CS486C – Senior Capstone Design in Computer Science

Project Description

Project Title: WearWare: DataWrangler Module	
Sponsor Information: WEARABLE INFORMATICS LABORATORY 	**Dr. Kyle N. Winfree**, Assistant Professor Wearable Informatics Lab kyle.winfree@nau.edu Dr. Eck Doerry, Professor Collaborative Computing Lab Eck.Doerry@nau.edu School of Informatics, Computing, and Cyber Systems Northern Arizona University

Project Overview:

In the United States (US), the prevalence of cardiovascular disease (CVD) among adults under age 60 is approaching 40% and nearly one-third of CVD deaths occur between 55 and 64 years. Physical activity (PA) is one of the most potent modifiable behaviors for the prevention and treatment of CVD. However, sleep deficits (e.g. short sleep duration) also exacerbate CVD risk factors. Multi-component behavioral interventions targeting changes in both PA and sleep could evoke synergistic improvements to cardiovascular health and inform novel CVD prevention strategies for mid-life adults.

Traditional intervention approaches typically hinge around personalized human-delivered health coaching, which costs both time and money and is often not covered by health insurance plans. This has led to growing popularity of wearable activity tracking monitors, such as Fitbit, which claim to offer "smart" feedback to help improve both fitness and sleep. Unfortunately, there has been very little clear scientific evaluation of these claims. To what extent can devices like Fitbit and their automated "smart" coaching really elicit meaningful, sustained health behavior change? In short, there is a critical need to develop efficient tools and techniques to carefully evaluate the efficacy of Fitbit and other "smart" platforms that claim to enhance user engagement and motivation through individually tailored behavior change techniques and feedback to improved PA and sleep behaviors.

My research focuses on measurement of physical activities and physically manifested behaviors. Key to this, is of course collection of wearable data from users in our study groups. While third party data brokers such as Fitabase do offer access to compiled activity data from Fitbits, the cost is steep: $20 per subject per week for data access. This rapidly becomes prohibitive for serious studies involving hundreds of subjects over months of observation. Fitabase also has little interest in providing near-real-time data access, much less in engaging users in any sort of community communications or motivational feedback.

The WearWare Project: a powerful platform for evaluating wearable fitness tracking

What is desperately needed is a powerful, robust, cost-effective study management/deployment platform that can support a wide variety of fitness and sleep studies driven by data from cheap common activity monitors like Fitbit. In particular, the platform would allow researchers to set up studies, select and enroll participants, register their devices, record/monitor their activity behavior, and allow configuration/deployment of a broad range of "study protocols" that monitor activity of subjects in near-real-time for certain events, which then trigger various messages to the participants and/or researchers. To make this ambitious project (which we call WearWare) manageable for Capstone teams, we have split it into three related Capstone projects, each focusing on a major functional subsystem of WearWare:

- **Wearware-DataWrangler (this project team):** This is the backend heart of the WearWare concept, aimed at providing a powerful and flexible cloud-based data collection, managment, and delivery system. It will allow registration of studies, users, and Fitbit devices, and will manage interaction with the Fitbit API to download data of all users in active studies. This includes physical activity at the minute level, heart rate at the sub-minute level, and sleep measures, among others. It will then provide a well-defined API for other modules to use to search, browse, overview, select, force a Fitbit refresh, and download targeted datasets or "slices" of datasets. A basic GUI, or Octave CLI, must be provided that connects to this API to support basic development/testing activities. The focus here is on efficient storage, searching, and extraction; a non-trivial design task for a system that collects literally millions of datapoints over the course of study.

- **Wearware-StudyWrangler:** This module is the primary interface for researchers using Wearware, allowing them to create secure accounts for their labs, develop and manage pools of potential study participants and set up and manage studies. This means configuring a study by searching/selecting/loading a set of participants, defining start-end dates, registering Fitbit devices issued to each participant … and then monitoring the study as it runs, to highlight problems with data collection as well as viewing graphical summaries of various data collected to date. A graphical data browser allows juxtaposing several data timeseries (i.e. multiple graph lines), and "zooming" in/out to different time scales. It also allows saving datasets or selected "slices" of datasets to a "study notebook," and exporting selected data slices in tabular format, e.g., for further off-system analysis. It interacts with the DataWrangler module to access datasets, and with the Interactions module to allow configurating/attaching various monitoring/event/actions protocols to studies and monitoring their behavior.

- **Wearware-Interactions:** This module supports near-real-time interactivity with study participants, allowing researchers to monitor for various events, and more importantly, to set up and deploy various "motivational protocols" they are seeking to evaluate. Specifically, researchers can create a library of "event triggers" that are essentially scanning incoming participant activity data for "events of interest." Similarly, it supports an extensible library of "Actions" (e.g., sending an email or SMS to a target). Researchers can then select from these libraries to create/edit/save "study protocols" that consist of a set of event-actions "monitors" that look for certain events and trigger the associated actions. These protocols can themselves be saved/edited in their own library; and ofcourse protocols can be search/selected/attached to various studies that have been defined. This module must provide some sort of API, allowing easy integration into the StudyWrangler module.

The entire WearWare system will be implemented in the cloud (most likely AWS) to support robustness, accessibility, and expandable compute resources. Each of the above modules above will be developed by independent Capstone teams to control complexity and avoid unnecessary development constraints. This means that teams (a) will each develop appropriate simple test harnesses to simulate the behavior of other modules and provide simple testing GUIs (for DataWrangler and Interactions; the StudyWrangler is the real GUI!); and (b) will communicate with the other teams on overall behavior and APIs to ensure that their products will integrate smoothly into a complete WearWear system moving forward.

This project, once successful, will be key to operations in the Wearable Informatics Lab. It will also position the lab for stronger collaborations with other labs, especially with relation to national funding opportunities (NIH and NSF). Finally, it will also allow WearWare to host (for a fee perhaps!) fitness/sleep studies based on wearable monitors for researchers from across the entire world; it could become the de-facto platform for doing wearables-based fitness and sleep research.

WearWare-DataWrangler module development project: Details

Having outlined the full Wearware development vision above, the following sections give specific features and development goals for the DataWrangler module of WearWare, which is the focus of this particular Capstone project. As outlined above, WW-DataWrangler provided a powerful data management core for WearWare, via an API that allows the other modules to configure and activate activity monitoring/recording for individual participants in studies; as well as an efficient mechanism for querying/extracting subsets of the data collected. The core of WW-DataWrangler will be a powerful relational database, carefully planned and optimized to allow rapid access to the massive data volumes it will need to manage. In addition, WW-DataWrangler will need to communicate efficiently

with the Fitbit API to continually query and download new activity data at specified intervals. Some key features include:

Bone Basics: Minimum viable product

- Connects to Fitbit API to download "new" data on a fixed timeframe, e.g., every ten minutes (to be determined)
- Provides a basic testing API to requests from an Octave CLI, e.g. urlread(): can see all configured devices, which ones are active, how much data is downloaded for each, participants and study id, and device status.
- Forced refresh functionality, that enables a researcher to fetch past data from Fitbit, filling in existing holes and identifying what data is missing on the Fitbit servers for any given participant.
- Can extract tabular summaries of current data available, e.g., to feed displays in other modules.
- Allows extraction of raw data associated with individual devices within a specified (start/end) time frame
- Allows extraction of all data associated with all participants in a study within a specified (start/end) time frame
- Supports a super user, with API functionality for additional research users.
- Packages above functions as an API that can be accessed by other WearWare modules.

A complete product: something actually usable

- A more robust testing http CLI response that allows richer reading of data being collected, e.g., a time-based data selectable to different timescales. Aims is to easily "see" gaps or other problems in incoming data via an Octave "query" with CSV like data response. E.g., a user could:

  ```
  >plot(urlread("http://user:password@wearableinformatics.com/study","get",{"steps","subjectX"}));
  ```

- An alerting subsystem: allows (via API of course) configuration of *generic* alerts (e.g. "Participant X has not reported any new data in N days") that can be attached to individual participants/devices. Based on inputs (e.g. triggering function) passed in, it attaches the alert to a specified data stream and fires off the configured alert message (back to a controlling module) each time the trigger fires.

Stretch goals: Cool add-ons that are not mission-critical

- A reporting and profiling subsystem for analyzing the modules performance: tracks the queries that come in, the compute/response time for the query, and other stats. Provides some GUI for overviewing recorded performance, as well as way to export performance data for further analysis.

As a side note, I prefer to meet weekly throughout the semester. This can (and maybe should) be virtually, but regular meetings are critical to project success. This isn't just a time to check in and see how things are moving along, but it's also an opportunity for us to talk through challenges and creative direction you want to consider.

Knowledge, skills, and expertise required for this project:

- Understanding of cloud-based computing; deploying/managing DB and compute instance on AWS
- Understanding of APIs and module-based software design.
- Will need to research and build advanced knowledge of DB design and optimization.
- Basic understanding of modern web2.0 web application technologies for building development GUI interface, including "get" and "post."
- Understanding of big data science: how studies work, how data is collected, large scale data management.

Equipment Requirements:

- Access to Amazon Web Service (AWS) account and products; will be provided.
- A Fitbit for each team, possibly each team member, will also be provided for dev purposes.
- No other specialized equipment should be needed, outside of standard (free) IDEs and a development station.

Software and other Deliverables:

- The WW-DataWrangler software module as described above, deployed and tested successfully with real data. Must include a complete and clear User Manual for configuring and operating the software.

- A strong as-built report detailing the design and implementation of the product in a complete, clear and professional manner. This document should provide a strong basis for future development of the product.
- Complete professionally-documented codebase, delivered both as a repository in an NAU internal Subversion or Git version control repository; and as a physical archive on a USB drive