University of Alaska Anchorage

Using Android Motion Sensors for Carpal Tunnel Exercise Program

Author:
Stryker McLane

Supervisor:

Diana Steer, OT/L

Instructor:

Adriano Cavalcanti, PhD

Abstract:

The project described in this document is an Android app that is designed to aid the recovery of patients with Carpal Tunnel Syndrome (CTS). Carpal Tunnel Syndrome is a common among conditions that affect the hand and is caused by repetitive motions that occur in one's occupation. CTS causes pain and tingling in the hand, and patients of CTS suffer from a lack of sensation in the hand and wrist. CTS is often remedied through surgery, and the operation is followed by a lengthy recovery process. The recovery process requires individual care from an occupational therapist, but this therapy may be unavailable in rural areas. In order to bring specialized post-surgery care to patients recovering from CTS, I have developed the Patient Exercise Guide for Android.

The Patient Exercise Guide (PEG) contains several features that are vital to the recovery process. Patients must understand what they can and cannot do during the recovery process, so PEG will provide advice for the users on how to live their lives during recovery. Users will be addressed personally, as the app will track the status of recovery based on time. Additional information about CTS will also be provided. A key feature of PEG is how the user can perform exercises to increase the strength and range of motion in the wrist. These exercises will provide graphical feedback for the user and will detect the user's motions using the accelerometer in the Android device.

Acknowledgements:

I would like to thank my girlfriend Deandra for giving me the idea for the app, my partner Trent for working on the app with me, my supervisor Diana Steer for overseeing the project and Adriano Cavalcanti for instructing me in the capstone class.

Table of Contents:

Table of Contents

Figure List:

Chapter 1

Introduction

1.1 Introduction

Surgery for carpal tunnel syndrome is among the most common hand surgeries. Every year, more than 500,000 people in the United States undergo surgeries for carpal tunnel syndrome [1]. Carpal tunnel syndrome occurs when the median nerve becomes compressed (figure 1.1), and this compression is caused by repetitive motions such as working at a computer. The median nerve is responsible for sensations to the palm side of the thumb and fingers (although not the little finger), as well as impulses to some small muscles in the hand that allow the fingers and thumb to move. When the median nerve becomes compressed it can cause numbness, tingling, and loss of strength in the hand. Carpal tunnel syndrome can be treated surgically or non-surgically. Non-surgical treatments include drugs, exercise, and rest. Surgical options typically involve the median nerve becoming decompressed, and occupational therapy is required after. Following surgery, patients must regain range of motion in the wrist and fingers as well as strength and dexterity in their hand. Post-surgery treatment is commonly done in an outpatient clinic. Patients are given a home program that includes picture of exercises to do. However, some patients forget how to correctly perform the exercises, which can result in a slower recovery process. A slower recovery process can adversely affect the patient because it can lead to them not gaining full range of motion in the wrist, therefore impacting their daily life.

Since technology is growing in the medical field, health professionals are pursuing opportunities to use technology to ease prolonged therapies. Since mobile is cheap to develop for and distribute, many medical institutions are using mobile apps to connect with their patients. Mobile technology provides an outlet for medical information and doctors can relay specific information to a large audience through it. When the patient leaves the doctor's office, there is no guarantee that the patient will be focused on recovering. Even with follow-up appointments and home recovery programs, there is currently no way for a patient with carpal tunnel to receive daily advice and exercises that promote healing. To solve this problem, I developed the Patient Exercise Guide for Android.

The Patient Exercise Guide is an Android application that will be used by patients with carpal tunnel syndrome. Since carpal tunnel can affect a person's daily occupation, therapy is necessary to decrease symptoms. Losing sensation of the hand can lead to accidents in day-to-day living, such as cooking with fire or cleaning with chemicals. The Patient Exercise Guide will compliment a traditional home

program that the patient receives after surgery. It will guide the patient through recovery by assigning exercises that gradually extend the range of motion in the patient's wrist and also provide detailed instructions on the patient's condition. The exercises will have the user rotate their wrist with the Android device attached to the hand. Motion sensors will detect these rotations to test their range of motion. As recovery progresses, users will be given visual feedback on how much they have increased range of motion so they can restore wrist flexibility to normal levels.

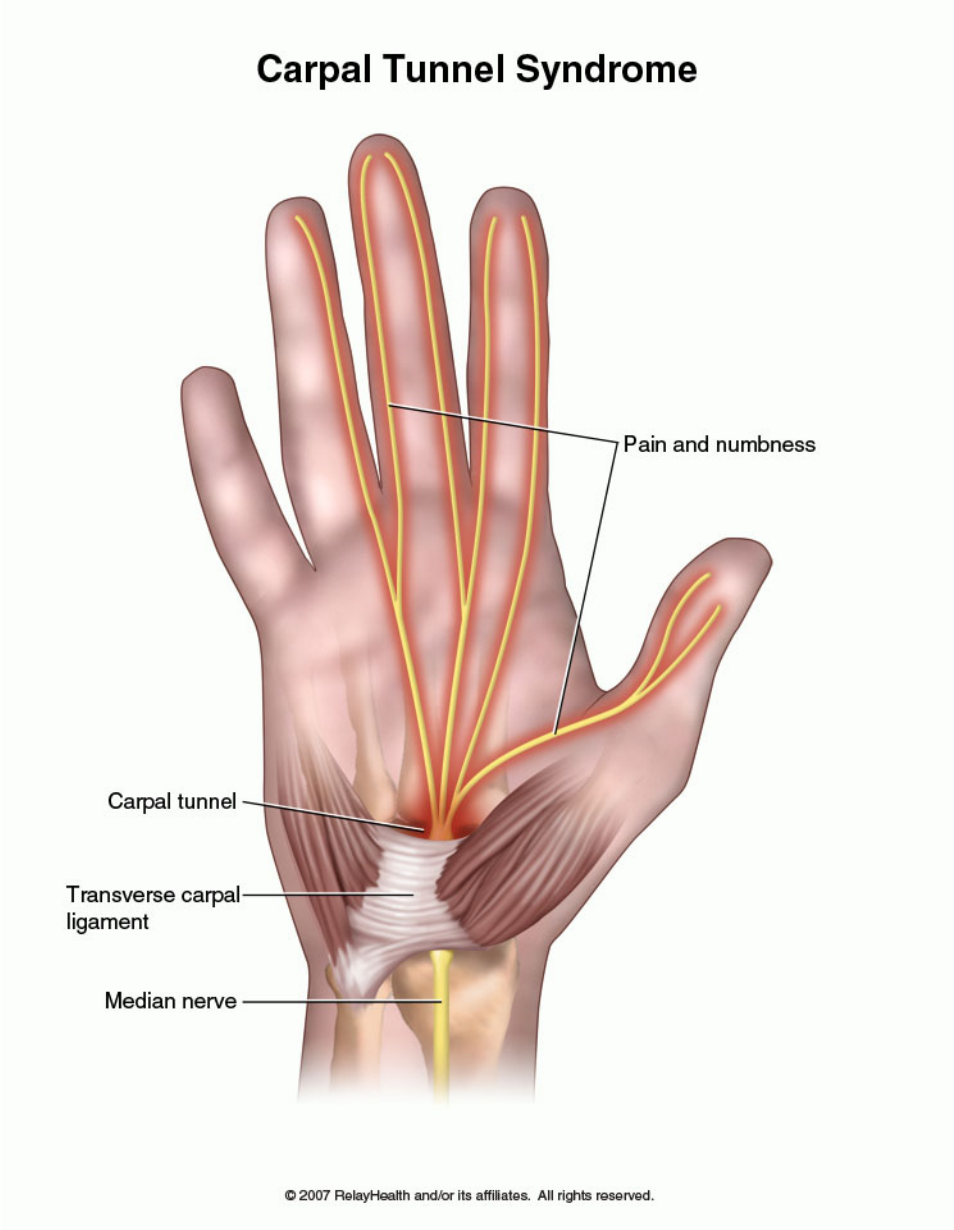


Figure 1.1 - Diagram of median nerve and areas affected by carpal tunnel syndrome from Summit Medical Group. [6]

1.2 Application

The first release of the Patient Exercise Guide will be designed for individuals recovering from carpal tunnel syndrome and will feature information and exercises that will guide the user through the recovery process. Later versions will feature exercises and information that cover a variety of conditions. Users will be instructed to use the Patient Exercise Guide after surgery. Once started, the app will begin to record the recovery process over several weeks. During week one, the Patient Exercise Guide will inform the user of what they should do in the early stages of recovery. Throughout the day, the user will be reminded to do exercises that strengthen the fingers and wrist. Introductory exercises will be executed by repeating motions shown on the screen of the Android device. As recovery progresses and the muscles begin to strengthen, the user will be assigned to exercises that use the motion-sensing capabilities of the phone.

Figure 1.2 – Logo for Android.

Google's Android platform allows for versatile app development, and developers have many options for health-based apps by using the features of handheld and wearable android devices. The Patient Exercise Guide will use motion-detecting hardware in the Android device to help the user complete motion-based tasks. Android devices with firmware version 2.3 or above have access to the gyroscope sensor type, which can detect radians per second on the x, y and z axes. When the user holds his or her phone and rotates their wrist, the axis of rotation detected by the device will align with the axis of rotation in the wrist. The app will test the user's pronation and supination of the wrist, as well as flexion/extension and radial/ulnar deviation (Figure 1.2). This allows the app to quantify the patient's rotation of the wrist and provide measurements of the patient's range of motion. The app will track how the user performs each exercise, whether they are able to complete it or not. In future versions, this data will be stored and displayed in a graph over time so that the user has a visual representation of his or her recovery. By showing daily trends through graphical data, the Patient Exercise Guide keeps up with other health-based apps that show the user's activity over time.

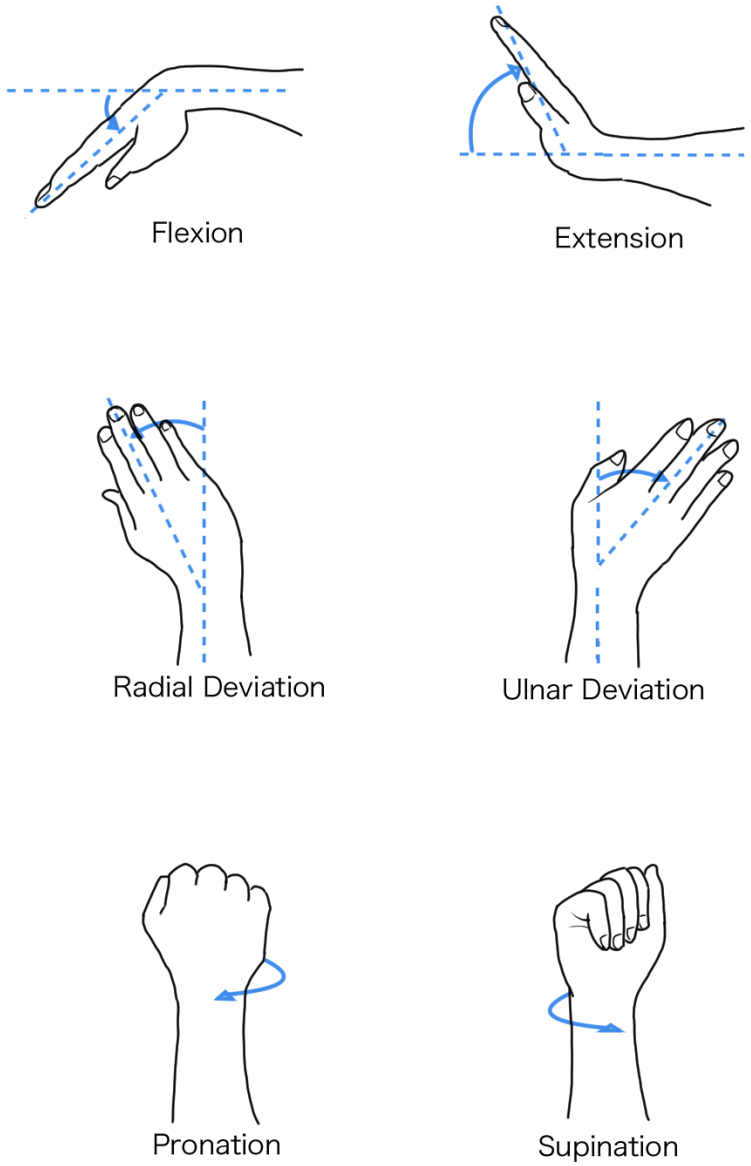


Figure 1.3 – Depiction of wrist movements that the Patient Exercise guide will assign for exercises.

In occupational therapy, it's the patient's responsibility to inform themselves of their condition to maximize recovery. The Patient Exercise Guide will serve as a valuable resource of information for patients with carpal tunnel with comprehensive knowledge of the syndrome. After surgery, extra care must be taken by the patient to ensure the healing of post-surgery wounds. In the early stages of recovery, the app will instruct the patient on regular cleaning of the incision wound and warn them of the activities that will detriment recovery. Informing the patient on how to live his or her life while

physically impaired is fundamental to occupational therapy, and this app will provide plentiful instructions on basic living without heavy use of one hand. The Patient Exercise Guide will instruct the user on how to eat, bathe and drive by using mostly one hand. It will also remind them to keep their splint and dressing clean and dry, and show the proper methods for doing so. Basic tips such as medicine intake and proper icing and elevation will also be present. By connecting to the user's calendar app and notifications, The Patient Exercise Guide will automatically schedule follow-up appointments in the user's agenda. Besides serving as a manual for post-surgery care, the app will include comprehensive knowledge of other conditions treated through occupational therapy.

1.3 Motivation

The vast audience of mobile applications combined with the need for occupational therapy for all demographics give way to an extremely wide potential user base for the Patient Exercise Guide. A 2015 survey by Pew Research Center revealed that two thirds of American adults own a smartphone [2] (figure 1.4). Health and fitness apps are some of the most common uses for smartphones, and many manufacturers today often have health apps preinstalled in the phones they offer. Additionally, health and fitness have influenced the hardware in smartphones. For example, the Samsung Galaxy S6 is equipped with a heart rate monitor and pedometer and comes with the app that uses these features preinstalled.

This hardware has helped the prevalence of wearable technology that is dedicated for fitness management, such as Fitbit. Mobile technology has opened a gateway to health and fitness for smartphone users. Over the first half of year 2014, data collected by the mobile analytics company Flurry indicated that the use of health and fitness apps increased by 62% [3]. This study examined 6,800 mobile apps on Apple's iPhone and iPad mobile devices. Considering these statistics and the fact that Android devices boast many of the same features as Apple's mobile devices, the time is right to release a health-based app for mobile.

The origins of occupational therapy trace back to the early 20th century, but knowledge of its practices is not widespread to many people. Rural areas, especially in Alaska, do not have occupational therapists available. The Patient Exercise guide will be designed to replicate sessions of occupational therapy and provide rural areas with therapy services. Simulating occupational therapy through a mobile app will be accomplished by using professional language and a unique interface. Information provided by the Patient Exercise Guide will be verified by actual occupational therapists and exercises will be based on professional practices. Since occupational therapy deals with all age groups, it is important to consider this in the user experience. Menus will be simple and concise to ease navigation for both young and elderly users, and the interface of the exercise screen will show clear instructions so that people who are not technologically inclined can understand them. Occupational therapists serve patients with physical, mental and cognitive disorders, so the app will be designed to be accessible by individuals with disabilities. Addressing such a large and varied user base proposes a unique challenge in creating a user interface that is intuitive, unique, and memorable.

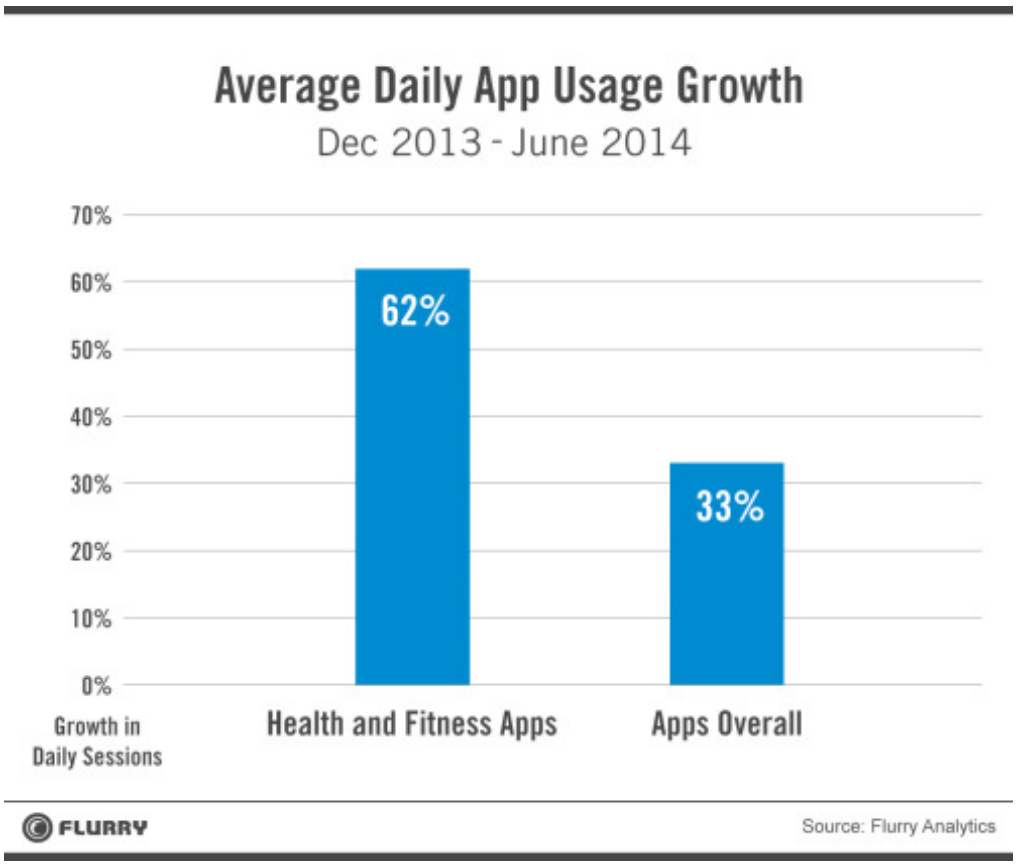


Figure 1.4 – Data collected by Flurry showing the growth of health and fitness mobile apps.

1.4 Recent Developments

Since the advent of smartphones, many developers have shifted their efforts toward mobile development and the wide audience of smartphone users permits applications to be designed for all possible situations and cater to all skill levels. People of all ages use mobile applications, be it for recreational activities or practical utilities. Over the past decade, Medical institutions have been using mobile technology to bring better healthcare to their patients. In 2015, the HIMSS Mobile Technology Survey revealed that 90% of participating healthcare professionals are using mobile technology to engage patients and 73% of them are using health-based apps designed for the patient [4]. Besides mobile apps for the patient, other features of smartphones are being used for providing healthcare such as remote monitoring and text messaging between doctor and patient. These factors allow patients to be more involved in their own healthcare since the interface is tied to the user. Mobile technology is emerging as a powerful tool for healthcare providers, and occupational therapy benefits from this.

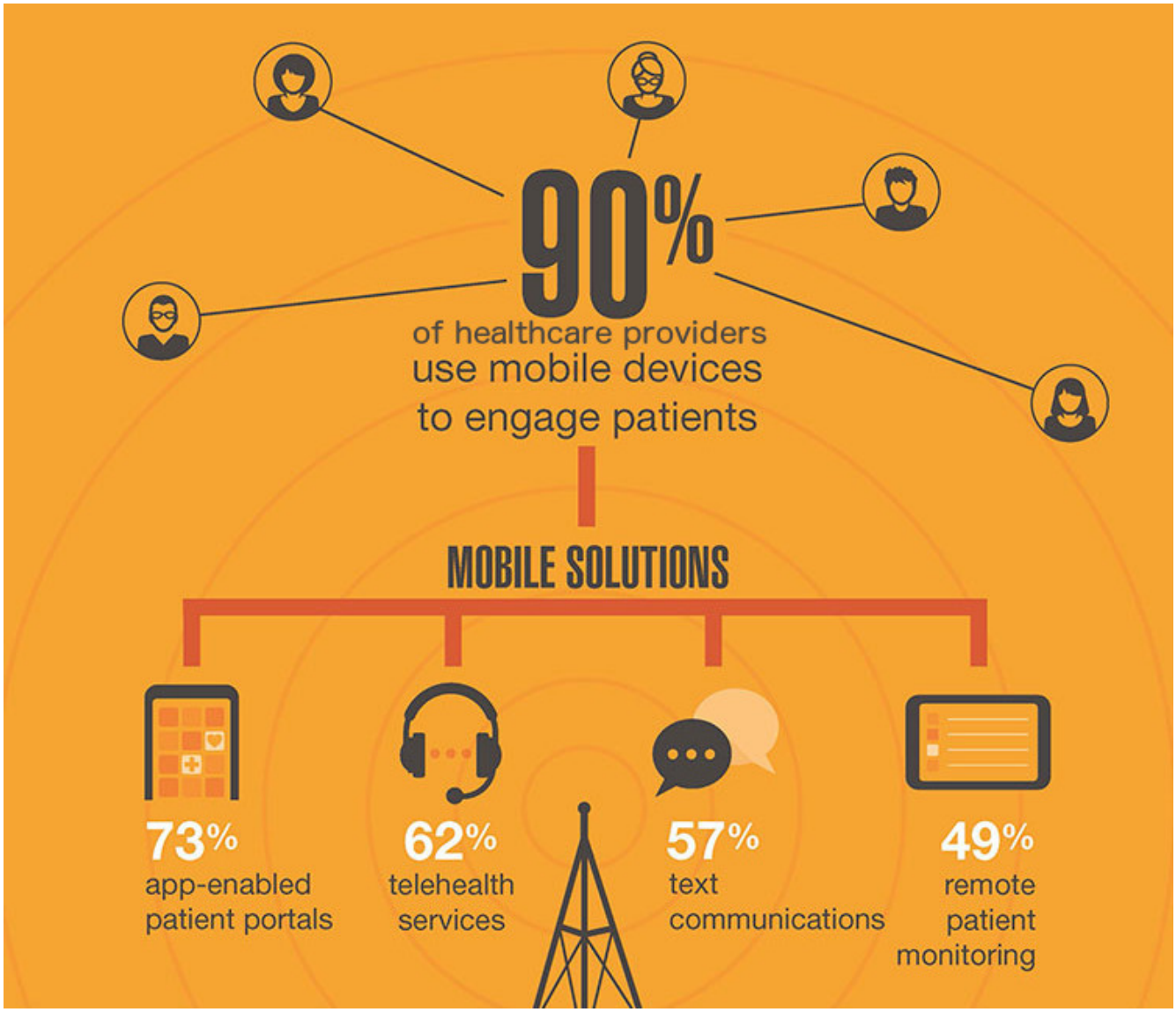


Figure 1.5 – Results of the HIMSS survey shows the effectiveness of mobile technology in healthcare.

Currently, several apps for Apple and Android devices are designed for patients of occupational therapy. Many of them are for pediatric care and employ simple exercises that test fine motor skills and dexterity in children. Since young children respond well to tablets, implementing simple cognitive exercises as mobile apps is very effective to bring therapy to children. The use of digital applications in a rehabilitative setting has been proven to be very conducive to patient recovery, as Occupational therapy practitioners have long standing expertise in providing services to clients that incorporate technology [5]. The landscape of occupational therapy could change for the better by including smartphone technology in their practice.

Chapter 2

System Integration and Modeling / Methodology

2.1 Introduction

When I started developing the initial design draft for the patient exercise guide, I had anticipated developing the app by myself. A few weeks into the semester, however, Trent Matthias joined the project. Although Trent had several options for his capstone project, he formed a team with me so that he can develop for a platform that he has more experience in. Trent's background in Android development has given a fresh perspective to the project. His advice has allowed me to optimize the program and avoid some amateur mistakes that many new developers fall prey to. Trent and I have worked together on projects in several other classes, and we both have experience in software development using the agile methodology. Together we have delegated tasks to be completed throughout the semester, and by practicing agile development we can stay organized and deliver working software consistently. With Trent's help, efforts to complete the Patient Exercise Guide have doubled.

2.2 Technology

I had only been developing for Android a few months before the project started, but the wealth of resources available for developers made it easy to implement the ideas for my design. Since the app has such a wide potential audience, it is imperative to create an interface that is easy for the user to digest. Android uses extensible markup language (XML) for its interfaces and I had an easy time adapting to this format despite my lack of xml experience. Before I started college, I had worked with HTML to create web page layouts. My background in HTML gave me a head start in understanding XML concepts such as tags and nested elements. It has also introduced me to the importance of visual elements such as colors and fonts. Since the Patient Exercise will serve as a learning tool for post-surgical care, it is important to portray information in a clear and concise fashion with legible text and non-clashing colors. Throughout development, I have been tweaking the interface design so that the look and feel of the Patient Exercise Guide can stand up to other popular health and fitness apps in terms of usability and accessibility.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto" android:layout_width="match_parent"
    android:layout_height="match_parent" android:paddingLeft="16dp" android:orientation="horizontal"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    android:paddingBottom="16dp"
    app:layout_behavior="android.support.design.widget.AppBarLayout$ScrollingVie..."
    tools:showIn="@layout/activity_landing_page" tools:context=".LandingPage">

    <TextView android:text="Welcome to PEG!" android:layout_width="wrap_content"
        android:layout_weight="1"
        android:layout_height="wrap_content" />

    <EditText android:id="@+id/edit_message"
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:hint="Enter your name here!" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="sendMessage"
        android:text="Click me when done!" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="goToTilt"
        android:text="@string/button_tilt" />

</LinearLayout>
```

Figure 2.1 – A sample of an XML layout file for Android.

As a platform, Android is both versatile and accessible. There are many options for the developer to make an effective user interface. Catering our interface to the user is a vital aspect of agile development. According to Steven Kreug, an interface should be designed based on how the user approaches it. The tasks the user undertakes should be simple, quick and memorable [7]. By simplifying our layout and anticipating all possible user input, we can create an effective interface that will be memorable for the users.

2.3 Components

Developing and testing the Patient Exercise Guide requires only a development environment and a device to test on. The device in which tests are being run is my personal cellular phone, a Samsung Galaxy S6. Since this device was released recently, it has some new hardware features that could benefits the Patient Exercise Guide. In addition to the magnetometer and accelerometer, the Galaxy S6 has a pedometer and vitality sensor. This hardware could be implemented in the Patient Exercise Guide for it to serve as a full-featured health app. Although Android Studio allows emulation of an Android Device, I opted to base development and testing on a physical device. Since much of the Patient Exercise Guide uses motion detection, capturing the imperfect gestures of human motion input would not be possible on a virtual device.

Figure 2.2 - The hand mount allows users to exercise their wrist without gripping the Android device.

Since the Patient Exercise Guide is aimed at users with impaired hands, we have considered that our typical user may be restricted from gripping their Android device. To counter this potential issue, we have designed a mount for the Android device that straps on to the user's hand. This mount allows the user to perform the exercises without having to grip the device. Carpal Tunnel Syndrome causes loss of sensation in the fingers, so this mount will increase both the functionality of the app and the effectiveness of the exercises. The prototype consists of a fitness armband for the Galaxy S6 device that is used for testing. The armband has been modified so that the device sits in landscape orientation on the user's hand. The hand mount will complement the Patient Exercise Guide by allowing the device to mimic wearable technology. While wearable technology is extremely effective in fitness, its use in health is emerging as a powerful rehabilitative tool. According to Paolo Bonato of the Journal of NeuroEngineering and Rehabilitation, "Wearable technology allows clinicians to gather data where it matters the most to answer this question, i.e. the home and community settings [8]."

2.4 Project Developments

Work on the project is progressing steadily in the first few weeks of development. With Trent's help, I am able to expand my knowledge of the Android platform at a quicker pace. With another person on the project, I can delegate smaller tasks and reach decisions more efficiently to create a better product. We mostly work on the same tasks simultaneously, but we have also defined our individual roles in the project. Trent has been working on creating an effective and memorable interface, while I have been designing and coding the motion-detection exercises..

Figure 2.3 - Logo for the Southcentral Foundation.

Interest in the project from the general public has increased since I unveiled the concept during class. I was approach by James Tweet, my colleague in the capstone class. He invited me to show off the Patient Exercise guide to the Southcentral Foundation once it is complete. The Southcentral Foundation is a non-profit healthcare organization that provides health services to Alaska's native community. Physical and occupational therapy is included in the foundation's services [9], and their emphasis on researching technological solutions for healthcare makes them a suitable outlet for the Patient Exercise Guide to gain recognition. James has given me a rare opportunity to gain support for the app and bring the expertise provided by the app to a community that needs it. I have also been asked about supporting iOS devices such as the iPhone. Apple controls a significant portion of the smartphone market since 42% of smartphone subscribers in the U.S. use an Apple device [10]. Although supporting these products would almost double our audience, it would require us to create a separate version from scratch. At this point in the project, I will refine the Android version before creating an iOS port.

2.5 Agile Methodology

Both Trent and I have been exposed to agile software development in a previous class. In that class, we learned both the values of agile as well as the virtue of testing first. Through Agile, we have been able to control our production so that we can easily make changes later in development without throwing out weeks of work. Agile code development starts as early as initial designs are made, since features are written as user stories. Some of the user stories that I planned at an early stage are: "User can rebuild muscle and expand rage of motion by rotating the Android device" and "User can track the progression of recovery by viewing data taken over the course of several weeks". These two stories created the essential functions of the Patient Exercise Guide. These stories are divided into tasks, and Trent and I share the work by assigning an equal amount of tasks to work on in the span of a week. Some of the tasks for creating motion exercises are "generate numeric output of motion sensor" and "separate exercises based on different motion gestures". Often, we would subdivide tasks into smaller tasks so that our efforts are organized.

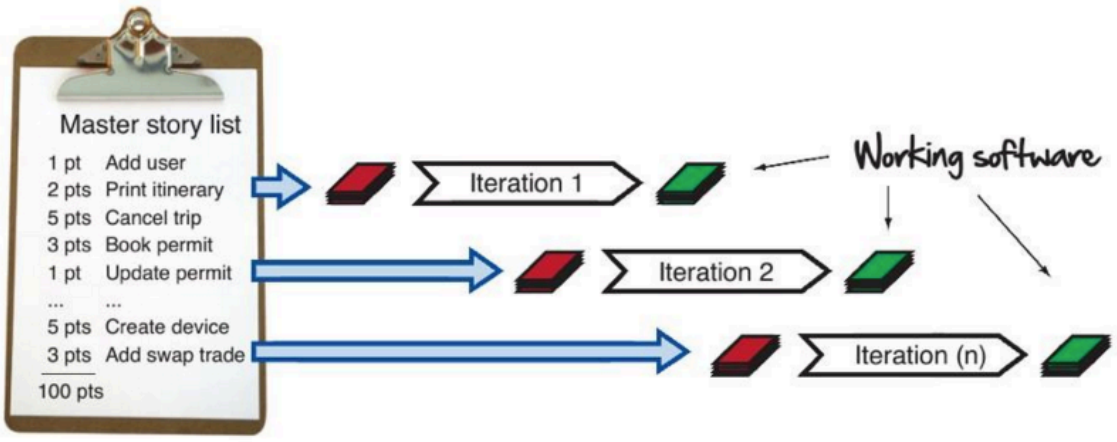


Figure 2.4 – Assigning the right stories to be completed per iteration is key to maintaining velocity in agile development.

A crucial element of agile development is velocity. In *The Agile Samurai*, Jonathan Rasmusson defines Team velocity as the speed at which we (the developers) turn user stories into working software [11]. User stories are assigned a point value that represents the time and effort required to complete that story. Since our user stories cover the main features of the final product, we will assign points to the subtasks that make up those stories instead. Assigning smaller tasks to be completed in a shorter amount of time allows us to develop the app at a more brisk pace, as opposed to working on a large task over the course of a month. To quantify our progress, we have assigned three points per person in a week. In our project, a point equates to about an hour of work on the project. Since we have about 12 weeks to complete the Patient Exercise Guide, both Trent and I combined will spend 72 hours to complete the project before the end of the semester. An overview of the team's workflow is expressed in the Gantt chart seen in figure 2.5 below.

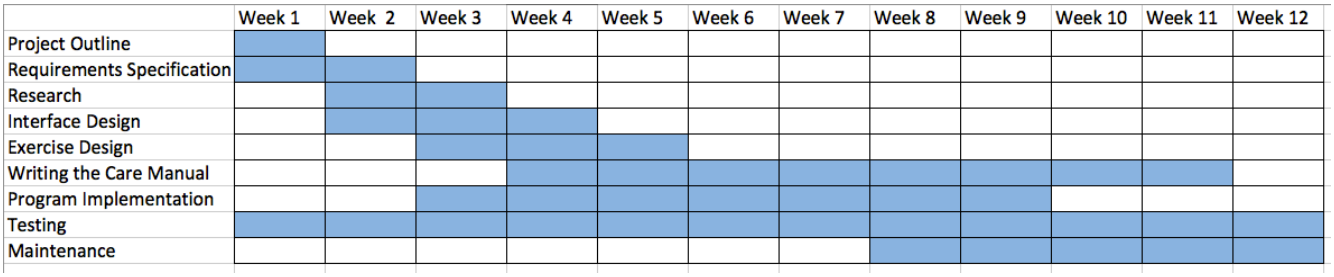


Figure 2.5 - Gantt chart used to evaluate our estimated workflow throughout the semester.

Chapter 3

Design Testing and User Interface

3.1 Introduction

For this project, we have placed a lot of effort into the front-end design. The front end of our project refers to the onscreen elements that the user interacts with to accomplish tasks within the app. In order for the user to accomplish tasks, they must access them without much effort. The reason for allocating our team's resources to the front end is that all of the confirmed features of the Patient Exercise Guide occur on the user's side. According to Seonghoon Kang and Won Kim of the Journal of Object Technology, "UI remains a factor that can help vendors differentiate from others [12]." Because of this, we are driven to form an identity for our product through a unique interface.

To ensure complete usability of our product, we will use a variety of testing techniques. Tests will be conducted on al features of the app, but in this section I will speak in detail about the techniques we use to test the user interface, displaying graphics, and tracking the user's recovery. I will also discuss how we are using the agile methodology to not only direct our development efforts, but also manage our workflow throughout the project.

3.2 User Interface

As previously stated, we are focusing on creating an effective user interface that allows users to do exercises and educate themselves without too many confusing steps. According to Ahmed Ghiduk and Mohammed Elashiry of the International Journal of Computer Applications, "time is important to mobile device users. Thus, they desire to reduce interactions times and increase pace of interaction. [13]" User interfaces for mobile applications must account for limited computing resources alongside a smaller screen size. For Android, we must consider devices of varying specifications and screen sizes.

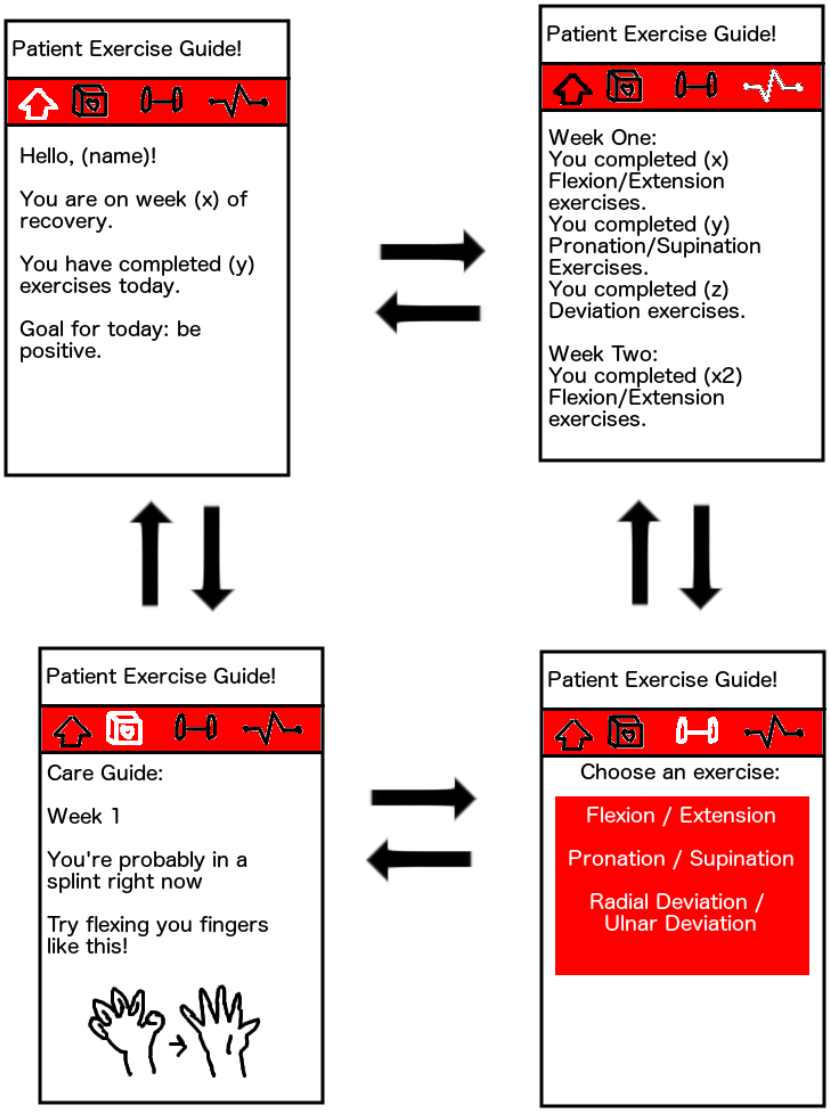


Figure 3.1 – Mockup showing the flow of action between main pages in the Patient Exercise Guide.

There are four main activities of PEG, so we have designed the interface so that all four of these features are available on the taskbar. This includes the home screen, the care guide, the exercise menu and the user's recovery progress. Instead of using long-form names for these items, we have designed icons that represent each one. Our unique graphical assets give our program an identity, and assigning these assets to user tasks makes the tasks more memorable.

3.3 Testing Methodology

Testing is crucial to delivering a reliable product, and the Patient Exercise Guide is no different. There are three main things that need to be tested in this app: the user interface, the exercises, and saving the user's recovery progress. These three elements require specific methodologies for testing, and these methodologies have been in use by software developers for years. For the user interface, we will use instrumentation testing. Instrumentation testing allows the developer to manually invoke the methods that are called automatically in any application. This testing is done on an actual Android device.

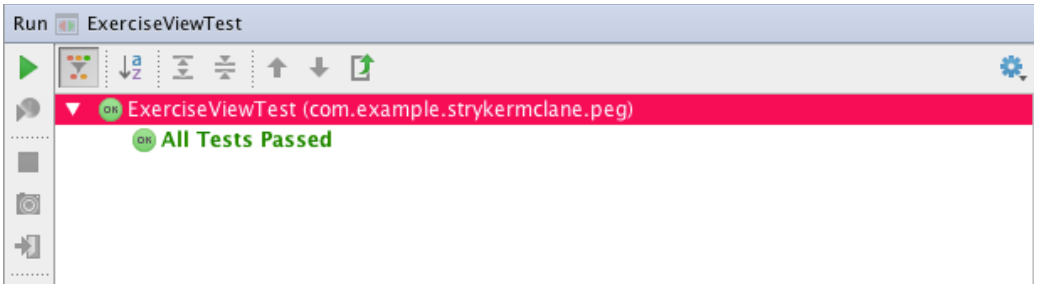


Figure 3.2 – Sample of a successful test in Android Studio.

Since the exercise activity relies on displaying graphical output for user motion input, I have implemented white-box testing to verify correct output. Since the orientation sensors of the Android device output raw data at a fast rate, I have written several methods to average out the data to achieve a smooth numeric reading. White-box testing requires me to carefully step through these methods to make sure I achieve the desired output.

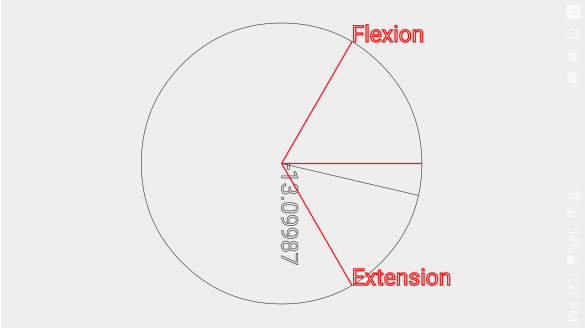


Figure 3.3 – Early version of the graphics shown during the flexion and extension exercise.

3.4 Test Cases

Manually testing the user interface of the app requires us to test all possible gestures on all possible coordinates on the screen. Testing this way allows us to check for unexpected inputs that might cause the program to run poorly. To counter unexpected inputs, we will implement conditional statements that detect these inputs and bring the program to a normal state.

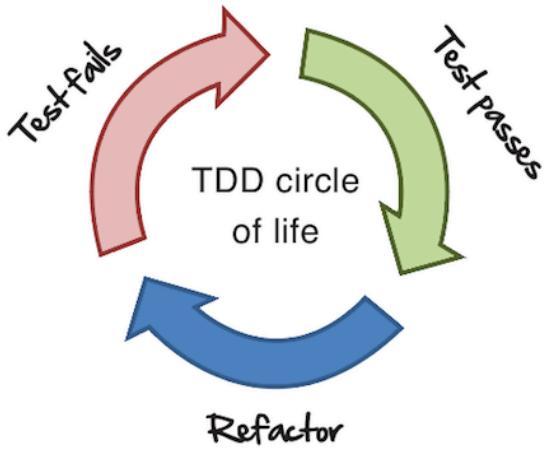


Figure 3.4 – Cycle of testing and refactoring in test-driven development.

For the graphics that show during the exercises, we have chosen white box testing as our test method. White box testing entails a thorough analysis of software and allows the developer to understand his or her system. According to Mohd. Ehmer Khan of the International Journal of Software Engineering and Its Applications, white box testing "is typically very effective in validating design, decisions, assumptions and finding programming errors and implementation errors in software [14]" Since the methods to display graphics will span over multiple files, it's important to understand how these methods play into one another.

Since users will be able to view the progress of recovery, we will save this information by writing to a file. Recovery happens over the course of several weeks, and the amount of time the user is supposed to spend with the app far exceeds that of our development time. In order to simulate the passage of time within the app, we will implement unit testing. Our test cases will consist of several different files that our program takes in as input. These files will test the recovery progress view by showing fake user data at different stages of recovery. For example, the contents of the care guide in the beginning of recovery are far different than toward the end of recovery. By writing several input files, we can debug the output that's shown at each stage of recovery. A key function of the patient exercise guide is how it guides the user through the entirety of recovery, and we can ensure the longevity of the product through unit testing.

3.5 Agile Methodology

Two separate aspects of agile coding are the approaches to project management and software design. Project management requires a skillset that does not overlap with programming, as one must effectively communicate in a group effort. Agile project management requires developers to bend to the whim of the client. The workflow of agile development is designed so that projects can be easily altered if the client wishes. Our supervisor, Diana Steer, has not set any requirements for our design and has given us free reign on our implementation. Because of this, time is the main constraint on developing the Patient Exercise Guide. We must balance the possibility of cutting content with delivering a full-featured app. Agile project management makes it easier to prioritize our efforts so we can make careful decisions about the final product.

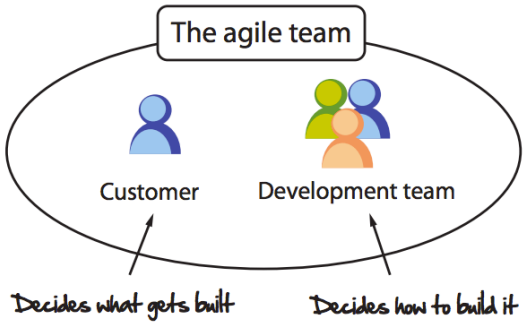


Figure 3.5 – Duties of the project team in agile development

The other main aspect of agile development is the approach to software design. The idea behind agile is to deliver working software frequently. Creating a working piece of software takes design, coding, testing and maintenance. All of these steps need to be performed per iteration, and the speed at which these steps are completed has much influence on how we design our software. Since tests are vital to each iteration, agile emphasizes the importance of writing tests first and then writing the corresponding code. By writing code around test cases, the tests will become more effective as the code evolves. According to the Agile Alliance, "many teams report significant reductions in defect rates, at the cost of a moderate increase in initial development effort. [15]"

Since we are working in a team, we have two perspectives that will influence our code structure. Agile development favors refactoring, a process by which code is minimized but retains all functionality. Scaled Agile, Inc. defines refactoring as "the activity of improving the internal structure or operation of a code base without changing the external behavior [16]." Aside from optimization, the purpose for this is to revise code so that developers outside the project can understand it. Since our project is open-source, we are welcome to other developers improving the Patient Exercise Guide. We will enable developers to understand our code so that we can improve the care that our product will provide.

Chapter 4

User Manual

4.1 Introduction

The Patient Exercise Guide is designed to help patients of carpal tunnel syndrome build wrist strength and range-of-motion after surgery. It also serves as an informational resource about carpal tunnel syndrome and the recovery process. In this chapter I will provide the necessary instructions to use the exercise and care guide functions of the Patient Exercise Guide.

It is important to note that since the Patient Exercise Guide is still in development, much of the content of this version of the user manual may not completely match the version of PEG that you are currently using. Despite this, much of the content will remain the same such as the exercise screen and care guide. We have designed the interface of PEG to be as intuitive as possible, so users should not need much assistance navigating the app.

4.2 Home Screen

The home screen of the Patient Exercise Guide serves as a portal to the main functions of the app. There are four buttons on the home screen, and these are divided into the two main functions of the app: Read the care guide and exercise your wrist. The first button (carpal tunnel care guide) takes the user to the care guide while the other three take the user to the exercise screen specific to each exercise. It is recommended that the user access the care guide once a day. Once the user is comfortable doing exercises, each exercise should be completed several times per day. In order to maximize recovery, the user should do their exercises depending on what the care guide says.

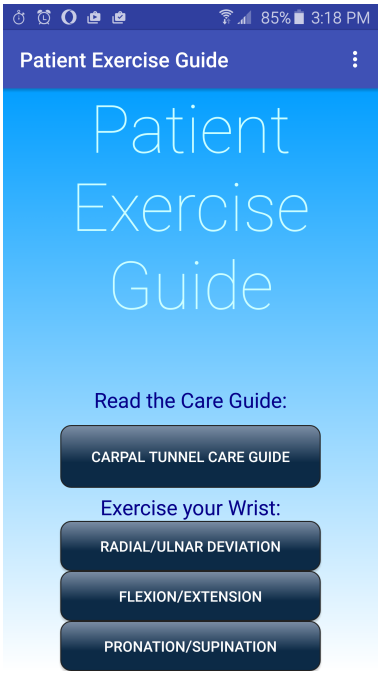


Figure 4.1 - The landing page.

4.3 Care Guide Screen

The care guide provides users with information on carpal tunnel syndrome as well as tips on how they can aid their recovery. Content in the care guide is divided by different time periods of recovery. Users can open the drawer menu as seen below to access different sections. To open the drawer menu, swipe right across the screen from the left edge of the screen. The default screen of the care guide is the introduction page. To access the content that applies to you, select the week that matches the length of time spent in recovery.

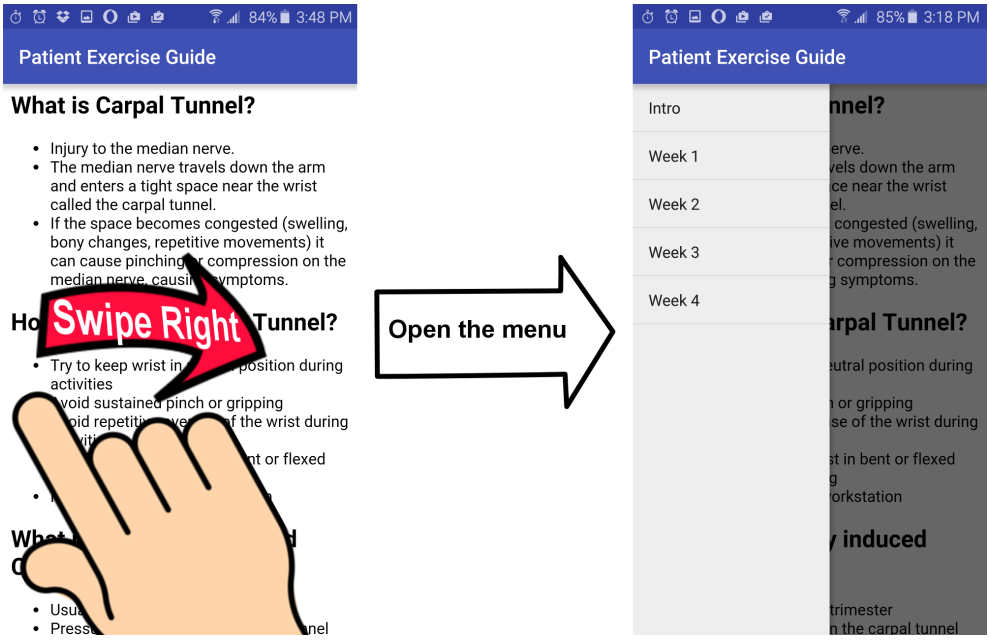


Figure 4.2 - Opening the menu of the care guide.

4.4 Exercise Screen

When the user selects one of the three exercises, they are brought to the exercise screen. This is where the user performs exercises and receives feedback on them in real time. Despite the three exercises sharing the same look, they each accept different motion inputs from the user. In order to complete the exercises effectively, the user should wear the hand strap so that the user does not have to grip the phone while performing exercises. To properly execute the exercise movements, the user keep their arm as still as possible while moving the wrist.

Figure 4.3 displays the screen for the flexion and extension exercise. The green pointer represents the user's current degree of rotation, and the red area represents the maximum level of rotation. A user with a healthy wrist should just barely reach the maximum level of rotation, so users in the early stages of carpal tunnel should not attempt to rotate this far. When the user rotates their wrist in the axis designated by the exercise, the green pointer will show this rotation in relation to the neutral position. The neutral position is when the phone is laid perfectly flat. As the current rotation/green pointer changes, the number in the red area changes as well.

In the flexion/extension and pronation/supination exercise screens, the maximum degree of rotation that the user has achieved is shown on screen. This value is called the user maximum, and is different from the maximum rotation represented from the red area. Since each of the three exercises has the user complete two different wrist movements, there are two user maximums shown on screen. In this case, the user's maximum flexion is shown on the bottom while the maximum extension is shown on top. In order for the user to expand their range of motion in the wrist, they should try and increase their user maximum by practicing each exercise every day and track the user maximum that they achieve each

day of recovery.

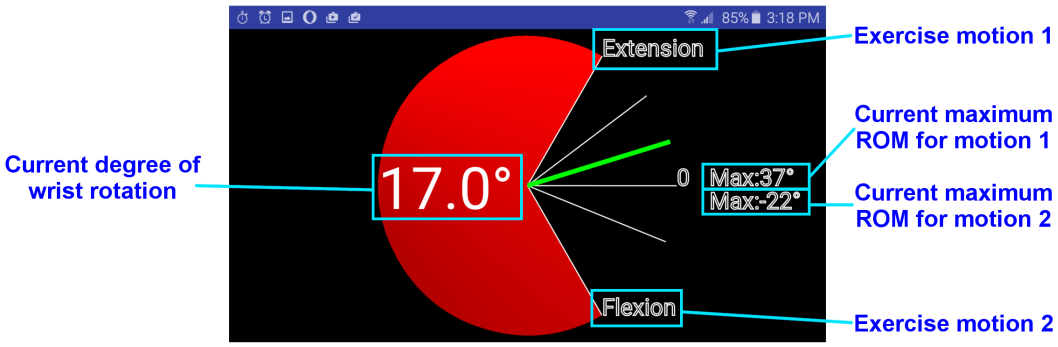


Figure 4.3 - Explanation of the exercise screen.

4.4.1 Flexion and Extension

To complete a flexion/extension exercise, the user must put on the phone strap such that the device is resting flat on the back of the hand, and make sure the phone is positioned in landscape orientation so that the text on screen faces correctly. Rotate the wrist up and down in a fanning motion to complete the exercise.

4.4.2 Pronation and Supination

To complete a flexion/extension exercise, the user must put on the phone strap so that the phone is perpendicular to the hand. Rotate the wrist similar to using a doorknob to complete the exercise.

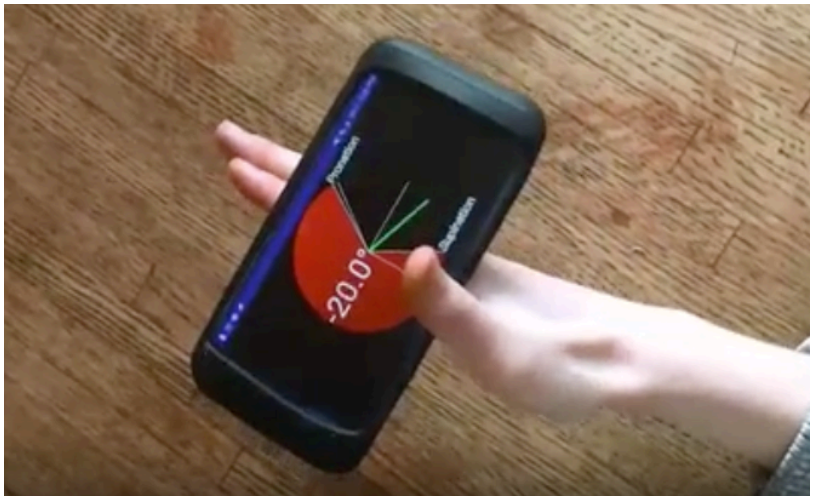


Figure 4.4 - Device positioning on the hand for pronation/supination exercises.

4.4.3 Radial and Ulnar Deviation

To complete a flexion/extension exercise, the user must put on the phone strap such that the device is resting flat on the back of the hand, and make sure the phone is positioned in landscape orientation so that the text on screen faces the correct way. Since the radial/ulnar deviation exercises use the magnetometer of the Android device, the readings of the app may not perfectly match the user's motions. In order to perform this exercise, the user must tap the screen once to calibrate the pointer once they are in the neutral position. Once this is achieved, the user must move their wrist from side to side in order to complete the exercise. Since this exercise is still in beta, the user maximum is not shown on screen during the radial/ulnar deviation exercises

Chapter 5

Summary and Conclusion

5.1 Introduction

Since this is the first time I had built a mobile app for consumers by myself, I had learned a lot about the development process. There's a lot to be gained from making amateur mistakes, and being in charge of planning for the app allows me to experience the full effect of development mishaps. Agile development takes a lot of planning, and since we had only a few short months to complete the app from scratch we could not always adhere to the agile development model. Despite the hardships and long nights spent developing, we are confident that the Patient Exercise Guide achieves its purpose: to help patients recovering from carpal tunnel syndrome.

Figure 5.1 - The launcher icon for the Patient Exercise Guide

5.2 Significant Findings

The most important takeaway from our development process is the significance of careful and realistic planning. In software development, there is a clear difference between intending to include a feature and knowing that feature will work. In my case, I wanted to include 3-D models that provide visual demonstrations for each exercise. While feasible in theory, the process of modeling, texturing, rigging and animating the 3-D model requires practice and experience. On top of that, we could not decide where to implement the 3-D models. If we had another team member to devote his or her efforts toward the feature, we would have found a place to include it.

5.3 Implications

The Patient Exercise Guide is designed to help patients with carpal tunnel syndrome build muscle and range-of-motion in the wrist. Since Android's motion-detection is reliable and easy to implement, users can exercise their wrist and see the results of doing so. However, since the magnetometer is not as reliable as the accelerometer, the app is unable to give accurate measurements for lateral rotations. This means the radial and ulnar deviation exercises lack the reliability of the other exercises. I will implement the gyroscope before I officially release PEG on the Google Play store.

5.4 Recommendations for Future Development

In the planning stages of development, I had lofty ideas for extra features for PEG. Since there's evidence of health apps increasing in use and marketability, I wanted PEG to serve as a full-featured health app. Many Android devices contain a pedometer and some have a vitality sensor, so I wanted to include these features so that the user can track his or her daily activities and heart rate.

Another important aspect that this version of PEG lacks is the ability to exercise different parts of the body when recovering from different conditions. Besides carpal tunnel syndrome, there are many other conditions that require basic exercises during recovery. The phone strap for PEG could fit on to other appendages and the Android device would detect motion of the arms, legs, and feet. Accounting for these conditions drastically expands the scope of PEG and would require a great deal of development time. However, it would ensure an even larger user base and create a long future for the app.

5.5 Summary

Developing the Patient Exercise Guide has been an insightful experience for me as a programmer. Mobile apps are the future of software because they bring such a large variety of accessible programs to an audience that grows daily. Occupational therapy is compatible with mobile technology because they both share a large audience. If the Patient Exercise Guide is successful, then I can bridge the gap between mobile technology and occupational therapy. Refining this app is something I want to spend time on in the future because I believe it can help a lot of people, including myself as a developer.

References

1. Simon, H. (2014, July 14). cCarpal tunnel syndrome. Retrieved from http://www.nytimes.com/health/guides/disease/carpal-tunnel-syndrome/surgery.html

2. Smith, A. (2015, April 1). U.S. smartphone use in 2015. Retrieved from http://www.pewinternet.org/2015/04/01/us-smartphone-use-in-2015/

3. Khalaf, S. (2014, June 19). Health and fitness apps finally take off, fueled by fitness fanatics. Retrieved from http://flurrymobile.tumblr.com/post/115192181465/health-and-fitness-apps-finally-take-off-fueled#.U6m_Oo1dfGB

4. Lofstrom, J. (2015, April 14). Mobile technology upgrades continue to impact healthcare engagement. Retrieved from http://www.himss.org/news/newsdetail.aspx?itemnumber=41520

5. Hammel, J., Knowland, D., Smith, R., Gitlow, L., Jones, R., Leech, S., Goodrich, B., Braveman, B. (2010). Specialized knowledge and skills in technology and environmental interventions for occupational therapy practice. American Journal of Occupational Therapy, 64(6).

6. Relay Health. (2014). Retrieved from http://www.summitmedicalgroup.com/library/adult_health/aha_obg_carpal_tunnel_syndrome/

7. Kreug, S. 2014. Don't make me think revisited: a common sense approach to web and mobile usability. Upper Saddle River, New Jersey: New Riders.

8. Bonato, P. February 2005. Advances in wearable technology and applications in physical medicine and rehabilitation. Journal of NeuroEngineering and Rehabilitation. http://jneuroengrehab.biomedcentral.com/articles/10.1186/1743-0003-2-2

9. Southcentral foundation. 2016. Home-based services. Retrieved from http://www.southcentralfoundation.com/services/home-based-services/

10. Chan, Vera. September 2014. By the numbers: iPhone vs. Android. Yahoo Tech. https://www.yahoo.com/tech/by-the-numbers-iphone-vs-android-97842025474.html

11. Rasumusson, J. 2010. The agile samurai: how agile masters deliver great software. Raleigh, North Carolina: The Pragmatic Bookshelf.

12. Seonghoon, K., Kim W. April 2007. Minimalist and intuitive user interface design guidelines for consumer electronics devices. *Journal of object technology*, 6(3), 40.

13 Ghiduk, A. S., Elashiry, M. May 2012. Design and implementation of the user interfaces and the applications for mobile devices. *International journal of computer applications,* 46, 12-13.

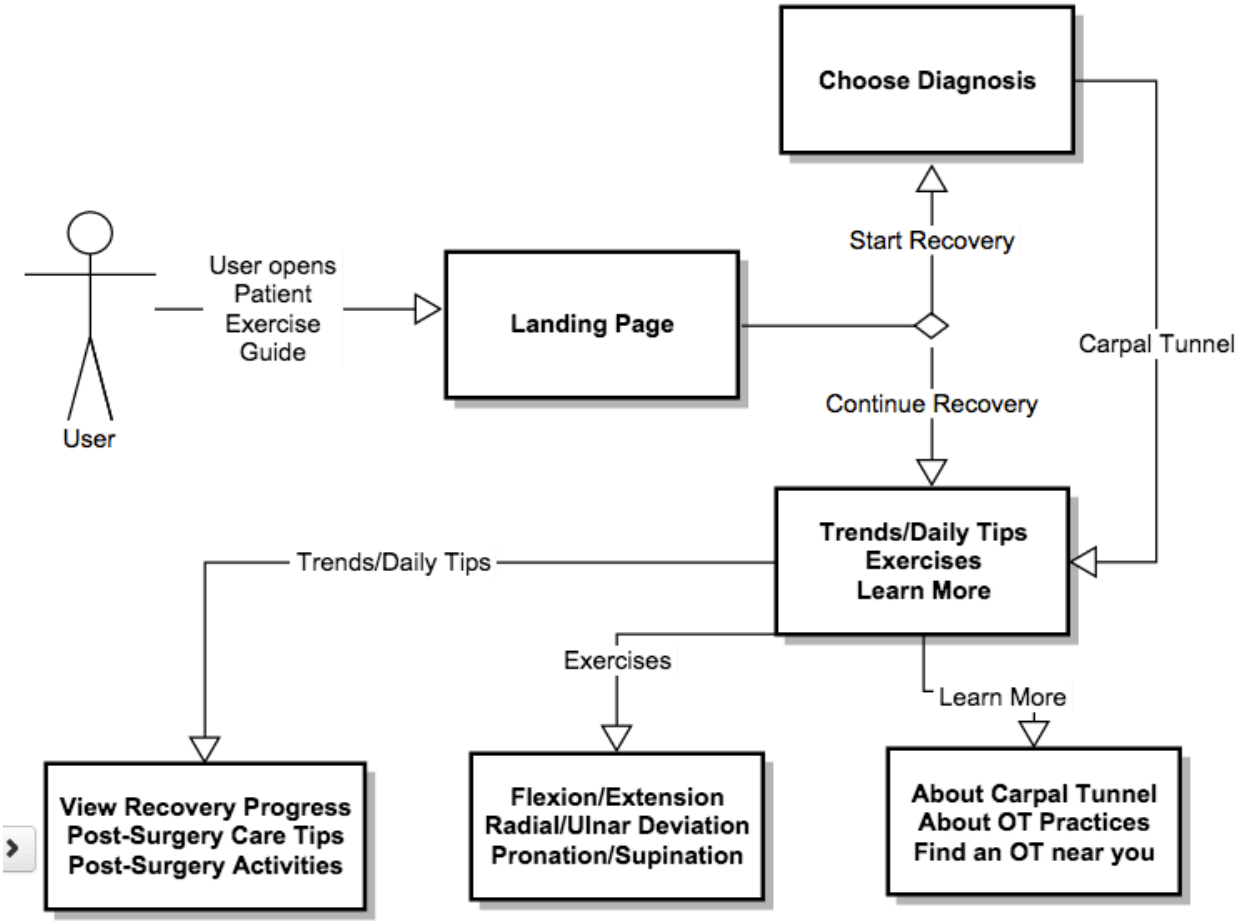
14. Kolawa, A. 2016. White box testing. Retrieved from http://www.wrox.com/WileyCDA/Section/White-Box-Unit-Testing.id-290413.html

15. Agile Alliance. 2016. Test-driven development. Retrieved from https://www.agilealliance.org/glossary/tdd/

16. Scaled Agile, Inc. 2016. Refactoring. Retrieved from http://scaledagileframework.com/refactoring/

License

Appendix A - UML Diagram



Appendix B – Source Code

exerciseActivity.java

```
package com.example.strykermclane.peg;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.os.Vibrator;
import android.util.Log;
import android.view.WindowManager;
import android.widget.Toast;


public class exerciseActivity extends Activity implements SensorEventListener{

    private static SensorManager sensorManager;
    private MyCompassView compassView;
    private Sensor sensor;
    int exerSwitchCompass;
    private static Vibrator vibrateManager;
    private Sensor accelerometer;
    private Sensor magnetometer;
    private float[] accelVals;
    private float[] compassVals;
    static final float ALPHA = 0.15f;
    public float orientationGesture;

    /** Called when the activity is first created. */

    protected float[] lowPass( float[] input, float[] output ) {
        if ( output == null ) return input;

        for ( int i=0; i<input.length; i++ ) {
            output[i] = output[i] + ALPHA * (input[i] - output[i]);
        }
        return output;
    }
```

```
@Override
public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   compassView = new MyCompassView(this);
   setContentView(compassView);

   getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);

   Intent myIntent = getIntent();
   exerSwitchCompass = (myIntent.getIntExtra("exerSwitch",3));
   switch(exerSwitchCompass){
      case 0:
         compassView.exerSwitchCompass = 0;
         break;
      case 1:
         compassView.exerSwitchCompass = 1;
         break;
      case 2:
         compassView.exerSwitchCompass = 2;
         break;
   }

   // Setup the sensors
   sensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
   accelerometer = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
   if (accelerometer != null) {
      sensorManager.registerListener(this, accelerometer, SensorManager.SENSOR_DELAY_UI);
      Log.i("Compass MainActivity", "Registerered for ORIENTATION Sensor");

   }
   else {
      Log.d("Compass MainActivity", "accelerometer is null");
      Toast.makeText(this, "accelerometer is null",
            Toast.LENGTH_LONG).show();
      finish();
   }

   magnetometer = sensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD);
   if (magnetometer != null) {
      sensorManager.registerListener(this, magnetometer, SensorManager.SENSOR_DELAY_UI);
      Log.i("Compass MainActivity", "Registerered for ORIENTATION Sensor");
   }
   else{
         Log.d("Compass MainActivity", "magnetometer is null");
         Toast.makeText(this, "magnetometer is null, BIATCH!",
```

```
                Toast.LENGTH_LONG).show();
            finish();
        }

    }


        @Override
        public void onAccuracyChanged(Sensor sensor, int accuracy) {
        }

        @Override
        public void onSensorChanged(SensorEvent event) {

            vibrateManager = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);

            int sensorType = event.sensor.getType();
            switch (sensorType) {
                case Sensor.TYPE_ACCELEROMETER:
                    accelVals = lowPass( event.values.clone(), accelVals );
                    break;
                case Sensor.TYPE_MAGNETIC_FIELD:
                    compassVals = lowPass( event.values.clone(), compassVals );
                    break;
                default:
                    Log.w("Compass MainActivity", "Unknown sensor type " + sensorType);
                    return;

            } if (accelVals != null && compassVals != null) {
                float R[] = new float[9];
                float I[] = new float[9];
                boolean success = SensorManager.getRotationMatrix(R, I, accelVals, compassVals);
                if (success) {
                    float orientation[] = new float[3];
                    SensorManager.getOrientation(R, orientation);

                    switch (exerSwitchCompass) {
                        case 0:
                            float azimuth = orientation[0];
                            int azimuthDeg = (int) Math.round(Math.toDegrees(azimuth));
                            if (((azimuthDeg - compassView.defaultDeviation > 40) || (azimuthDeg -
compassView.defaultDeviation < -25)) && compassView.vibeSwitch) {
                                vibrateManager.vibrate(100);
                            }
                            orientationGesture = azimuthDeg + 270;
```

```
                    break;
                case 1:
                    float pitch = orientation[1];
                    int pitchDeg = (int) Math.round(Math.toDegrees(pitch));
                    if ((pitchDeg > 60) || (pitchDeg < -60)) {
                        vibrateManager.vibrate(100);
                    }
                    if((pitchDeg > orientationGesture - 270) && (pitchDeg > 0) && (orientationGesture -
270 > 0) && (pitchDeg > compassView.maxUpper)){
                        compassView.maxUpper = pitchDeg;
                    }
                    else if((pitchDeg < orientationGesture - 270) && (pitchDeg < 0) &&
(orientationGesture - 270 < 0) && (pitchDeg < compassView.maxLower)){
                        compassView.maxLower = pitchDeg;
                    }
                    orientationGesture = pitchDeg + 270;
                    break;
                case 2:
                    float roll = orientation[2];
                    int rollDeg = (int) Math.round(Math.toDegrees(roll));
                    if ((rollDeg > 60) || (rollDeg < -60)) {
                        vibrateManager.vibrate(100);
                    }
                    if((rollDeg > orientationGesture - 270) && (rollDeg > 0) && (orientationGesture -
270 > 0) && (rollDeg > compassView.maxUpper)){
                        compassView.maxUpper = rollDeg;
                    }
                    else if((rollDeg < orientationGesture - 270) && (rollDeg < 0) && (orientationGesture
- 270 < 0) && (rollDeg < compassView.maxLower)){
                        compassView.maxLower = rollDeg;
                    }
                    orientationGesture = rollDeg + 270;
                    break;
            }
            compassView.updateData(orientationGesture);
        }
    }


}


  @Override
  protected void onResume() {
    super.onResume();
```

```
        sensorManager.registerListener(this, accelerometer, SensorManager.SENSOR_DELAY_UI);
        sensorManager.registerListener(this, magnetometer, SensorManager.SENSOR_DELAY_UI);
    }

    @Override
    protected void onPause() {
        super.onPause();
        sensorManager.unregisterListener(this);
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        if (sensor != null) {
            sensorManager.unregisterListener(this);
        }
    }

}
```

exerciseView.java

```
package com.example.strykermclane.peg;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Path;
import android.graphics.RectF;
import android.graphics.SweepGradient;
import android.util.Log;
import android.view.MotionEvent;
import android.view.View;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;

public class exerciseView extends View {
```

```
// max unlar deviation line: 30 - 39 deg
// max radial deviation line: 20 - 25 deg
// max flexion line: 60 - 80 deg
// max extension line: 60 - 75 deg
private int maxFlexion = -60;
private int maxExtension = 60;
private int maxPronation = 60;
private int maxSupination = -60;
private int maxUlnarDeviation = 40;
private int maxRadialDeviation = -25;

String newString = "blank";

private int regulateValue(int currentPosition)
{
   int regulatedValue = 0;
   regulatedValue = 90 - currentPosition;
   return regulatedValue;
}

private Paint paintGreen;
private Paint paintRed;
private Paint paintWhite;
private Paint paintGradient;
private float position = 0;
public int defaultDeviation = 0;
public int exerSwitchCompass;
public boolean vibeSwitch = false;
Bitmap calibrateButtonBMP;
private String currentRotation;
public int maxUpper = 0;
public int maxLower = 0;
int[] userMaximumArray = new int[2];

public MyCompassView(Context context) {
   super(context);
   init();
}

private void init() {

   //paintGreen: used for pointer
   paintGreen = new Paint();
   paintGreen.setAntiAlias(true);
```

```
        paintGreen.setStrokeWidth(20);
        paintGreen.setTextSize(100);
        paintGreen.setStyle(Paint.Style.STROKE);
        paintGreen.setColor(Color.GREEN);

        //paintRed:
        paintRed = new Paint();
        paintRed.setAntiAlias(true);
        paintRed.setStrokeWidth(5);
        paintRed.setTextSize(100);
        paintRed.setStyle(Paint.Style.STROKE);
        paintRed.setColor(Color.WHITE);

        //paintWhite: used for current rotation reading
        paintWhite = new Paint();
        paintWhite.setAntiAlias(true);
        paintWhite.setStrokeWidth(5);
        paintWhite.setTextSize(250);
        paintWhite.setStyle(Paint.Style.FILL);
        paintWhite.setColor(Color.WHITE);

        //paintGradient: used for body of compass
        paintGradient = new Paint();
        paintGradient.setStrokeWidth(1);
        paintGradient.setStrokeCap(Paint.Cap.SQUARE);
        paintGradient.setStyle(Paint.Style.FILL);

        getUserMaximum();
    }

    @Override
    protected void onDraw(Canvas canvas) {

        int xPoint = getMeasuredWidth() / 2;
        int yPoint = getMeasuredHeight() / 2;

        //Draw Background
        canvas.drawColor(Color.BLACK);

        //Draw Calibrate BMP if exerSwictch = 0 (Radial / Ulnar Deviation Exercise Selected)
        calibrateButtonBMP =
BitmapFactory.decodeResource(getResources(),R.drawable.calibratebutton);
        if(exerSwitchCompass == 0) {
            canvas.drawBitmap(calibrateButtonBMP, 0, 0, null);
        }
```

```
    //Calculate radius of circle based on screen size
    float radius = (float) (Math.max(xPoint, yPoint) * 0.5);

    //rect1: rectangle used to create compass arc
    RectF rect1 = new RectF(xPoint - radius, yPoint - radius, xPoint + radius, yPoint + radius);

    int[] colors = {Color.RED, Color.BLACK, Color.RED};
    float[] positions = {0f, 0.5f, 1f};
    SweepGradient gradient = new SweepGradient(100, 100, colors, positions);
    paintGradient.setShader(gradient);;

    //

    //tempX, tempY: coordinates for endpoint of neutral position line
    float tempX = (float) (xPoint + radius * Math.sin((double) (regulateValue(0)) / 180 * 3.143));
    float tempY = (float) (yPoint - radius * Math.cos((double) (regulateValue(0))  / 180 * 3.143));

    Path p = new Path();
    p.moveTo(xPoint, yPoint);
    p.lineTo(tempX, tempY); // neutral position line
    //canvas.drawText(String.valueOf(defaultDeviation), tempX, tempY, paintRed);
    canvas.drawText(String.valueOf(newString), tempX, tempY, paintRed);
    p.moveTo(xPoint, yPoint);


    switch(exerSwitchCompass) {
       case 0:
          //30 degrees
          tempX = (float) (xPoint + radius * Math.sin((double) (regulateValue(maxUlnarDeviation))  /
180 * 3.143));
          tempY = (float) (yPoint - radius * Math.cos((double) (regulateValue(maxUlnarDeviation)) /
180 * 3.143));
          p.lineTo(tempX, tempY); // top line
          canvas.drawText(("Ulnar"), tempX, tempY - 100, paintRed);
          canvas.drawText(("Deviation"), tempX, tempY, paintRed);
          p.moveTo(xPoint, yPoint);
          //-25 degrees
          tempX = (float) (xPoint + radius * Math.sin((double) (regulateValue(maxRadialDeviation))  /
180 * 3.143));
          tempY = (float) (yPoint - radius * Math.cos((double) (regulateValue(maxRadialDeviation)) /
180 * 3.143));
          p.lineTo(tempX, tempY); // bottom line
          canvas.drawText(("Radial"), tempX, tempY + 100, paintRed);
          canvas.drawText(("Deviation"), tempX, tempY + 200, paintRed);
```

```
            canvas.drawArc(rect1, 25, 295, true, paintGradient);
            break;
        case 1:
            tempX = (float) (xPoint + radius * Math.sin((double) (regulateValue(maxPronation)) / 180 *
3.143));
            tempY = (float) (yPoint - radius * Math.cos((double) (regulateValue(maxPronation)) / 180 *
3.143));
            p.lineTo(tempX, tempY); // top line
            canvas.drawText(("Pronation"), tempX, tempY, paintRed);
            p.moveTo(xPoint, yPoint);
            tempX = (float) (xPoint + radius * Math.sin((double) (regulateValue(maxSupination)) / 180 *
3.143));
            tempY = (float) (yPoint - radius * Math.cos((double) (regulateValue(maxSupination)) / 180 *
3.143));
            p.lineTo(tempX, tempY); // bottom line
            canvas.drawText(("Supination"), tempX, tempY, paintRed);
            canvas.drawArc(rect1, 60, 240, true, paintGradient);
            break;
        case 2:
            tempX = (float) (xPoint + radius * Math.sin((double) (regulateValue(maxExtension)) / 180 *
3.143));
            tempY = (float) (yPoint - radius * Math.cos((double) (regulateValue(maxExtension)) / 180 *
3.143));
            p.lineTo(tempX, tempY); // top line
            canvas.drawText(("Extension"), tempX, tempY, paintRed);
            p.moveTo(xPoint, yPoint);
            tempX = (float) (xPoint + radius * Math.sin((double) (regulateValue(maxFlexion)) / 180 *
3.143));
            tempY = (float) (yPoint - radius * Math.cos((double) (regulateValue(maxFlexion)) / 180 *
3.143));
            p.lineTo(tempX, tempY); // bottom line
            canvas.drawText(("Flexion"), tempX, tempY, paintRed);
            canvas.drawArc(rect1, 60, 240, true, paintGradient);
            break;
    }
    if((exerSwitchCompass == 1) || (exerSwitchCompass == 2)) {
        p.moveTo(xPoint, yPoint);
        tempX = (float) (xPoint + radius * Math.sin((double) (regulateValue(maxUpper)) / 180 *
3.143));
        tempY = (float) (yPoint - radius * Math.cos((double) (regulateValue(maxUpper)) / 180 *
3.143));
        p.lineTo(tempX, tempY); // top line
        canvas.drawText(("Max:" + maxUpper + "\u00B0"), getMeasuredWidth() - 500,
getMeasuredHeight() / 2, paintRed);
        p.moveTo(xPoint, yPoint);
```

```
            tempX = (float) (xPoint + radius * Math.sin((double) (regulateValue(maxLower)) / 180 *
3.143));
            tempY = (float) (yPoint - radius * Math.cos((double) (regulateValue(maxLower)) / 180 *
3.143));
            p.lineTo(tempX, tempY); // bottom line
            canvas.drawText(("Max:" + maxLower + "\u00B0"), getMeasuredWidth() - 500,
(getMeasuredHeight() / 2)+ 100, paintRed);
            canvas.drawArc(rect1, 60, 240, true, paintGradient);
        }


        canvas.drawPath(p, paintRed);



        if(exerSwitchCompass == 0) {
            position -= defaultDeviation;
        }


        //Draw Pointer
        canvas.drawLine(xPoint,
                yPoint,
                (float) (xPoint + radius
                        * Math.sin((double) (-position) / 180 * 3.143)),
                (float) (yPoint - radius
                        * Math.cos((double) (-position) / 180 * 3.143)), paintGreen);


        currentRotation = String.valueOf(position - 270) + "\u00B0";
        //Draw current Rotation Value
        canvas.drawText(currentRotation, xPoint - (radius), yPoint + 100, paintWhite);
    }

    public void updateData(float position) {
        this.position = position;
        invalidate();
    }

    @Override
    public boolean onTouchEvent(MotionEvent ev) {

        switch (ev.getAction()) {

            case MotionEvent.ACTION_DOWN: {
```

```
            calibrateButtonClicked();
            syncUserMaximum();

            break;
         }
         case MotionEvent.ACTION_MOVE: {
            break;
         }
         case MotionEvent.ACTION_UP:
            break;
      }
      return true;
   }

   void calibrateButtonClicked(){
      defaultDeviation = (int) position - 270;
      if(exerSwitchCompass == 0){
         vibeSwitch = true;
      }
      invalidate();
   }

   void syncUserMaximum(){
      Context context = this.getContext();
      String maxUpperString = Integer.toString(maxUpper);
      String maxLowerString = Integer.toString(maxLower);
      String userMax = maxUpperString +","+ maxLowerString;
      try {
         OutputStreamWriter outputStreamWriter = new
OutputStreamWriter(context.openFileOutput("recovery.txt", Context.MODE_PRIVATE));
         outputStreamWriter.write(userMax);
         outputStreamWriter.close();
      }
      catch (IOException e) {
         Log.e("Exception", "File write failed: " + e.toString());
      }
   }private String readFromFile() {
      Context context = this.getContext();

      String ret = "";

      try {
         InputStream inputStream = context.openFileInput("recovery.txt");

         if ( inputStream != null ) {
```

```
            InputStreamReader inputStreamReader = new InputStreamReader(inputStream);
            BufferedReader bufferedReader = new BufferedReader(inputStreamReader);
            String receiveString = "";
            StringBuilder stringBuilder = new StringBuilder();

            while ( (receiveString = bufferedReader.readLine()) != null ) {
               stringBuilder.append(receiveString);
            }

            inputStream.close();
            ret = stringBuilder.toString();
         }
      }
      catch (FileNotFoundException e) {
         Log.e("login activity", "File not found: " + e.toString());
      } catch (IOException e) {
         Log.e("login activity", "Can not read file: " + e.toString());
      }

      return ret;
   }

   void getUserMaximum(){
      newString = readFromFile();
      int[] userMaxInput = new int[2];
      String[] userMaxInputString;
      userMaxInputString = newString.split(",",2);
      for(int i = 0;i<2;i++){
         userMaxInput[i] = Integer.valueOf(userMaxInputString[i]);
      }
      maxUpper = userMaxInput[0];
      maxLower = userMaxInput[1];
   }

}
```