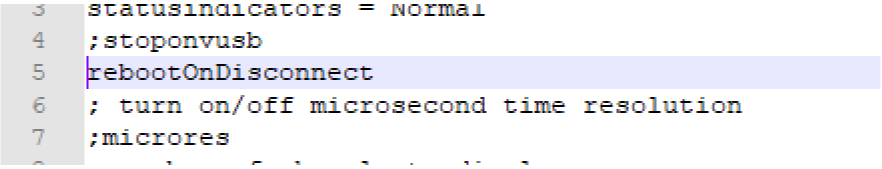
There's a problem with this drive. Scan the drive now and fix it.

    - The data logger behaves like a flash drive. On a Windows machine, it will map to a letter drive (I: , E: , etc). Open it as you would a flash drive.
    - Upon opening, you should see the following:



Name	Date modified	Type	Size
B190208	2/8/2019 9:39 PM	File folder	
B190212	2/12/2019 11:13 PM	File folder	
B190213	2/13/2019 12:09 AM	File folder	
B190214	2/14/2019 2:52 PM	File folder	
BACKUP	5/7/2018 10:37 PM	File folder	
CURR	12/17/2018 6:39 PM	File folder	
READY	12/17/2018 6:38 PM	File folder	
config.txt	12/10/2018 1:44 PM	Text Document	
GCDC-CCDC1012AC1DF64.txt	9/27/2018 4:57 PM	Text Document	
GCDC-CCDC1012F636B53.txt	11/6/2018 9:12 PM	Text Document	
NotesForMike.txt	12/10/2018 6:30 PM	Text Document	
testInfo.txt	11/6/2018 9:12 PM	Text Document	

- ○ Open config.txt with Notepad++.
  - ▪ Verify that the samplesperfile are set to 480 (2 minutes)
  - ▪ Verify that adc_sampleinterval = 250
  - ▪ Verify that stoponVusb and lowPowerOnVusb are present and uncommented. (Comments are semicolons (;) in the config file. If either are preceded by a semi-colon, they are commented out.)
  - ▪ Verify that rebootOnDisconnect is present and not commented out:



```
statusindicators = normal
;stoponvusb
rebootOnDisconnect
; turn on/off microsecond time resolution
;microres
```

  - ▪ If necessary, sample config files are available at R:\PressureReliefs\DOD WiSAT project\Data\Data Dumps. Select a config file from the last known working logger.
- ○ Open the READY directory and sort by Name. Verify that the most recent file (i.e. the one with the highest number) has not been reset to the year 2006 or 2007 nor has it been forwarded to a future date.
  - ▪ If the date looks off, you will need to reset it by pairing with the Realtime app
- ○ In the most recent file, check in the “Status” line for the firmware version. It should show the up-to-date version.
- ○ If that’s not the case, Either 1) Flash the new firmware (refer to Readme.txt in R:\PressureReliefs\DOD WiSAT project\Hardware\Gulf Coast Source Code) or 2) Contact GCDC
- ○ Safely eject the logger from your computer
- ○ Unplug the logger

4. Bluetooth Evaluation
   - ○ Attach a sensor mat to the logger
   - ○ Turn the logger on (if it is off) by pressing the power button on the faceplate.
   - ○ Give it a few seconds to “warm up”.

- Use a testing app or the REARLab's "realtime" WiSAT app to start streaming on the logger.
  - Note: the Bluetooth connection may waver while testing. Be sure to attempt multiple times if the connection is lost while streaming or sending batched data.
  - App instructions are separate from these steps. Source code and instructions live at: https://github.gatech.edu/RearLab/wisat
    - Note: If you get a 404 error, you will need to be added to the repository. Email rearlab@design.gatech.edu for this.
- Apply force to each of the mat's sensors while streaming and confirm that the app receives incoming data in a timely manner.
- Stop streaming
  - Note: this is down automatically in the realtime app by requesting a batch file
- Request a Batch File. Wait for it to be transmitted in its entirety.
  - This can take upwards of 1 minute to complete. The status light should blink more rapidly than normal while this transfer is taking place. If this rapid blinking stops before the file transfer is complete, or the transfer takes longer than one minute, it may be necessary to repair the logger and app and try again.
- After verifying the receipt of the batch file, let the logger continue to log data for 2 minutes. Afterwards, proceed to data validation

5. Data Validation
   - Using the sub-steps from Step 3, attach the logger to the computer and open the READY folder. Sort by Name.
   - The highest file number should be dated to the time you took data in step 4. As of Feb 2019, the app should have forced the logger to record data in the GMT+0 timezone.
   - Move to one folder above READY. Find the folder named BYYMMDD, where YYMMDD is the current year, month, and day.

Confirm that the batch file from Step 4 made it to this destination

E.2 Capacity Test Protocol

Filegen.bat generates fake WiSAT data to load test the datalogger. Edit it with Notepad++ or your favorite text editor. Assign the number of files you wish to generate to the "count" variable Use the filesizeInKB variable to assign a filesize (2 minute batch files = 38kb). To run:

1. Place the filegen batch file in the datalogger's READY folder.
2. Open a command prompt with admin privileges
3. Type E: (or whatever the datalogger's drive letter is)
4. If the command prompt does not show you in the READY folder, cd to the READY folder (i.e. "cd .\READY")
5. Run filegen.bat by typing "filegen.bat" in the command prompt and hitting enter.

Method (original):

1. Fill Logger to capacity using filegen.bat
2. Remove files gradually until logger becomes performant again. As it nears capacity, the logger may experience symptoms such as slow file save times and suspended Bluetooth communication. Once these symptoms cease, the logger is considered performant.
3. Verify that both Bluetooth streaming and writing to the SD Card function properly
4. Record total file count and total hard drive space used at which logger becomes functional

Method (updated):

If the logger is brought to a state where it experiences file system delays due to capacity issues, the file system will not necessarily correct itself after files are deleted. This means that the best way to test this issue may be to slowly increase the number of files present on the logger rather than max out the number of files on the logger and slowly decrease files.

1. Decide capacity benchmarks before running the test. (For example, 5,000 file, 10,000 files, 20,000 files, etc).
2. Start with the lowest number of files you wish to examine
3. Configure filegen.bat file to create the desired number of files
4. Run filegen.bat to completion
5. Verify that both Bluetooth streaming and writing to the SD Card function properly. Record the amount of delay that is experienced at the end of each batch.
    a. This can be measured by streaming data via Bluetooth in realtime and by comparing timestamps from the beginning and end of each file.
6. Continue testing with every desired benchmark

APPENDIX F: INITIAL SYSTEM REQUIREMENTS LIST

I. Data Flow

   A. Transmitted by Data Logger Mobile Application

      1. Hardware Status: Battery Life
      2. Initialization Success: Confirmation of the receipt of initialization data parameters
      3. Sensor Data/Time: A JSON object containing 6 integer values (sensor data) and a timestamp. One or two additional fields may be added if deemed necessary. The data logger collects data at a rate of 4 Hertz.

   B. Transmitted by Mobile Application to Data Logger

      1. Initialization Data Parameters: values that allow the data logger to perform an initialization calculation
      2. Bluetooth connection: Connection to data logger and mobile phone Bluetooth chips
      3. Request to Send Data: A request for #A3 above
      4. Confirmation of Receipt of Data: a confirmation of the receipt of #A3

   C. Transmitted by Mobile Application to Remote Server

      1. Errors: Errors and Exceptions that result in the loss of any data or interfere with the display of any UI elements shall be sent to a provided Georgia Tech RESTful API Endpoint when an internet connection is available. When offline, the application shall log errors to be sent once the application becomes online again.
      2. Data Transmission: Twice daily, the application shall transmit all unsynced data to a second, provided Georgia Tech RESTful API Endpoint if online. If offline at the scheduled sync time, the application shall do nothing.
      3. Manual Data Transmission: As a backup to the automatic data transmission, a button in the settings page shall send the Realm database to a Georgia Tech server when pressed.

   D. Overview: A data logger will record data from a pressure sensor mat regarding the user's sitting position at a rate of 1 to 4 Hertz. The data logger will save the sensor data to an internal database and transmit it, upon request, to the mobile application. The data transfer rate between the data logger and the mobile application is yet to be determined, but we expect it to be around once every minute or so.

II. User-Interface

   A. Subject ID Prompt Screen: Prompt the researcher to input an alphanumeric subject identifier and other clinical variables upon first use.
   B. Initialization: Prompt the user to provide initialization data on their first use of the system.
   C. Bluetooth Connection Pop-Over Window
   D. Settings: User-configurable settings screen
   E. Medical Information screens: static screens displaying medical information about pressure ulcers and their treatment

F. Notifications: Ongoing research will determine the best format(s) of the notifications to display to the user.
   1. Active Notifications: Display reminders to the user regardless of phone sleep status if the processed sensor data indicates noncompliance with pressure relief goals. We prefer these to take the form of offline-only notifications but are willing to utilize Push Notifications/Firebase-based notifications (in the case of Android devices) if they are determined to be the best option.
   2. Passive Notifications: UI screens containing visualizations detailing the user's past performance on the three metrics

III. Settings Options
   A. Goals
      1. Customizable pressure relief goals to fall within a predetermined range of values
   B. Alert Options
      1. Push Notification On/Off
   C. Clinical
      1. Weight
      2. User ID
      3. Cushion Type
   D. Initialization Setup
      1. Option to trigger an initialization to gather updated baseline data.

IV. Database: Local-only, Realm database

V. Error-Logging: Relevant errors/exceptions (as determined by REAR Lab in consultation with the vendor) shall be displayed to the user. All errors/exceptions shall be saved to the database.

VI. Clinical Metrics: The mobile application shall track three primary metrics
   A. Weight Shifts (algorithm to be provided)
   B. Time between movements (time since last weight shift)
   C. In-Seat Movement Score: a measure of the frequency of fidgeting that occurs throughout the day (algorithm to be provided)

VII. Usage monitoring:
   A. Log how frequently the application is opened.
   B. Log how frequently individual screens are opened.

VIII. Application Updates
   A. As a research tool, the REAR Lab will manage the distribution of APK and IPA files to the user base until the WiSAT system completes clinical trials. No commercial sale on the Google Play Store or Apple App Store is required at this time.

APPENDIX G: TEST CASES

Epic	Task	Happy Path Test Case	Edge Cases	Platforms
Database Schema	Create a database schema that stores raw data, initialization data, Weight Shift (raw count and time between), In-Seat Movement score (activity score), Device Information, User Information, and Error Logs	Export the database contents to a format that can be viewed outside the app. Verify that all tables and columns appear as specified in the schema with correct datatypes.	N/A	Realm Database
Datalogger Communication	Create an Application Program Interface (API) that allows for communication between the datalogger and app. The API should allow the app to initiate data transfer.	The API should function as documented (see tasks below for more details)	N/A	Datalogger
	The API should include a "streaming" command to transmit sensor data from the datalogger to the app in realtime.	A "start streaming" command sent from an app should result in sensor data being returned in realtime. A "stop streaming" command should stop this	Test sending commands out of order (stop before start)	Datalogger
	The API should include a command to request batched files from the datalogger	When this command is issued, the batch files should be streamed to the datalogger in order of oldest to newest.	Test Issuing this command multiple times before completing an individual file	Datalogger

	The datalogger should archive transmitted batches in its file system according to the date in which the batched data was taken.	After receiving a batch file, issue this command to let the data logger know that the file has been processed. The received file should appear in the datalogger's filesystem in a BACKUP directory named according to the date in which the data was taken.	Try issuing this command without receiving a file first. Try issuing it multiple times for the same file.	Datalogger
	Timestamps should be transmitted accurately in UNIX timestamp format	When requesting data in either batched or streamed formats, timestamps should be attached to each datapoint that accurately represent the UNIX time in which the data was taken.		Datalogger
	Mobile app should properly receive and store UNIX timestamp.	The mobile app should recognize the UNIX timestamp format and save this time internally. It should only convert to local time when displaying times to the user.		Mobile
	The datalogger should transmit battery life and hardware status upon request	Request a batched file. The file should contain this information in the header		Datalogger
	The datalogger should allow the mobile app to reset its timestamp	Sending this command should immediately update the time in the logger		Datalogger

	The mobile app should reset the datalogger timestamp hourly to avoid timing issues.	Run the app for a period of several hours. Afterwards, review the app's log files. There should be a record of an updated timestamp being sent to the logger in UNIX time every hour.		Mobile (both platforms)
	The mobile app shall start streaming mode before each of the desired initialization shifts and end it as each initialization shift is complete. It shall store the raw data internally and keep a record of the start and endtime of each shift.	Run through the initialization sequence in the mobile app. Afterwards, export the database. The raw data along with start/stop times for each of the initialization leans should be present in the database	Force the data logger to crash while attempting initialization. See if the app responds gracefully or not.	Mobile (both platforms)
	The mobile app shall request batched data during its normal operation (i.e. when not in initialization mode)	Run the app for a period of one hour with any screen open besides the initialization screens. Export the database. The database should contain raw data from batch files requested from the logger. Examine the app's logs to verify that batch requests (and receipt acknowledgements) were sent	Try this on multiple screens	Mobile (both platforms)
	The mobile app shall request batched data even when it is in the background or the phone is "asleep"	Repeat the above test case for 5 hours with the app in the background of the phone. Raw data should show that batched data was received while the app was in the		Mobile (both platforms)

		background and while the phone was asleep.		
	The mobile app shall contain a bluetooth connection screen as part of the first time setup	Progress through this screen as part of the first-time setup. With a logger running nearby the phone, attempt to connect to the logger. The logger should connect successfully, and the app should progress to a screen indicating a successful connection. Export the database. Ensure that the connection history was logged	Try connecting the app to the same logger multiple times. Ensure that the logger does not need to be power cycled to reconnect with the app.	Mobile and Datalogger
	The mobile app shall alert the user if the Bluetooth connection failed during setup	Repeat the above test case, but force the connection to fail by keeping the logger powered off. The app should display a screen indicating that the connection failed. Export the database and confirm that this was logged	Repeat this, but cause the app to be the source of failure. Ensure that the logger handles this gracefully.	Mobile and Datalogger
User-Interface	The mobile app shall contain a home screen that displays Weight Shift Count, Time Between Weight Shifts, and In-Seat Movement score	Export the database of an app that has already collected data on the three metrics. Confirm that the home screen reflects the data seen in the database	Try reinitializing and changing goals to see if this impacts the data displayed on the various screens.	Mobile (both platforms)

	The Weight Shift Screen shall show daily data	Confirm that the UI for the screen matches the prototype specifications and the exported database table		Mobile (both platforms)
	The Weight Shift Screen shall show weekly data	Confirm that the UI for the screen matches the prototype specifications and the exported database table		Mobile (both platforms)
	The Weight Shift Screen shall show monthly data	Confirm that the UI for the screen matches the prototype specifications and the exported database table		Mobile (both platforms)
	The In-Seat Movement Screen shall show daily data	Confirm that the UI for the screen matches the prototype specifications and the exported database table		Mobile (both platforms)
	The In-Seat Movement Screen shall show weekly data	Confirm that the UI for the screen matches the prototype specifications and the exported database table		Mobile (both platforms)
	The In-Seat Movement Screen shall show monthly data	Confirm that the UI for the screen matches the prototype specifications and the exported database table		Mobile (both platforms)
	The Contact screen shall display the contact information of appropriate research personnel	Confirm that the UI displays accurate information and matches the		Mobile (both platforms)

		prototype specification		
	The Information and Training section of the app shall appear as seen in the prototype specification	Confirm that the UI matches the prototype specification		Mobile (both platforms)
	The Settings screen shall allow the user to set goals, reinitialize, reconnect bluetooth, and activate notifications	Confirm that the UI matches the prototype specification		Mobile (both platforms)
	Goal Setting shall allow the user to set Weight Shifts per hour, Max Time between Shifts, and In-Seat Movement per hour.	Confirm that the UI matches the prototype specification. Change each goal at least twice. Export the database. Confirm that the database reflects your changes		Mobile (both platforms)
	The option to Reconnect bluetooth shall return you to the Connect Bluetooth dialog from the initialization/setup screens	Confirm that this is functional		Mobile (both platforms)
	The reinitialization option shall return the user to the initialization screens	Confirm that this is functional		Mobile (both platforms)
	The Activate Notifications feature shall prompt the user to enter a code to toggle the IsUsingActiveNotifications column	Enter the correct code. Export the database. The database should show that notifications are enabled	Attempt this with incorrect code	Mobile (both platforms)
	When the app is launched for the first time after installation, the first screen to appear shall prompt the user to input their setup information	Confirm that the UI matches the prototype specification.Export the database. Confirm that the database reflects your entered data.	Try entering invalid characters (i.e. strings in number fields, etc)	Mobile (both platforms)

	As part of setup, the user should be prompted to enter calibration data	Confirm that the UI matches the prototype specification.Export the database. Confirm that the database reflects your entered data.	Try entering invalid characters	Mobile (both platforms)
	Initialization screens shall appear at the end of the setup. These screens shall instruct the user to sit in their normal posture and to perform three consecutive weight shifts	Confirm that the UI matches the prototype specification.		Mobile (both platforms)
	Minimum screen size should be 1136x640 (iPhone 5) and 320dp (Android)	Load the UI onto a variety of screen sizes and confirm that the UI loads without scrolling	Play with accessibility settings on the phone. See if this breaks the UI.	Mobile (both platforms)
	Screens should progress as described through the Adobe XD prototype	Confirm that the UI matches the prototype specification	Rage click on various screens. See if you can force a crash.	Mobile (both platforms)
Remote Server Communication	Implement a Series of RESTful API Endpoints that allow the mobile app to POST data from the following tables: RepoParticipant, RepoGoal, RepoErrorLog, RepoDevice, ConnectionHistory, WeightShiftHistory, ActivityHistory, InSeatHistory, MonthArchive	For each endpoint, ensure that the request body allows us to POST data with each of the attributes of its respective table. A 200 OK response should be returned. The response body should contain the same data as the request body	Attempt sending data while ignoring nullable fields. Ensure that 200 OK is returned	Remote Server
	Mobile app should only attempt communication with remote server when connected to a WiFi network	Run the app with the WiFi feature turned off. No data should be sent during this time		Mobile (both platforms)
	Mobile app should send data at least twice a day when connected to a WiFi network	Run the app with a WiFi connection for a day. All endpoints should be updated at least twice		Mobile (both platforms)

Algorithm Communication	An Initialization Algorithm should take in a list of raw sensor data for the normal posture and three weight shifts and return the values in the ScalingFactors table	This algorithm is based on earlier work in Matlab. Run the algorithm in isolation from the rest of the app. Pass in initialization data to the algorithm. The algorithm should return the same scaling factor data as its counterpart in Matlab.	Set breakpoints inside the algorithm code and ensure that each sub-step aligns with the Matlab code	Java/Swift
	A classifier algorithm should take in raw data and scaling factors and return data for WeightShifts, In-Seat Movement, and In-Seat History	Same as initialization algorithm	Same as initialization algorithm	Java/Swift
	A calibration method should apply calibration to raw sensor data	Same as initialization algorithm	Same as initialization algorithm	Java/Swift
Error Logging	Lost Bluetooth Connections should be logged to the database	Drop the bluetooth connection intentionally. Export the database. Confirm that the dropped connection was logged		Mobile (both platforms)
	When the logger refuses to send batched or streamed data, this error should be logged	Simulate this with the datalogger. Export the database. Confirm that the dropped connection was logged		Mobile (both platforms)
	An error shall be logged when the WiSAT battery drops below 3600mv	Simulate this with the datalogger. Export the database. Confirm that the dropped connection was logged		Mobile (both platforms)

	Unhandled Exceptions should be logged	Simulate this with in the source code (perhaps with a "throw new Exception"). Export the database. Confirm that the dropped connection was logged		Mobile (both platforms)
	Algorithm Errors	Trigger an algorithm error manually (possibly by passing in data out of order). Export the database. Confirm that the dropped connection was logged		Mobile (both platforms)
App Lifecycle	Day 1 shall prompt the user for setup and initialization	On a fresh installation of the app, confirm that the setup and initialization screens appear as specified in the prototype specifications		Mobile (both platforms)
	Day 1 to 14 shall present the "Lock Mode" screen as specified in the prototype	After completing an initialization on a fresh installation of the app, the app should display the Lock Mode screen		Mobile (both platforms)
	Day 1 to 14 shall collect data in batched mode, as usual	After seeing the Lock Mode screen, leave the phone running within range of the datalogger for an hour. Export the database. Confirm that raw data from batched file taking during the hour exist in the database		Mobile (both platforms)

	Day 14 to Day 149 shall allow the user to interact with WiSAT in normal operating mode	Download a fresh installation of the app. On the setup screen, set the trial start date as 30 days in the past. The Lock Mode screen should no longer be visible. Instead, the home screen should appear		Mobile (both platforms)
	Day 149 and beyond should display an "end of study" screen	Repeat the test case for Days 14-149 but set the start date to 150 days in the past. The home screen should no longer appear, but the Completion Screen should appear as seen in the prototype specifications		Mobile (both platforms)
Notifications	Use the JITAI Framework to determine when Push Notifications should be issued to remind users of their goals	Run the app with for a day without making any progress towards goals. The app should receive push notifications	Try this while the app is offline to ensure that it is functioning	Mobile (both platforms)
	Alert user if the WiSAT Module and the smartphone app have been unable to reconnect after 5 or more attempts	Connect the app and the data logger. Disable the datalogger's bluetooth communication. The app should alert the user in a minute or two.		Mobile (both platforms)
	Alert the user if the WiSAT battery drops below 3600mv	Connect the app to a data logger with a low battery (~3700mv). Wait for the battery to drop to 3600mv. The app should alert the user		Mobile (both platforms)

	If an unhandled exception is encountered, the app should alert the user and provide them a code to give to the research team	Manually trigger an exception. Verify that the app has alerted the user with a code		Mobile (both platforms)
App Distribution	The app should be able to be installed on both iOS and Android platforms without any research team intervention	Install the app on a device through the chosen distribution platform	Try this for multiple devices	Mobile (both platforms)
Battery Evaluation	Find a battery with the greatest capacity that fits inside the datalogger case without overheating	Fully charge each potential battery candidate. Run a WiSAT logger with the battery inside until the logger drains completely. Use the batch file headers to create a battery discharge curve. Select the battery with the longest lifespan		Datalogger
Inductive Charger Evaluation	Find a 5W or lower inductive charger that can charge WiSAT overnight through Seat Cushions	Charge WiSAT with various inductive charger transmitter and receiver combinations. Record the final temperatures after charging for 30 minutes or more. Attempt to reduce temperature with spacer fabric and manipulating magnetic fasteners. Select the receiver/transmitter combination that allows for the fastest charge time without a dangerous temperature		Datalogger

Datalogger Capacity	The datalogger shall be able to store 6 months or more of WiSAT sensor data taken at 4Hz	Simulate 6 months of Sensor data on the logger. Monitor performance and response times. Ensure that the Bluetooth module still functions as expected, and that data is logged without any lost datapoints.		Datalogger

APPENDIX H: UI EVALUATION RESULTS

H.1 General, Open-Ended Comments from UI Evaluation Participants about WiSAT

Participants	Notes
Screen 01	• Battery icon might be confused with the phone's battery icon
Screen 02	• Wants to see in-seat movement/weight shift info filled out • Commented on Phone icon and asked if there were people he could call
Screen 03	• Goal setting: wants a number with a +/- or an up/down arrow • Says she thinks that folks with limited dexterity would appreciate this • Having the slider stop on tick marks would be a compromise, though
Screen 04	• Buttons are big and straightforward • Pretty straightforward overall • Got all the information • Wanted contact numbers/maintenance line
Screen 05	• Problem with set time: thinks having a rigid time schedule might mean that the notification may come too late for the user to do anything about it • Thinks the app is great • All information is in both, but he likes A better overall
Screen 06	• Asked about sliding functionality => part of the prototype, not the app • When he got to the wireless charging screen, he mentioned that he liked that idea for its convenience • "Pretty easy to navigate through"
Screen 07	• Graphs: "Reminds me of the Push Tracker app". This app apparently measures the manual vs coasting movement of wheelchairs. "Can't honestly say I've utilized it". It can track how much you are exercising your arms • Likes the visual of the colors and the ability to scroll back in time and see what goes where next. • Excited for WiSAT: asked when it would be available to the public • Likened it to fitbit • "Nagging mother app"
Screen 08	• "Easy to use" • Hard to find settings on his own apps, but "this one's real easy"
Screen 09	• Asked what re-initialization was • App Seems simple enough • "Not too difficult for any non-app user to try and figure it out"
Screen 10	• Referring to the app as a whole" "I think it's a good idea. Definitely something needed in the disability community." • Likes how interactive it is. Likes how things are clickable. Used max time between shifts as an example.

H.2 Summary of Open-Ended SUMI Results

Subject	**What do you think is the best aspect of this software and why?**	**What do you think needs the most improvement and why?**
1	Data displays – easy to read; quick to look at day, week, month	Clarify in-seat movement score – avoid arbitrary number. Maybe percentage of day. Make it easier to understand
2	I think the Activity points will be the best. It is something that you can use to measure how much work you are doing. You can gauge physical activity easily and understand the information.	Everything was good. Things that are unfinished need the most improvement
3	The breakdown in numbers. It is similar to the apple watch. You can see as much as you want or as little when it comes to your movement. For someone who really struggles with pressure sores, this is great!	Readability and wording. Add numbers to the graph and change the wording of the categories and it will be great!
4	Helps me make health decisions. Simple user interface allows for efficient interactions. Information modeled clearly.	Goal setting needs to be more free-form
5	The amount of information	Maybe an option that allows you to examine where you're shifting your weight and for how would be helpful and open your product to other applications such as sports
6	Letting the user know they have missed a weight shift	The wording on pressure ulcer vs pressure sore
7	Awareness	Modular information for other application or sub components. Better Posture
8	Easily Organized	Sometimes it took me a while to go through it because I'm not very functional at the end of the day
9	Software seems very user friendly for all skill levels and can give the information needed without confusing the user	*left blank*
10	*Can't read handwriting*	Charging it would be hard to get home and remember

H.3 SUMI Addendum Responses

How important for you is the kind of software you have just been rating? 1 (lowest) – 4 (highest)	How would you rate your software skills and knowledge? 1 (lowest) to 4 (higest)	What do you think is the best aspect of this software and why?	What do you think needs the most improvement and why?
2	2	Data displays – easy to read; quick to look at day, week, month	Clarify in-seat movement score – avoid arbitrary number .Maybe percentage of day. Make it easier to understand
2	3	I think the Activity points will be the best. It is something that you can use to measure how much work you are doing. You can gauge physical activity easily and understand the information.	Everything was good. Things that are unfinished need the most improvement
2	2	The breakdown in numbers. It is similar to the apple watch. You can see as much as you want or as little when it comes to your movement. For someone who really struggles with pressure sores, this is great!	Readability and wording. Add numbers to the graph and change the wording of the categories and it will be great!
2	1	Helps me make health decisions. Simple user interface allows for efficient interactions. Information modeled clearly.	Goal setting needs to be more free-form
3	2	The amount of information	Maybe an option that allows you to examine where you're shifting your weight and for how would be helpful and open your product to other applications such as sports

1	1	Letting the user know they have missed a weight shift	The wording on pressure ulcer vs pressure sore
1	3	Awareness	Modular information for other application or sub components. Better Posture
2	2	Easily Organized	Sometimes it took me a while to go through it because I'm not very functional at the end of the day
2	2	Software seems very user friendly for all skill levels and can give the information needed without confusing the user	*left blank*
1	2	*Can't read handwriting*	Charging it would be hard to get home and remember

H.4 Full Side-By-Side Responses

#	Home	Day	Week	Month
1	A: Reminds him of fitbit/apple watch. Wants these circular graphs with B's color scheme B: Prefers B's green/red color scheme. Better Graphs overall	B: Likes to be able to read off data points. Likes that there are numbers in the circles in the graphs B: Likes the color-coded data points	B: Likes having more data	A: Explains more
2	A: More modern, high tech B: likes straight bar graphs because it's easier to estimate where the halfway point is	B: "really easy" – likes numbers. He would rather not look on X and Y axis to figure out the time and score "Bar graphs are easier to read" "One is easier to read. One is more convenient"	B: Likes "days of the week at bottom and times and things" B: seems like more information	A: More information at bottom of same screen B: Green and red are simple to understand. Thought there might be an ability to scroll and get more data
3	A: "layout is better rather than a list" Likes how weight shift and time between are next to each other Similar to Apple Watch Grouped Together	Likes bar graph style of A but likes numbers in B Wants a combination where we have a bar graph with numbers Likes the way that screen A displays information ("title": "number") Likes how B also includes the Goal on screen	B: Likes bar graph with numbers Wants to have more summarized information (as in A) by default, but wants to see more information by clicking on a day Likes screen variants about the same Likes bar graph	Likes color in B Likes Xs in A's month but also likes that she can see the day number in B's screen Likes the breakdown of additional details in A but wants the option to remove these details and only see the month, if desired Color code checkmarks

		Likes the wording on screen A		
4	A: more color; more attractive to look at He liked the time being displayed in the middle of the screen Asked for the Phone Icon to be added to B B: Straight bar easier to read than circular, though	A: Prefers bar graphs Line straight across lets me pick out when goals are met Likes having the current time displayed B: Looks like an Excel graph He said it was a close decision between the two	A: more info on bottom	B: Pops out more and is color- coded Had to put a caption on A, whereas B is intuitive Likes the simplicity of both Likes Added information in A Concerned that folks who have brain injury might have a hard time interpreting the graph
5	A: liked aesthetics B: would like to see an exact number	Not sure which he liked better, at first A: Aesthetics better; function the same B: likes numbers	B: likes having more information (especially with regards to max time) “Definitely B”	A: worried about color-blind users
6	A: “seems more clear, more sharp” A: “Better in all ways”	A: Clearer; Can read chart right away B: Has to go from left to right and count with his finger	A: Seems more clear	A: Seems clearer
7	Changed his mind from A B: easier to see left and right Boldness makes it easier. Draws me into it more	Doesn’t like PM/AM B: Likes that the circle contains numbers; More information at a glance	B: Crisper, more direct Harder for him to process more information Would like B because of the breakdown of	B: primary colors pop Recognizes the red; has to read the X and the check A: “You lose the numbers”. He wants to be able to see the numbers.

	Vibrant, more engaging A: looks more muddled	“Brain resets” when having to shift focus: He compared this to how he keeps his file folders all in one line	max weight shifts per day: doesn’t have to think about it as much	
8	Easier to read A: “I like that size font” “It just seems like this screen is better organized”	“I prefer bar charts” “To me it’s easier to read bar charts”	“Better organized” “More detailed”	“I just prefer the checkmarks and X’s as opposed to the color-coding”
9	B is easier to read to any non-technical person because it’s a horizontal scale A gives more information – mentioned that it includes longest time between shifts He liked that the power was in the top right, but mentioned that most people will think of it as the phone’s power rather than the WiSAT’s power Likes the current time being displayed in A Likes the overall look of A but thinks that B is easier	Prefers line graphs versus bar graphs “Information is the same” Likes the actual number being displayed in the line graph	A: “I like this one a little bit better” A: “it gives you a little bit more information”. B: “Some of it more relevant than this one”	B: a little bit easier to read Would like B’s calendar on A’s screen because he would like the extra information of A with the calendar of B Likes the color coding and calendar information
10	A: looks more engaging He liked the central dividing bar in A because it’s easier to separate	A: easier to read “Less things going places” - referring to line graph “B confuses me for a minute”	“About the same to me” B: “I may like this one a little more” – referring to the max time between shifts at the bottom of	A: hourly shift goal B: Calendar is easier to track by dates Circled but also has date number

		"Times in A match up easier for me" – referring to matching X axis to the goals	the screen Liked max time in bottom of screen "A doesn't show goal"	A blocks it out and you can't figure out a specific date

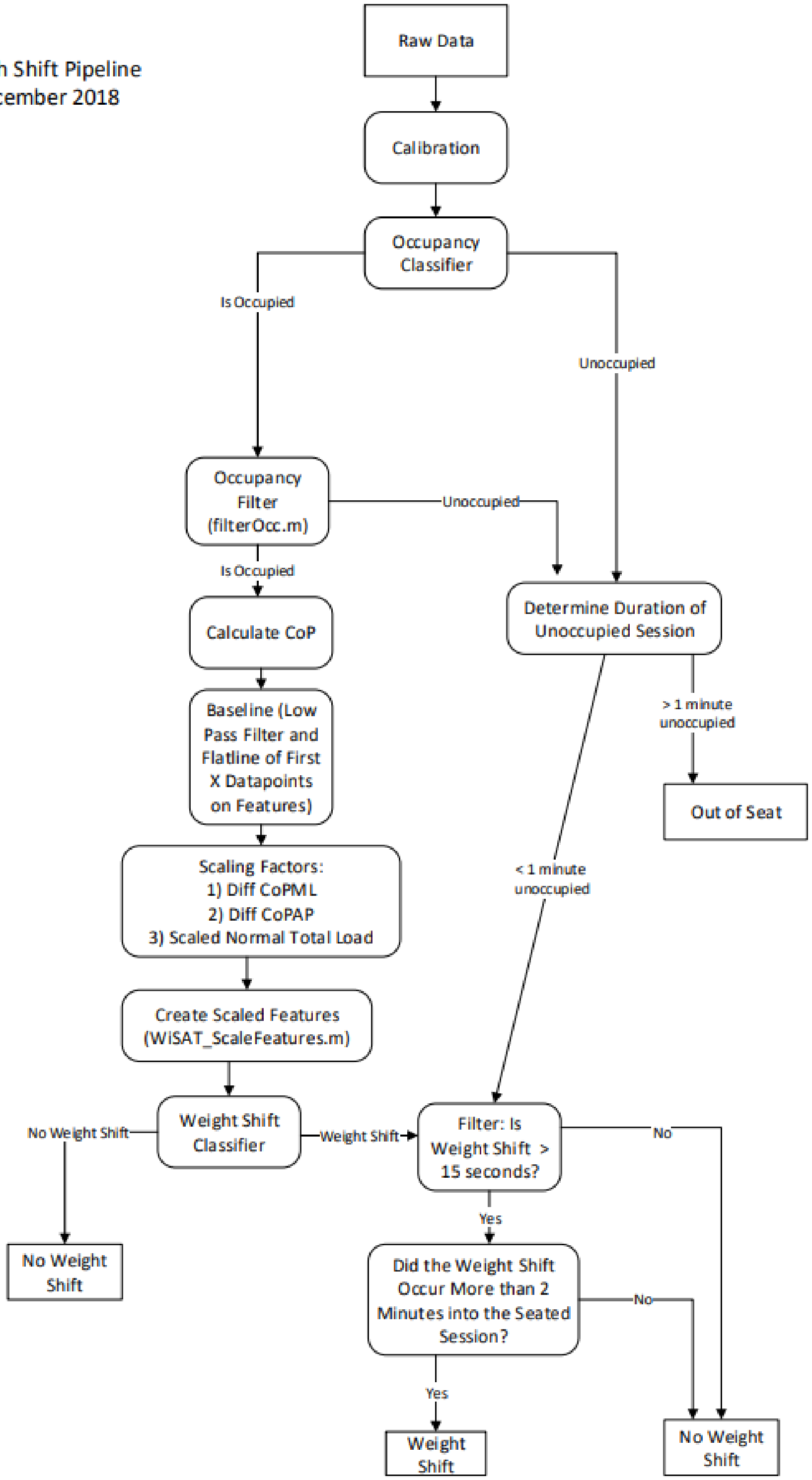
Appendix I: WiSAT’s JITAI Framework

Tailoring Variables	**Decision Point**	**Decision Rule**	**Intervention Options**	**Example Text***	**Alternate Text***
Weight Shift Count and **In-Seat Movement Score**	Every five hours of Occupancy	Compare average weight shifts per hour for the past five hours of occupancy to the Weight Shifts per Hour goal. Do the same for In-Seat Movement. • **If** both goals were met, **then** Scenario 1. • **If** weight shift was met, but not in-seat movement score, **then** Scenario 2 • **If** weight shift was not met, but in-seat movement score was, **then** Scenario 3 • **If** neither were met, **then** Scenario 4	Scenario 1	Congrats! You exceeded your WiSAT goals over the past 5 hours. X Weight Shifts/Hour Y Movement Score/Hour	Great job! You exceeded both your goals over the past 5 hours. X Weight Shifts/Hour Y Movement Score/Hour
			Scenario 2	*Try to move more often*. You averaged X hourly weight shifts in the past 5 hours, but only Y Movement Scores per hour.	
			Scenario 3	*Try to shift more often.* You averaged Y hourly Movement Scores in the past hours, but only X Weight Shifts per hour.	
			Scenario 4	Keep trying! Over the past 5 hours, you missed your shift and movement goals: X Weight Shifts/Hour	Try to shift and move more often. In the past 5 hours, you missed your goals: X Weight Shifts/Hour

				Y Movement Score/Hour	Y Movement Score/Hour

APPENDIX J: WEIGHT SHIFT CLASSIFIER PIPELINE (MATLAB)

Weigh Shift Pipeline
December 2018


Raw Data
Calibration
Occupancy Classifier
Is Occupied
Unoccupied
Occupancy Filter (filterOcc.m)
Unoccupied
Is Occupied
Calculate CoP
Determine Duration of Unoccupied Session
> 1 minute unoccupied
Out of Seat
Baseline (Low Pass Filter and Flatline of First X Datapoints on Features)
< 1 minute unoccupied
Scaling Factors:
1) Diff CoPML
2) Diff CoPAP
3) Scaled Normal Total Load
Create Scaled Features (WiSAT_ScaleFeatures.m)
No Weight Shift
Weight Shift Classifier
Weight Shift
Filter: Is Weight Shift > 15 seconds?
No
No Weight Shift
Yes
Did the Weight Shift Occur More than 2 Minutes into the Seated Session?
No
Yes
Weight Shift
No Weight Shift

REFERENCES

[1] EPUAP and NPUAP, "Prevention and treatment of pressure ulcers: quick reference guide," *International NPUAP/EPuap Pressure Ulcer Classification System. Perth, Australia: Cambridge Media,* 2014 2014.

[2] S. L. Hitzig, M. Tonack, K. A. Campbell, C. F. McGillivray, K. A. Boschen, K. Richards, and B. C. Craven, "Secondary health complications in an aging Canadian spinal cord, injury sample," (in English), *American Journal of Physical Medicine & Rehabilitation,* vol. 87, no. 7, pp. 545-555, 2008/07// 2008.

[3] E. A. Kruger, M. Pires, Y. Ngann, M. Sterling, and S. Rubayi, "Comprehensive management of pressure ulcers in spinal cord injury: Current concepts and future trends," *The Journal of Spinal Cord Medicine,* vol. 36, no. 6, pp. 572-585, 2013/11/01/ 2013.

[4] B. C. Chan, N. Nanwa, N. Mittmann, D. Bryant, P. C. Coyte, and P. E. Houghton, "The average cost of pressure ulcer management in a community dwelling spinal cord injury population," (in en), *International Wound Journal,* vol. 10, no. 4, pp. 431-440, 2013/08/01/ 2013.

[5] P. E. Houghton, K. Campbell, and P. Cpg, *Canadian best practice guidelines for the prevention and management of pressure ulcers in people with Spinal Cord Injury: a resource handbook for clinicians*. Ontario Neurotrauma Foundation Toronto, ON, 2013.

[6] M. D. Stillman, J. Barber, S. Burns, S. Williams, and J. M. Hoffman, "Complications of Spinal Cord Injury Over the First Year After Discharge From Inpatient Rehabilitation," (in English), *Archives of Physical Medicine and Rehabilitation,* vol. 98, no. 9, pp. 1800-1805, 2017/09/01/ 2017.

[7] G. DeJong, C.-H. J. Hsieh, P. Brown, R. J. Smout, S. D. Horn, P. Ballard, and T. Bouchard, "Factors Associated with Pressure Ulcer Risk in Spinal Cord Injury Rehabilitation," (in en-US), *American Journal of Physical Medicine & Rehabilitation,* vol. 93, no. 11, p. 971, 2014/11// 2014.

[8] S. E. Sonenblum, T. E. Vonk, T. W. Janssen, and S. H. Sprigle, "Effects of Wheelchair Cushions and Pressure Relief Maneuvers on Ischial Interface Pressure and Blood Flow in People With Spinal Cord Injury," (in English), *Archives of Physical Medicine and Rehabilitation,* vol. 95, no. 7, pp. 1350-1357, 2014/07/01/ 2014.

[9] S. E. Sonenblum and S. H. Sprigle, "Some people move it, move it… for pressure injury prevention," *The Journal of Spinal Cord Medicine,* vol. 41, no. 1, pp. 106-110, 2018/01/02/ 2018.

[10] S. Sprigle and S. Sonenblum, "Assessing evidence supporting redistribution of pressure for pressure ulcer prevention: A review," (in English), *Journal of Rehabilitation Research and Development; Washington,* vol. 48, no. 3, pp. 203-13, 2011 2011.

[11] S. E. Sonenblum, S. H. Sprigle, and J. S. Martin, "Everyday sitting behavior of full-time wheelchair users," (in English), *Journal of Rehabilitation Research and Development; Washington,* vol. 53, no. 5, pp. 585-597, 2016 2016.

[12] M. J. Coggrave and L. S. Rose, "A specialist seating assessment clinic: changing pressure relief practice," (in eng), *Spinal Cord,* vol. 41, no. 12, pp. 692-5, Dec 2003.

[13] National Pressure Ulcer Advisory Panel, "Prevention and treatment of pressure ulcers: quick reference guide," Washington, DC2009.

[14] D. A. Nawoczenski, "Pressure Sores: Prevention and Management," in *Spinal Cord Injury: Concepts and Management Approaches*, L. E. Buchanan and D. A. Nawoczenski, Eds. Baltimore: Williams &Wilkins, 1987.

[15] M. M. Sliwinski and E. Druin, "Intervention Principles and Position Change," in *Spinal Cord Injuries: Management and Rehabilitation*, S. A. Sisto, E. Druin, and M. M. Sliwinski, Eds. 1 Har/DVD edition ed.: Mosby, 2009.

[16] NPUAP/EPUAP/PPPIA, C. Media, Ed. *National Pressure Ulcer Advisory Panel/European Pressure Ulcer Advisory Panel/Pan Pacific Pressure Injury Alliance: Prevention and Treatment of Pressure Ulcers: Clinical Practice Guideline* (http://internationalguideline.com). Perth, Australia, 2014, pp. 1-308.

[17] J. S. Krause and L. Broderick, "Patterns of recurrent pressure ulcers after spinal cord injury: identification of risk and protective factors 5 or more years after onset," (in eng), *Arch Phys Med Rehabil,* vol. 85, no. 8, pp. 1257-64, Aug 2004.

[18] S. L. Garber, D. H. Rintala, K. A. Hart, and M. J. Fuhrer, "Pressure ulcer risk in spinal cord injury: predictors of ulcer status over 3 years," *Arch Phys Med Rehabil,* vol. 81, no. 4, pp. 465-71, Apr 2000.

[19] P. Raghavan, W. A. Raza, Y. S. Ahmed, and M. A. Chamberlain, "Prevalence of pressure sores in a community sample of spinal injury patients," (in eng), *Clin Rehabil,* vol. 17, no. 8, pp. 879-84, Dec 2003.

[20] S. E. Sonenblum and S. Sprigle, "You Got to Move It, Move It! Pressure Reliefs, Weight Shifts, and Wheelchair Mobility in Individuals with SCI," in *RESNA*, Arlington, VA, 2016.

[21] S. Carr and B. Wilson, "Promotion of Pressure-relief Exercising in a Spinal Injury Patient: A Multiple Baseline Across Settings Design," (in en), *Behavioural Psychotherapy,* vol. 11, no. 04, p. 329, 1983/10// 1983.

[22] M. Bächlin, K. Förster, and G. Tröster, "SwimMaster: A Wearable Assistant for Swimmer," 2009, pp. 215-224, New York, NY, USA: ACM.

[23] S. Consolvo, K. Everitt, I. Smith, and J. A. Landay, "Design Requirements for Technologies That Encourage Physical Activity," 2006, pp. 457-466, New York, NY, USA: ACM.

[24] S. Consolvo, P. Klasnja, D. W. McDonald, D. Avrahami, J. Froehlich, L. LeGrand, R. Libby, K. Mosher, and J. A. Landay, "Flowers or a Robot Army?: Encouraging Awareness & Activity with Personal, Mobile Displays," 2008, pp. 54-63, New York, NY, USA: ACM.

[25] A. Möller, L. Roalter, S. Diewald, J. Scherr, M. Kranz, N. Hammerla, P. Olivier, and T. Plötz, "GymSkill: A personal trainer for physical exercises," in *2012 IEEE International Conference on Pervasive Computing and Communications*, 2012, pp. 213-220.

[26] A. N. Thorndike, S. Mills, L. Sonnenberg, D. Palakshappa, T. Gao, C. T. Pau, and S. Regan, "Activity Monitor Intervention to Promote Physical Activity of Physicians-In-Training: Randomized Controlled Trial," (in en), *PLOS ONE,* vol. 9, no. 6, p. e100251, 2014/06/20/ 2014.

[27] L. Mamykina, E. Mynatt, P. Davidson, and D. Greenblatt, "MAHI: Investigation of Social Scaffolding for Reflective Thinking in Diabetes Management," 2008, pp. 477-486, New York, NY, USA: ACM.

[28] A. S. Miller, J. A. Cafazzo, and E. Seto, "A game plan: Gamification design principles in mHealth applications for chronic disease management," *Health Informatics Journal,* vol. 1, p. 10, 2014 2014.

[29] J. C. Kvedar, A. L. Fogel, E. Elenko, and D. Zohar, "Digital medicine's march on chronic disease," (in en), *Nature Biotechnology,* Comments and Opinion 2016/03/10/ 2016.

[30] T. Thadani, "Skip the pillbox — the answer to taking your medicine might be in your hand," in *San Francisco Chronicle*, ed, 2017.

[31] R. Dai, S. E. Sonenblum, and S. Sprigle, "A robust wheelchair pressure relief monitoring system," in *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 2012, pp. 6107-6110.

[32] D. Mravyan, M. Popovic, and M. Mravyan, "Monitoring system for pressure sore prevention," Patent US9149211B2, 2015-10-06, 2015. Available: https://patents.google.com/patent/US9149211B2/en.

[33] P. Ryan, "Integrated Theory of Health Behavior Change," *Clinical nurse specialist CNS,* vol. 23, no. 3, pp. 161-172, 2009 2009.

[34] K. Glanz and D. B. Bishop, "The role of behavioral science theory in development and implementation of public health interventions," *Annual review of public health,* vol. 31, pp. 399-418, 2010 2010.

[35] E. B. Kahn, L. T. Ramsey, R. C. Brownson, G. W. Heath, E. H. Howze, K. E. Powell, E. J. Stone, M. W. Rajab, and P. Corso, "The effectiveness of interventions to increase physical activity: A systematic review1,2 1The names and affiliations of the Task Force members are listed in the front of this supplement and at www.thecommunityguide.org. 2Address correspondence and reprint requests to: Peter A. Briss, MD, Community Guide Branch, Centers for Disease Control and Prevention, 4770 Buford Highway, MS-K73, Atlanta, GA 30341. E-mail: PBriss@cdc.gov," *American Journal of Preventive Medicine,* vol. 22, no. 4, Supplement 1, pp. 73-107, 2002/05/01/ 2002.

[36] J. O. Prochaska and W. F. Velicer, "The transtheoretical model of health behavior change," (in eng), *American journal of health promotion: AJHP,* vol. 12, no. 1, pp. 38-48, 1997/10//Sep- undefined 1997.

[37] G. W. Heath, D. C. Parra, O. L. Sarmiento, L. B. Andersen, N. Owen, S. Goenka, F. Montes, and R. C. Brownson, "Evidence-based intervention in physical activity: lessons from around the world," *The Lancet,* vol. 380, no. 9838, pp. 272-281, 2012/07/21/ 2012.

[38] Y. Wang, A. Fadhil, J.-P. Lange, and H. Reiterer, "Towards a Holistic Approach to Designing Theory-based Mobile Health Interventions," *arXiv preprint arXiv:1712.02548,* 2017.

[39] S. Michie, M. Richardson, M. Johnston, C. Abraham, J. Francis, W. Hardeman, M. P. Eccles, J. Cane, and C. E. Wood, "The Behavior Change Technique Taxonomy (v1) of 93 Hierarchically Clustered Techniques: Building an International Consensus for the Reporting of Behavior Change Interventions," (in en), *Annals of Behavioral Medicine,* vol. 46, no. 1, pp. 81-95, 2013/08/01/ 2013.

[40] H. Kuru, "Behavior Change Techniques Used in Mobile Applications Targeting Physical Activity: A Systematic Review," *Current and Emerging mHealth Technologies: Adoption, Implementation, and Use,* pp. 23-35, 2018.

[41] K. Mercer, M. Li, L. Giangregorio, C. Burns, and K. Grindrod, "Behavior change techniques present in wearable activity trackers: a critical analysis," *JMIR mHealth and uHealth,* vol. 4, no. 2, 2016.

[42] E. J. Lyons, Z. H. Lewis, B. G. Mayrsohn, and J. L. Rowland, "Behavior Change Techniques Implemented in Electronic Lifestyle Activity Monitors: A Systematic Content Analysis," *Journal of Medical Internet Research,* vol. 16, no. 8, 2014/08/15/ 2014.

[43] L. S. Belmon, A. Middelweerd, S. J. te Velde, and J. Brug, "Dutch young adults ratings of behavior change techniques applied in mobile phone apps to promote physical activity: a cross-sectional survey," *JMIR mHealth and uHealth,* vol. 3, no. 4, 2015.

[44] D. Simons, I. De Bourdeaudhuij, P. Clarys, K. De Cocker, C. Vandelanotte, and B. Deforche, "A smartphone app to promote an active lifestyle in lower-educated working

young adults: development, usability, acceptability, and feasibility study," *JMIR mHealth and uHealth,* vol. 6, no. 2, 2018.

[45] I. Nahum-Shani, S. N. Smith, B. J. Spring, L. M. Collins, K. Witkiewitz, A. Tewari, and S. A. Murphy, "Just-in-Time Adaptive Interventions (JITAIs) in Mobile Health: Key Components and Design Principles for Ongoing Health Behavior Support," *Annals of Behavioral Medicine,* vol. 52, no. 6, pp. 446-462, 2018.

[46] S. Dantzig, G. Geleijnse, and A. T. Halteren, "Toward a persuasive mobile application to reduce sedentary behavior," *Personal and ubiquitous computing,* vol. 17, no. 6, pp. 1237-1246, 2013.

[47] F. Spillers and S. Asimakopoulos, "Does Social User Experience Improve Motivation for Runners?," in *International Conference of Design, User Experience, and Usability*, 2014, pp. 358-369: Springer, Cham.

[48] D. Johnson, S. Deterding, K.-A. Kuhn, A. Staneva, S. Stoyanov, and L. Hides, "Gamification for health and wellbeing: A systematic review of the literature," *Internet Interventions,* vol. 6, pp. 89-106, 2016/11/01/ 2016.

[49] J. Nielsen, *Usability engineering*. Elsevier, 1994.

[50] M. F. Theofanos, "Common Industry Specification for Usabilty--Requirements," 2007.

[51] *ISO 9241-210:2010; Ergonomics of human-system interaction -- Part 210: Human-centred design for interactive systems*, 2010.

[52] J. H. Choi and H.-J. Lee, "Facets of simplicity for the smartphone interface: A structural model," *International Journal of Human-Computer Studies,* vol. 70, no. 2, pp. 129-142, 2012/02/01/ 2012.

[53] J. Maeda, "Simplicity," (in en), *BT Technology Journal,* vol. 22, no. 4, pp. 285-286, 2004/10/01/ 2004.

[54] L. Meloncon and E. Warner, "Data visualizations: A literature review and opportunities for technical and professional communication," in *2017 IEEE International Professional Communication Conference (ProComm)*, 2017, pp. 1-9.

[55] J. S. Ancker and D. Kaufman, "Rethinking Health Numeracy: A Multidisciplinary Literature Review," *Journal of the American Medical Informatics Association : JAMIA,* vol. 14, no. 6, pp. 713-721, 2007 2007.

[56] D. C. Mohr, S. M. Schueller, E. Montague, M. N. Burns, and P. Rashidi, "The Behavioral Intervention Technology Model: An Integrated Conceptual and Technological Framework for eHealth and mHealth Interventions," *Journal of Medical Internet Research,* vol. 16, no. 6, 2014/06/05/ 2014.

[57] F. Bentley, K. Tollmar, P. Stephenson, L. Levy, B. Jones, S. Robertson, E. Price, R. Catrambone, and J. Wilson, "Health Mashups: Presenting statistical patterns between wellbeing data and context in natural language to promote behavior change," *ACM Transactions on Computer-Human Interaction (TOCHI),* vol. 20, no. 5, p. 30, 2013 2013.

[58] I. Apple, "Physical and Motor Skills Accessibility," ed.

[59] Mobgen, "Apple WWDC," ed, 2016.

[60] M. Bruce and P. A. Pereira, *Microservices in Action*. Manning Publications Company, 2018.

[61] P. Cheng, "A smartphone application that informs weight shifting behavior to promote tissue health," Thesis, Georgia Institute of Technology, 2015.

[62] J. Kirakowski, "The use of questionnaire methods for usability assessment," *Unpublished manuscript. Recuperado el,* vol. 12, 1994.

[63] J. Kirakowski. *What is SUMI?* Available: http://sumi.uxp.ie/about/whatis.html

[64] G. O'Malley, G. Dowdall, A. Burls, I. J. Perry, and N. Curran, "Exploring the usability of a mobile app for adolescent obesity management," *JMIR mHealth and uHealth,* vol. 2, no. 2, 2014.

[65] S. van der Weegen, R. Verwey, H. J. Tange, M. D. Spreeuwenberg, and L. P. de Witte, "Usability testing of a monitoring and feedback tool to stimulate physical activity," *Patient preference and adherence,* vol. 8, p. 311, 2014.

[66] K. Beck, M. Beedle, A. Van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, and R. Jeffries, "Manifesto for agile software development," 2001.

[67] C. Larman and V. R. Basili, "Iterative and incremental developments. a brief history," *Computer,* vol. 36, no. 6, pp. 47-56, 2003.

[68] A. Cockburn, "Using both incremental and iterative development," *STSC CrossTalk (USAF Software Technology Support Center),* vol. 21, no. 5, pp. 27-30, 2008.

[69] K. Townsend, R. Davidson, and C. Cufí, *Getting Started with Bluetooth Low Energy*. O'Reilly, 2014.

[70] M. Todorov. (2016). *Realm is an Object-Centric Modern Database for Mobile Apps*. Available: https://academy.realm.io/posts/realm-object-centric-present-day-database-mobile-applications/

[71] A. O. S. P. . *Background Execution Limits*. Available: https://developer.android.com/about/versions/oreo/background