Team DataBit

Software Testing Plan

March 26, 2021

Team Members: Andrea Caviglia, Cheyenne Clutter, Samantha Rodriguez, Jensen Roe, Steven Sprouls
Sponsor: Dr. Kyle N. Winfree
Mentor: Dr. Eck Doerry

Table of Contents

1 Introduction

Cardiovascular diseases (CVDs) are the leading cause of death globally. In the United States alone, nearly half of adults over 20 years of age have some form of cardiovascular disease, and approximately 655,000 people die from heart disease each year – around one in every four deaths nationally. Moreover, CVDs pose significant concerns in light of the coronavirus pandemic, as those who suffer from cardiovascular disease are at an increased risk of severe illness and death from COVID-19. Lifestyle changes, such as engaging in physical activity and getting enough sleep, are often the best way to prevent and treat cardiovascular diseases, so being able to monitor such behaviors in some way is an important part of addressing the problem at large.

More research is needed to better understand the relationship between lifestyle changes and their impact on disease risk/outcomes. Typically, researchers enroll hundreds of participants in a study and observe them over a certain period of time. The nature of these observations vary, but in the case of health and fitness studies, researchers are often interested in continuous monitoring and recording of data such as heart rate, activity levels, and so on. This data is collected and recorded over weeks, months, or even years, and is analyzed for patterns that support a given hypothesis.

To enable and support such large-scale, long-term studies, researchers require a way to efficiently maintain and collect exertion and movement data from large numbers of patients. One approach is to employ wearable activity tracking monitors that can analyze physical activity and sleep behaviors throughout the day. For example, devices such as Actigraphs™ are used in clinical settings to capture and record physical activity. These are small wearable devices that go on a person's wrist, much like a watch, and that monitor rest and activity cycles. However, Actigraph devices are expensive (around $250 per device), difficult to use, and incapable of storing much data over long periods of time, which makes them ill-suited for the purposes of long-term study with a large number of participants.

Our project sponsor, Dr. Kyle Winfree, is a researcher who is interested in the utility and accessibility of Fitbits: a cheap and highly usable consumer version of an activity monitor as a potential alternative to Actigraphs in exercise and movement studies. As the leader of the Wearable Informatics Lab at Northern Arizona University – whose primary interest is in utilizing wearable technologies to measure and improve health care – one of his current concerns is the collection of wearable data from users. While Fitbit devices themselves provide a relatively inexpensive solution to data collection, it is not immediately obvious that they could be useful for large-scale studies. In particular, a major obstacle is collecting and accessing data collected by Fitbits, i.e. retrieving heart rate and other data from Fitbits

given to hundreds of study participants in near-real-time and making it accessible to researchers.

[redacted] These issues have prompted Dr. Winfree to devise the WearWare Project – an informatics platform for evaluating wearable fitness tracking. This would allow researchers to create and manage studies, enroll participants, and access their Fitbit data for further analysis. A first prototype was developed and proved the basic concept, but suffered from poor performance and an inability to handle the sheer amount of data.

Our envisioned solution is to develop a new system from the ground up to create a powerful data management cornerstone that we call DataWrangler as the backend of the WearWare concept. This will involve creating a powerful and flexible data collection, management, and delivery system that allows for the registration of studies, participants, and Fitbit devices, and communicates with the Fitbit API in order to download data in real-time. We will also be providing an API for future modules of the project in order to perform various operations on the data, as well as a basic development client that connects to this API for testing. In doing so, we will be providing Dr. Winfree and his lab – as well as potentially other researchers in the future – with an innovative and affordable product for use in various health studies.

In this document, we will outline our software testing plan to ensure that our implementation meets the client's requirements. Namely, we will go over the unit tests that we will be using to verify functional characteristics, integration tests for inter-module interaction, and usability testing to confirm that our system is intuitive and effective for end users. In doing so, we can guarantee a cohesive, working product that meets our client's expectations for study management functionality as well as optimized performance.

2 Unit Testing

Unit tests are fundamental in a team's application testing strategy. By creating and running unit tests against our code, we can easily verify that the logic of individual units (the smallest pieces of functional code) is correct in isolation. These tests can also help the team to quickly catch and fix software regressions introduced by changing code, and to generally exercise the functionality of units of code in an easily repeatable way.

The small scope and efficient nature of unit testing will be beneficial to the team as our development time is very limited. Fortunately, there is a testing framework for Django called PyTest. PyTest will help us to streamline our unit testing process and help us gather metrics via its PyTest-cov plugin, which is used to test the coverage of a test by determining which lines of codes are actually being run. In this section we will outline tests for the following units:

- Study management functions
    - Creating a study
    - Deleting a study
    - Editing a study
    - Creating a participant and enrolling the participant into a study via email
    - Querying / Collecting data from a study
- User management functions
    - Creating a new user
    - Logging in
    - Resetting a user's password
    - Permissions

2.1 Study Management

Given that the WearWare system is ultimately a study management platform, it is imperative that we test study-related functionality. This involves being able to create, delete, and edit certain information relating to a specific study that the user has access to. Additionally, the user should be able to manage participants in a study, as well as to retrieve the data collected by those participants' Fitbit devices.

2.1.1 Create Study

To test this, we will run a test that will create a new study with a unique name. If this passes, we will know this function works. To make sure that a study is not created if its name is not unique, we will try creating another study with the same name as the last one. If this test fails, it can be considered working as intended.

2.1.2 Delete Study

This function's purpose is to delete a study. We will run a test that will attempt to delete the study we created for the previous test. If it passes, we will know that this works. Following this, we will also create another study, but instead of leaving it empty we will fill it with fake participants and mark it as active. When trying to delete this study, it should not allow us to because it is considered an ongoing study.

2.1.3 Edit Study

Users of WearWare should be allowed to edit certain information about a study, such as its name and description. To test this functionality, we will use a dummy study and attempt to change its name and comment fields. We should receive a passing result for this test, unless in the case of changing the name to something that is not unique.

2.1.4 Manage Participants

Researchers using the WearWare web app should be able to create a user object, which has a Fitbit account. To test this functionality, we will navigate inside of a study through the GUI and create a participant object using one of our emails. After doing so, we will check that we received the email from WearWare telling us we were invited to participate in a study that includes a link to register a Fitbit device. We will register the device, and then check to be sure that the refresh and authentication tokens were correctly parsed and stored in the correct spot in the database.

2.1.5 Query Data

The purpose of the WearWare system is to collect data from Fitbit devices and store it so it can be used for analysis. To this end, researchers need to be able to extract slices of data:

- All participants in a given study
- All participants' Fitbit data for a given study
- A single participant's Fitbit data for a given study
- Participants' Fitbit data within a given date range

To test this, we will fill a study with fake data, i.e fake participants who have fake activity levels, heart rates, and sleep data. We will then write out test functions that pass SQL queries in order to return a given set of data. If we are correctly able to retrieve these complex sets of data, we will know that this function works and we can consider the test as passed.

2.2 User Management

WearWare also requires some form of user management system to ensure that not only can researchers log into the system, but that users cannot access or modify data they are not authorized for. Given that we are storing the personal information of participants, and that multiple researchers and research assistants may have access to studies, these tests are critical for ensuring that WearWare has adequate security.

2.2.1 User Creation

Only the admin user of the WearWare system (who, after project deployment, will be our client) has the ability to create a new user. To test if a new user can successfully be created, we will log into the admin panel using the superuser credentials we have made, go to the "users" section, and then create a new user using an email using the name and email address of one of the team members. We will then check the list of users, and will consider this functionality to work if the new user shows up in the list.

2.2.2 Login

To verify that the login is working, we will write a test that uses the username and password of a test user we previously created to attempt to gain access to the system. This function will be considered working if the user is able to log in to the main WearWare page.

2.2.3 Password Reset

A password reset feature is an important component of any web application, and WearWare is no different in this regard. If a user forgets their password, they need to be able to reset it themselves so they can access their studies; otherwise, they would have to notify the administrator to reset it for them in the admin console, which is highly inefficient. To verify that this function works, we will navigate to the "forgot password" link on the login page, then enter the email address associated with a given test account. We will then check to see if we have received the email with the link to reset the password., navigate to the reset password page and enter a new password. We will consider this test passed if we can then log in using the new password for that user.

2.2.5 Permissions

For any given study, it is expected that multiple users will have access to the data within that study, with varying degrees of administrative privileges. Namely, these are defined in the WearWare system as:

- Admin (SI)
- Researcher (PI)
- Research Assistant (GRA, RA)

Users should have different permissions depending on which category they fall into. For instance, a research assistant may not be allowed to enroll participants, while a researcher can. To test this functionality, we will create users of each type, and attempt to perform various actions – viewing data, editing studies, and so on – for each user. In some cases, we would expect to see the completion of the action, or if the user is not authorized to perform that task, then we would anticipate a warning message.

3 Integration Testing

After unit testing is performed to determine if the individual components perform as intended, integration testing is used to test the interactions between the units. In other words, it tests whether different parts of the system pass the necessary information correctly between each other. This is of the utmost importance in the WearWare system, as we need to maintain accurate data that can be used for research purposes.

For the purposes of integration testing, we will need to test the inter-functionality of the Fitbit API, the WearWare API, the database, and the task management module. In particular, we need to look at the interactions between:

- The Fitbit API and the database
- The database and the WearWare API
- The Fitbit API and the task management module

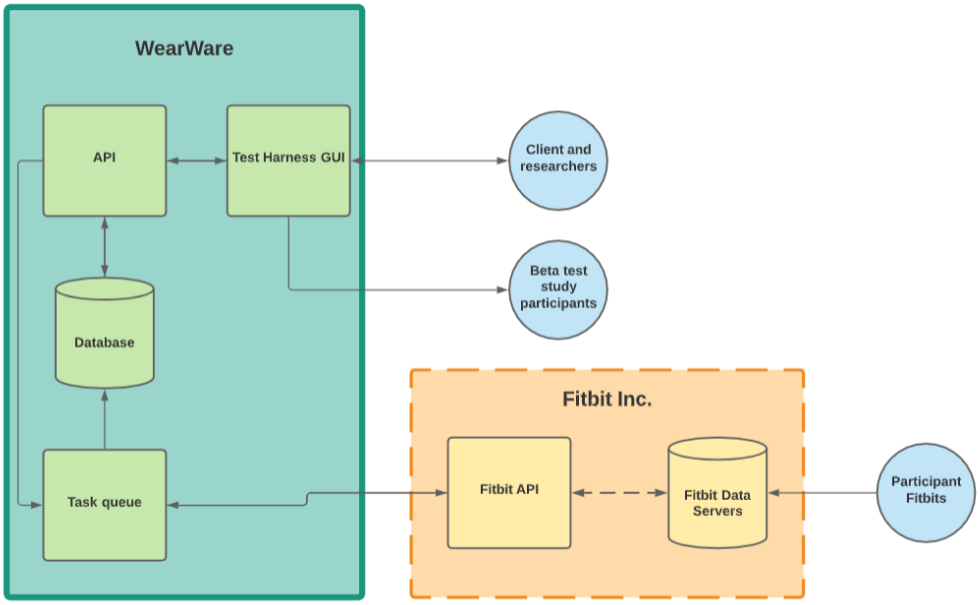

***Figure 1**: Modules in the WearWare system architecture*

3.1 Fitbit API to Database

The data passage from the Fitbit API to WearWare's database is arguably the most integral part of the system. Data needs to effectively be gathered from the Fitbit API via a participant's Fitbit device's authentication tokens and stored on WearWare's database. This

will be tested by registering one of our team's own Fitbit devices with the app and wearing it for a few days to collect a meaningful amount of data. After wearing the device for a few days, we will then check to make sure the database has stored the collected data in the correct tables and columns directly via pgAdmin, a database management system designed to be used with PostgreSQL databases.

3.2 WearWare API to Database

The transmission of data between the WearWare API and the database is also extremely important to the core functionality of the DataWrangler component. We need to be sure that the WearWare API can correctly post new records to the database and that it is able to correctly request information from the database. To test that information can be posted to the database, we will post a new study to the database using the app. If we can then see that the new information is stored in the database through the use of pgAdmin again, we will know this works. We need to check that records can be properly edited as well; to do this, we will navigate to the study we just created in the last test, change the name and description of the study through the use of a patch request, then check if the changes were reflected in the database.

3.3 Fitbit API and Task Manager

The last integration test we will be conducting deals with the communication between the task manager module and the Fitbit API. Part of the information we will be requesting from the Fitbit API is the sync records of each device. One of the functions required of WearWare's task management system (Celery) is to send an email requesting action to a study participant whose device has not synced within a six hour time period. Other functions of the task manager include requesting new authentication and refresh tokens for each device from Fitbit every eight hours. To be able to test both of these tasks, there must be another task that is set to make WearWare request information from participant's device every so often.

We will test these functionalities by registering one of our Fitbit devices and making sure data is continuously collected for over 8 hours. We will also register another device at the same time, but only allow it to sync a single time for the entire 8 hour time period. If both of these tests are able to pass, we will know the data can be sent between Fitbit and the task manager successfully and correctly.

4 Usability Testing

Once we can verify that the system is working as intended through unit and integration testing, we need to conduct usability testing to confirm that our product is serviceable for

end users. Traditional usability testing has real people interacting with an application. There are variations on how they are observed and if/how the moderator asks questions, but its general goal is to determine if the application is usable for a general audience, meaning that the user can navigate the application effectively.

Given that we are developing a study management platform for researchers, we have decided to test our application with users who have such specialization, rather than random individuals who may not understand the end-user flows. In particular, our target audience would be our client, Dr. Kyle Winfree, and if that goes smoothly, we may also like for some of his colleagues to test the system. If feedback from the client is not overwhelmingly positive, he may not want his research colleagues to use it.

To conduct our process for user testing, we will schedule a Zoom meeting with Dr. Winfree. We will ask him to walk through the major use cases of creating a study, enrolling a participant and registering their Fitbit device, then querying the study for data. We will also record this Zoom call in order to take notes on which parts of the application the user is having difficulty navigating. It should not take more than 15 minutes for the user to figure out how to go through this use case. If it does, we will know that the application is not sufficiently intuitive to use.

In addition to end-user flow, this will also allow us to verify that our performance requirements are up to par. Ideally, our client should be able to make both simple and complex queries within a single testing session; if our client finds that accessing the data takes too long, and that he is not willing to wait that long, then we may need to make further adjustments to our database.

We will give ourselves approximately one week to conduct user testing. This will ensure we have the time required to schedule a meeting with Dr. Winfree, record his feedback, and make any changes necessary to improve the user experience. If successful, this testing will verify the intended end user experience and will hopefully lead to the creation of a product that suits the needs of the client.

5 Conclusion

Given the prevalence of CVDs, which kill 655,000 people in the US each year, it is important for researchers to be able to conduct large scale health studies that can gauge the impact of physical activity and sleep on disease risk and outcomes. The WearWare system is intended

to address this concern, by allowing researchers to utilize inexpensive wearable devices, namely Fitbits, and collect information such as:

- Heart rate, which can be tracked on a one-second interval.
- Activity level, which gives a range of a brisk walk to a full exercise routine.
- Sleep data, which provides in-depth analysis such as restless sleep, R.E.M. cycles, and light sleep.

An existing prototype of the WearWare system proves the underlying concept of feasibility, but has significant performance and feature limitations. The aim of this project is to build a solid, high performance database module to serve as the data management core of a future redesigned WearWare system.

In this document, we discussed our testing plan for verifying that our system meets these requirements. We have identified the major use cases in study and user management, and outlined our plan to confirm that we have correct functionality at both the unit- and module-level. Additionally, we intend to conduct usability testing with our client to ensure that the WearWare system is intuitive as well as efficient enough to be deployed in a research setting. We are confident that this will allow us to demonstrate that we have a working and highly effective product.