Wrkr: Wearable Wrist Activity Tracking and User Exercise Application

University of Massachusetts Lowell
Michael Stowell and Sergei Fomichev

1. Goal

Using a keyboard and mouse while sitting at a computer workstation can impose several health risks related to the user's wrists. These risks are greatly heightened when using a computer is necessary for daily work either at home or in an office. It is important to maintain proper posture and take frequent breaks to exercise the wrists.

Failure to take such breaks can lead to wrist problems such as computer repetitive strain injuries (RSIs), De Quervain tendinitis, carpal tunnel, and more. RSIs are caused by repetitive movements of the same parts of the wrist, which can damage parts of the body including soft tissue, nerves, muscles, and tendons [1]. De Quervain tendinitis is caused when these tendons in the wrist become swollen from overuse [2]. Carpal tunnel syndrome involves pressure of the median nerve which sits within the carpal tunnel in the wrist [3].

What makes these risks especially dangerous is that they are often not sudden, but rather develop gradually [4]. A slight onset of pain can go unnoticed for a while and also has the risk of numbing the user to the pain until it becomes more excruciating. At this point, the injury may be too late to treat via non-surgical methods.

Since systems, design, IT, software engineer, and other computer-related jobs have shown a positive projected growth over the years [5], it is critical to help prevent the onset of such wrist problems. The Wrkr application aims to deliver such a goal by combining a user's smart watch, mobile phone, and the use of the Leap Motion technology to actively enable the user to practice daily wrist exercises.

2. Features

2.1. Smart Watch Application

To create an ubiquitous computing application that requires minimal user intervention, Wrkr uses a smart watch to determine when a user needs an exercise. The watch will track the user's movements and keep a record of the time the user spent sitting in front of a computer at a keyboard. The watch will vibrate and notify the user when it is time to take a break from his or her work and perform a set of wrist exercises.

2.2. Mobile Application

A mobile application on the user's Android smart phone will present a miniature dashboard of notifications to the user, giving him or her a brief breakdown of exercises that have been triggered as well as exercises he or she has completed. The mobile application also acts as a way for a user to set up a profile or gain insight into how the application works.

2.3. Web Dashboard

Having a web dashboard built into the Wrkr suite enables a user to gain more detailed information into his or her habits while at work. A synchronized profile between the website and mobile application allows the user to view detailed information of how often he or she is performing wrist exercises. A karma score will also provide a user a colored number that gives either positive, neutral, or negative feedback to his or her progress and motivation to keep up with wrist exercises.

2.4. Exercise Tracker

In combination with the web dashboard, the Wrkr website enables a connection with the Leap Motion device to give wrist exercise guidance. With the Leap Motion on the desk, the user can hover his or her hand(s) over the device and follow the exercise prompts provided by the website. The website will feedback the user how many repetitions of each exercise are complete as well as indicating if the user is performing the exercises correctly.

3. Design

Figure (1) on the following page details the flow of the mobile,wear, and web applications as well as how events are triggered and how the Leap Motion connects to the website.

The mobile application uses an activity with a navigation drawer to present four main content fragments: a home screen, profile screen, settings screen, and help screen. The home screen is responsible for showing the user a list of exercises recorded by the application that he or she has not yet complete at the website. Additionally, the home screen will show a karma score that weighs how successful a user is in using the application. The default karma score is 80, shifts towards 150 for persistent wrist exercise and application use, and shifts towards 0 when the user is not performing recommended exercises. The profile screen performs Google account synchronizing. The settings or debug screen is a development option that allows the user to send a dummy watch notification or manually start and stop the data collection procedure. The help screen simply provides information about the application.

To collect user wrist activity, a BroadcastReceiver is set up to receive messages both on device boot and whenever the user launches the application. Its job is to send messages to two additional BroadcastReceiver classes that will use the AlarmManager to schedule the start and stop time of the data recording service class.

The data service emits a persistent notification to inform the user that Wrkr is collecting data to monitor wrist activity. This notification will dismiss when the stop-alarm runs. Within this data service is an API instance to communicate to the web server and a BroadcastReceiver class to receives messages from the user's wearable device, such as when it is time for the user to start an exercise. All exercise information is uploaded via the API class. Another service is responsible for listening for wear messages and firing the aforementioned BroadcastReceiver.

The wearable application uses a similar service to listen for messages from the mobile device. When it receives a message to begin recording accelerometer data for wrist activity tracking, another service is created. This service is critical to the functionality of the application. As a result, it displays a persistent notification on the watch to inform the user that data is being collected, holds a partial wakelock, uses an alarm to frequently reschedule the service, and re-registers the accelerometer listener every minute. A listener class collects accelerometer data on the X, Y, and Z axes as well as performing calculations to obtain a magnitude and weighted moving average value for each data point. Once 10 seconds of data are collected (50 data points at 5 hertz), it sends a broadcast to the BroadcastReceiver internal to the critical wearable service containing the raw data. A logistic regression machine learning classifier trained on the developer's wrist data is then used to determine the likelihood that the user has been sitting at the keyboard for these previous 10 seconds. If the classifier returns with a certainty greater than 85%, a time value kept in SharedPreferences is incremented by 10. Once this time value reaches 30 minutes, a message is sent to the mobile device, along with the current timestamp, that a new exercise is due. The timer is then reset back to 0.

When the mobile device receives this message and sends the information to the web server via

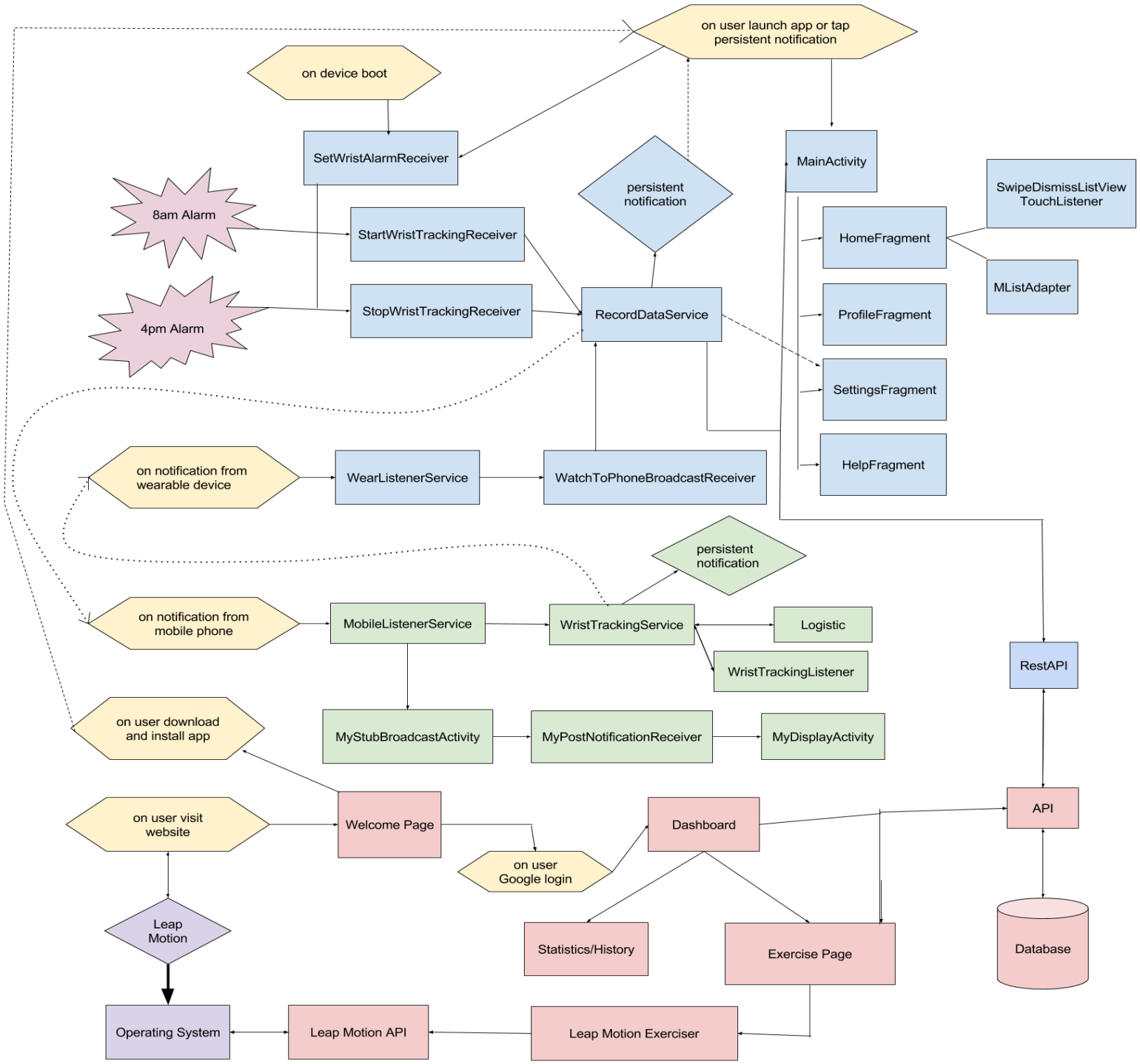

Figure (1) – Design of the mobile application (blue), wear application (green), and web application (red) and how each component connects. See full resolution image at https://github.com/uml-ubicomp-2016-spring/wrkr/blob/master/wrkr_component_diagram.png

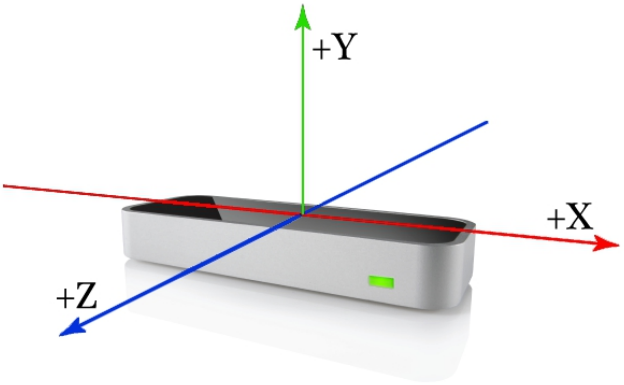

Figure (2) – Leap Motion device and the coordinate system it uses to record data about the objects it classifies hovering over it.

the API, it again sends a message to the wearable device as an acknowledgment of success. The wear device then vibrates the user's wrist and displays a top level notification saying that a new exercise has been issued. If at any point the communication between the wear and mobile are broken, the wear messages are queued on a five minute retry timer that cancels only once the message is successfully delivered.

The user can now navigate to the website where he or she will be presented with a home page, dashboard, statistics, and exerciser page. The home page gives information about the Wrkr project and enables Google sign-in. This sign-in is synchronized with the mobile application to give each user a unique profile and stored in a SQL database in the back-end. The dashboard view gives the user information about exercises that are due, his or her current karma score, and a statistics log of all exercises recorded along with a completion status.

The exerciser page is critical to the web portion of Wrkr. On this page, the user can use a connected Leap Motion device to perform wrist exercises that have been customized for the user to perform. A Leap motion device is able to recognize objects like hands and fingers. Using motion sensors and infrared light, the device reports the object's coordinates, gestures, and motion at a particular frame rate. The coordinate system used is Cartesian with the origin located on top of the device. The X axis lies in parallel orientation left and right, Y lies up and down, and Z lies towards and back from the user. A representation of this can be seen in Figure (2). A Javascript controller allows interaction with the data from the device in real time using object oriented design.

The Leap Motion API assigns a frame as a root object. Each frame contains child elements such as hands or fingers. When a hand is detected by the Leap Motion, the frame object will instantiate an instance of a “hand” class object with an ID of zero and all the characteristics about the current view of the hand. If another hand is detected, a second hand class object is instantiated in the same way, allowing simultaneous tracking. The hand class contains multiple “finger” children classes where every finger identifies by it's name. The finger class contains information such as the position of the finger, its extended state, and more.

When the user is ready to begin a wrist exercise, the Leap Motion controller starts and waits to receive a signal from the device. The main goal, aside from completing an exercise, is ensuring that the exercise is performed correctly. For this, we need to provide enough information about the exercise and ensure that the movements of the user's hand matches the exercises requirements. Each frame captured by the device is analyzed by Wrkr's script, and when a set of frames indicate a successful repetition of the exercise, the repetition counter is incremented on screen for the user to confirm that he or she is performing the exercise correctly.

A signal strength box helps guide the user's hands

to the right position over the Leap Motion device. Wrkr requires that each exercise be performed individually with each hand and then together over the Leap Motion device. Picture demonstrations of each exercise are provided. Once all exercises are complete, a message is sent to the API to update the user profile and mark the exercise as complete. The user at this point can also confirm that the mobile application will no longer have a notification for that exercise once it is complete.

4. File Structure

4.1. Smart Watch Application

Written by Michael Stowell

4.1.1: *AndroidManifest.xml*
This file provides the definitions of used services, activities, broadcast receivers, and permissions to use the application.

4.1.2: *MobileListenerService.java*
This service extends the WearableListenerService to receiver messages sent by the mobile phone.

4.1.3: *WristTrackingService.java*
This service creates a persistent notification, takes 10 second accelerometer data chunks from the WristTrackingListener, performs the logistic regression training and classification, tracks how long the user has been at a keyboard, and sends a message to the mobile device when an exercise is due.

4.1.4: *WristTrackingListener.java*
Registered by the WristTrackingService, this SensorEventListener listens to the accelerometer at 5 hertz and returns X, Y, Z, acceleration magnitude, and a weighted moving average [6] to WristTrackingService via a BroadcastReceiver.

4.1.5: *MyStubBroadcastActivity.java*
This BroadcastReceiver receives a message from the MobileListenerService when it is time to vibrate the user's watch and display a notification saying it is time for an exercise.

4.1.6: *MyPostNotificationReceiver.java*
This class builds the custom wrist exercise notification.

4.1.7: *MyDisplayActivity.java*
This is the activity associated with the custom wrist exercise notification display.

4.1.8: *activity_display.xml*
This layout file designs the customized wrist exercise notification.

4.2. Mobile Application

Written by Michael Stowell

4.2.1: *AndroidManifest.xml*
This file provides the definitions of used services, activities, broadcast receivers, and permissions to use the application.

4.2.2: *MListAdapter.java*
This is a ListAdapter extension class that defines the ListView notification items on the HomeFragment.

4.2.3: *RestAPI.java*
The RestAPI class contains all API communication with the web server, including all GET, POST, and PUT requests.

4.2.4: *SetWristAlarmReceiver.java*
Called either on system boot or on application launch, this BroadcastReceiver schedules the first start and stop alarms for recording data.

4.2.5: *StartWristTrackingReceiver.java*
This BroadcastReceiver decides whether to begin recording data now (if between eight AM and four PM on a weekday) or schedule the RecordDataService for the following weekday.

4.2.6: *StopWristTrackingReceiver.java*
When the application is to stop recording the user's wrist activity data, this BroadcastReceiver is called by the AlarmManager to stop the RecordDataService and schedule the next alarm

to begin recording data on the next weekday.

4.2.7: *RecordDataService.java*
This service creates a persistent notification and houses the API instance and WatchToPhone-BroadcastReceiver responsible for listening for communications from the wear device. These messages either contain raw data for viewing on the SettingsFragment or a notification that the user has a new wrist exercise due. If the latter, it will publish this wrist exercise and its timestamp to the web server via the RestAPI.

4.2.8: *WearListenerService.java*
Similar to the MobileListenerService, this extension of the WearableListenerService listens for messages sent by the wear device.

4.2.9: *MainActivity.java*
This main activity of the mobile application presents a navigation drawer and the four fragment classes for the application's UI. It also hovers a button over all four fragments that, when tapped, launches the Wrkr website on the mobile device.

4.2.10: *HomeFragment.java*
The initial screen seen on application launch, the HomeFragment shows the user's karma score and a list of the exercises the user has not complete.

4.2.11: *ProfileFragment.java*
This fragment is simply responsible for configuring the user's Google account synchronized with the application and website.

4.2.12: *SettingsFragment.java*
This fragment provides a debug view that could be removed from a final product version of this application. It provides a button to send a notification to the wear device, manually start the data collection, and manually stop the data collection. It also has a scrollable TextView that shows all messages received by the wearable.

4.2.13: *HelpFragment.java*
This fragment provides a ScollView of information regarding this application's usage.

4.2.14: *activity_main.xml*
This is the design of the main activity containing the navigation drawer.

4.2.15: *fragment_home.xml*
This is the design of the HomeFragment.

4.2.16: *fragment_profile.xml*
This is the design of the ProfileFragment.

4.2.17: *fragment_settings.xml*
This is the design of the SettingsFragment.

4.2.18: *fragment_help.xml*
This is the design of the HelpFragment.

4.2.19: *list_item.xml*
This is the design for the custom ListView notifications that show in the HomeFragment.

4.3. Mobile and Wear Common Library

Written by Michael Stowell

4.3.1: *APIClientCommon.java*
A wrapper around the GoogleApiClient class, this class provides a singleton instance of the GoogleApiClient and creates a common sendMessage method for all implementers to utilize.

4.3.2: *Globals.java*
This file contains all hard-coded variables of both the wear and mobile application.

4.3.3: *User.java*
This class is utilized by the API and all users of the API to hold user information such as his or her ID, email, exercise count, and so forth.

4.3.4: *train.csv*
This CSV file contains the preliminary training data used to build a weight vector in the logistic regression classifier.

4.4. Website API
Written by Sergei Fomichev

4.4.1: *README.md*
This README file defines the API specification, including all JSON responses for each GET, POST, and PUT request that can be handled by the API.

4.4.2: *api.php*
This file contains all the code for handling API requests from the mobile device.

4.4.3: *respond.php*
This file is used by the API to format responses.

4.5. Website Code
Written by Sergei Fomichev

4.5.1: *index.html*
The main page of the website, this file presents to the user a description of Wrkr, a navigation menu, and a Google profile sign in tool.

4.5.2: *dashboard.html*
The dashboard gives the user access to both the hand-wrist exerciser page and the progress page. It shows the user's current outstanding exercise count and the current karma score.

4.5.3: *stat.html*
The progress/statistics page is a detailed log of every exercise that was recorded by the mobile device, including timestamps and whether or not the user has completed the exercise.

4.5.4: *scripts.js*
This script performs the main website functionality, such as a Google console API authorization, retrieving and sending messages to the server, and so forth.

4.6 Leap Motion Code
Written by Sergei Fomichev

4.6.1: exerciser.html
The exerciser view presents the page for the user to perform exercises on, including images of the exercises to perform, a Leap Motion signal strength indicator, and progress bars for each exercise done by the left, right, and both hands.

4.6.2: script.js
This script performs the main work by organizing the exercising process. It creates a framework for the exercising cycle and increments the interactive counter. It also creates a virtual box for the user to view his or her hand location on screen to confirm that the hands are visible to the Leap Motion device.

4.6.3: exercises.js
This is the object representation of the exercises. Properties of the object refers to exercise parameters, such as a number of repetitions per exercise or the description of the exercise. Also included are methods to perform each exercise. The object has a JSON syntax and is used when the script.js is loaded.

5. Project Evaluation

Wrkr is a novel application in its area. While there are studies that confirm that onset wrist problems such as carpal tunnel can be caused by keyboard use [7], there are little efforts to actively monitor keyboard use the way Wrkr does.

Voom by Kinetic Concepts LLC is the closest mobile application in relation to the goals of Wrkr. In Voom, the user is asked to perform an exercise hourly while at work. Most of these exercises are targeted towards postural pain points such as the back, but there are exercises for wrists as well. Voom presents video demos on how to perform each exercise and lets the user configure the time period which he or she is at work and wishes to receive these hourly notifications. A “health score” out of 100 gives the user feedback based on how often he or she uses the application. This appears to be the end of the feature set of Voom. While its simplicity can be viewed as a pro, it does not actively monitor the user's activity as Wrkr does. As a

result, a user not sitting at a computer may still get notifications to perform exercises in Voom, while a user of Wrkr can rely on its machine learning classification capabilities to only trigger exercises based on time spent at the computer. The Voom application only exists on the smart phone, not on the smart watch or website, so it has limited features. Since Voom does not watch exercises, it cannot guarantee a user is completing exercises and instead relies on self-reporting. Wrkr utilizes the Leap Motion to give feedback to users in this way.

Pain Questionnarie Carpal Tunnel by Canvas is a simple mobile application that has one function: to input answers to a questionnaire and feedback results to the user's physician. The questionnaire asks the user to rate pain levels, when the pain is observed, activities that happened around the time of the pain, and so forth. The advantage of this application is that it gives the physician a better diagnosis of the user's wrist issues so that he or she may provide a proper treatment plan. Where this application fails is that it is targeted at users who are already in pain and requires a large amount of self-reporting. Additionally, the user must make an appointment with a physician to receive a treatment plan, which can take days, weeks, or longer. Wrkr is instead aimed at users who use a computer for a career, regardless of their current wrist pain levels. Wrkr requires minimal user intervention and provides exercise plans immediately, completely cutting out the wait for a physician. While professional advice supersedes exercise plans presented by Wrkr, the immediate feedback will help to prevent onset wrist issues.

Relief Wrist Pain by Tak Wai Wong is another mobile application that gives users acupressure tutorials for relieving wrist pain. The user goes through a training phase before delivering the acupressure exercises on him- or herself. However, while acupuncture and acupressure are proposed treatments to De Quervain tendinitis [8], these methods are effective for those who have already contracted such tendinitis. Wrkr instead aims to prevent RSIs and De Quervain tendinitis by getting the user to take breaks from the keyboard and exercise to prevent swollen wrist tendons. Exercises given to the users by Wrkr are those that are recommended by therapists and health professionals [9] to prevent onset RSIs.

Overall, Wrkr is a more flexible and powerful solution to preventing onset wrist health issues than the other available options aside from seeking professional guidance. It requires more technology to utilize, but Wrkr is minimalist, requires little user intervention aside from performing the exercises, and can be adapted to users ranging from office workers with or without current wrist issues to an everyday computer user that may want or need wrist exercise reminders. Like Voom, a motivational score will help to keep the user motivated to use the application and maintain a healthier lifestyle. Wrkr can be extended in the future to allow further scheduling flexibility and training on user-recorded data.

6. References

[1] Marxhausen, Paul. "Computer Related Repetitive Strain Injury." RSI. Page. University of Nebraska-Lincoln, 2013. Web.

[2] Ma, Benjamin, David Zieve, and Isla Zieve. "De Quervain Tendinitis." U.S National Library of Medicine. ADAM, Inc., 5 May 2014. Web.

[3] "Carpal Tunnel Syndrome: Symptoms, Diagnosis, Treatment, and Prevention." Harvard Health. Harvard Medical School, n.d. Web.

[4] "Carpal Tunnel Syndrome Fact Sheet". National Institute of Neurological Disorders and Stroke. January 28, 2016. Web. http://www.ninds.nih.gov/disorders/carpal_tunnel/detail_carpal_tunnel.htm#227043049

[5] Lauren Csorny, “Careers in the growing field of information technology services,” Beyond the Numbers: Employment & Unemployment, vol. 2,

no. 9 (U.S. Bureau of Labor Statistics, April 2013). Print.

[6] Tomlein et al., “Advanced Pedometer for Smartphone-Based Activity Tracking”. Slovak University of Technology in Bratislava. Retrieved April 22, 2016. Web.

[7] Toosi et al., “Computer keyboarding biomechanics and acute changes in median nerve indicative of carpal tunnel syndrome.” *Clinical Biomechanics*. Volume 30, Issue 6. (July 2015): 546-550. Print.

[8] Da Silva, João Bosco Guerreiro and Fernando Batigália, “Acupuncture in De Quervain's disease: a treatment proposal.” *Preto Medical School (FAMERP).* 2013 December 9. Print.

[9] Schneider, Mark and Melissa Moody, “How to Address and Prevent Repetitive Strain Injuries of the Body and Eyes.” *Positive Health Publications, Ltd.* May 2006. Print.