6-9-2019

H.E.A.R.T.

Jacob Day

Yutong Li

Samuel Rietz

Brendan Watamura

SANTA CLARA UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING

Date: June 9, 2019

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

Jacob Day
Yutong Li
Samuel Rietz
Brendan Watamura

ENTITLED

H.E.A.R.T.

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

Thesis Advisor

Department Chair

H.E.A.R.T.

by

Jacob Day
Yutong Li
Samuel Rietz
Brendan Watamura

Submitted in partial fulfillment of the requirements
for the degree of
Bachelor of Science in Computer Science and Engineering
School of Engineering
Santa Clara University

Santa Clara, California
June 9, 2019

H.E.A.R.T.

Jacob Day
Yutong Li
Samuel Rietz
Brendan Watamura

Department of Computer Engineering
Santa Clara University
June 9, 2019

ABSTRACT

Healthy parenting and family resilience in early childhood has been shown to be an important factor in building emotional resilience for the children: it illustrates that when parents have higher emotional resilience, their children tend to have higher emotional resilience as well. However, the tools that available in the market right now only teach people what emotional resilience rather than how to practice it in daily life.

This report describes our project to create a virtual reality tool that can not only teach the importance of emotional resilience, but also help the parents develop personal resilience. The system is based on the VR Empathy Training Tool created by a former senior design project in which the user can interact with a crying child and learn how to handle stress under certain circumstances. The new system will add new features so that it can inform users about their stress level and allow the users to track their progress.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Statement

Parents, specifically those of at-risk youth, are not providing their children with responsive caregiving. These children are growing up in an environment in which their parents have low emotional resilience, a quality developed early on in one's childhood. When parents display this behavior in front of their children, these children grow up susceptible to stress and emotional distress, perpetuating a vicious cycle. Healthy parenting and family resilience in early childhood has been shown to be an important factor in effectively managing stress, promoting school readiness and achievement, and preventing adolescents from participating in high-risk behaviors.

At this time, there lacks a means to teach emotional resilience to reach a wide target audience. Traditional solutions such as psychotherapy or medication prescribed by a doctor may ameliorate the symptoms of stress, but fail to address the underlying issue. Even with the emergence of electronic wearable devices (i.e. stress detectors), this solution alone fails to help as they simply alert individuals if they are stressed, potentially worsening symptoms with constant reminders. Dr. Barbara Burns, professor and director of child studies at Santa Clara University (SCU), presents the most promising solution at this time; her Resilient Families Program (RFP) aims to promote family and community resilience with community-led, science-based parent education programs. However, the program's outreach is severely limited to the number of individuals Dr. Burns can train at a given time.

1.2 Related Work

A former senior design project collaborated with Dr. Burns and came up with a VR Empathy Training Tool. The user can interact with a crying child and therefore learn how to handle stress (under certain circumstances). However, the project has the same level of difficulty for all individuals regardless of the difference of their emotional resilience. The project also lacks a sense of immersion and virtual "presence", which is a key component in creating empathy within VR. Furthermore, their design decision to measure stress using only heart rate is non-comprehensive.

Fortunately, resilience is not a fixed characteristic; it can be learned. As individuals, we can build an awareness of

the situations in which we are least resilient and focus our efforts on developing personal resilience there.

1.3 Objectives

Our solution builds upon the framework laid out by RFP. We propose a virtual reality (VR) mobile application that allows parents to interact with a virtual world objects, including a virtual child, in a time-based experience. The child will display signs of distress and will react based upon the user's decision. Unlike prior solutions, ours does more than inform users they are stressed. The intensity of the experience will vary based on perceived stress levels. This teaches users how to handle stress in a productive manner without overwhelming them from the start. To address the lack of outreach of RFP, our solution will utilize an accessible and low cost VR implementation. Our solution will also provide users with the ability to track their progress, based on stress levels exhibited during each session. As a result, different individuals can get personalized training.

Chapter 2

Requirements

2.1 Functional Requirements

2.1.1 Critical

- The system will train users in emotional resilience.
- The system will keep track of users' personal progress.
- The system will collect user data anonymously (including personal progress).
- The system will measure user heart rate using the phone's camera lens.

2.1.2 Recommended

- The VR experience will contain spatialized audio.
- The VR experience will halt if the user is too uncomfortable.
- Users' data will be sent for research, with their consent.

2.1.3 Suggested

- The system will function on Android devices.

2.2 Non-Functional Requirements

2.2.1 Critical

- The application will be user-friendly.
- The VR experience will be immersive.
- The system will take into consideration the safety of its users.

2.2.2 Recommended

- The system will be supportive of multiple languages.
- The application will be energy efficient.

2.2.3 Suggested

- The system will be maintainable and portable.

2.3 Design Constraints

- Low-cost and accessible
- Operate on iOS devices using Google Cardboard
- Experience lasts a maximum of 2 minutes
- All functionality contained within user's phone

Chapter 3

Use Cases

The sole actors involved are users. Their functions are listed in Figure 3.1.

3.1 User Use Cases

The users can play the game from their own perspective, which will be interacting with a virtual child and other virtual world objects. The users can also keep track of their personal progress by checking out the report generated.

Table 3.1: Play as Parent Use Case

Use Case Name	Play as parent
Goal	Interact with virtual world objects
Actor(s)	User
Precondition(s)	Complete baseline calibration
Postcondition(s)	All tasks completed in home
Exception(s)	N/A

Table 3.2: Track Personal Progress Use Case

Use Case Name	Track personal progress
Goal	View and learn from progress
Actor(s)	User
Precondition(s)	Complete at least one session
Postcondition(s)	Display historical data
Exception(s)	N/A

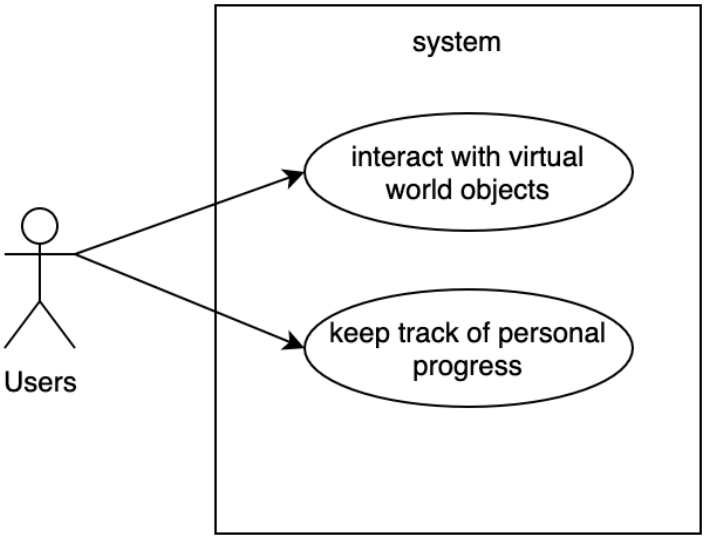

Figure 3.1: Use Cases

Chapter 4

Activity Diagram

The activity diagram shown in Figure 4.1 describe the flow of actions when users access the application. The potential actions are based on the use cases described in Figure 3.1.

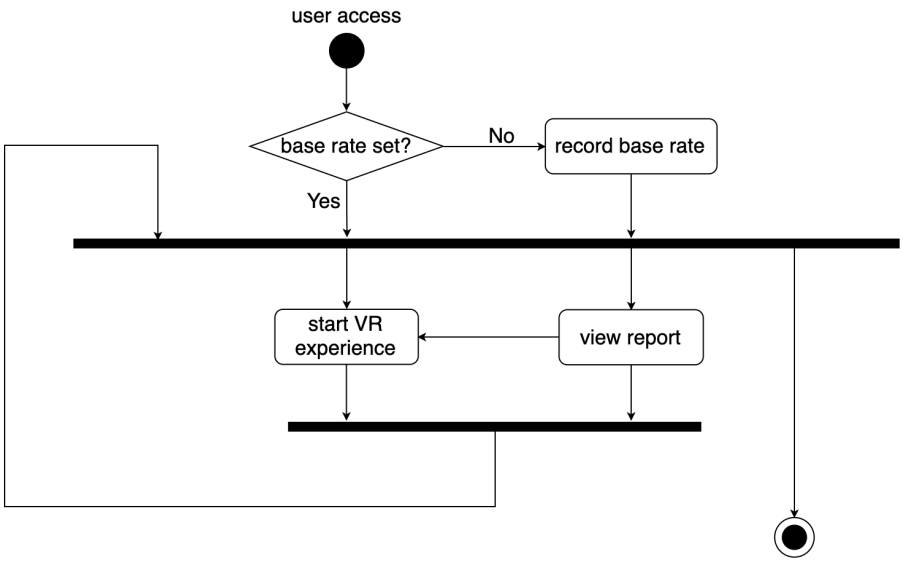

Figure 4.1: Users Activity Diagram

Chapter 5

Technologies Used

Several of the technologies we chose are based off their relevant functionality as well as any previous knowledge members had with these technologies. This allowed us to move forward quicker and more efficiently with our project. Our software technologies include C#, Objective-C, MySQL, PHP, Google VR SDK for Unity, ResearchKit, Unity Engine, and Xcode. We also used Google Cardboard for our hardware technology. Finally, GitHub was used as our version control system to store all our code.

Table 5.1: Technologies Used

Technology	**Purpose**
C#	Scripting
Objective-C	iOS Development
MySQL	Database
PHP	Query MySQL Data
GitHub	Version Control
Google Cardboard	VR Platform
Google VR SDK for Unity	VR Features
ResearchKit	Heart Rate Monitor
Unity Engine	Game Engine
Xcode	IDE

Chapter 6

Architectural Diagram

We opted for a component-based architecture for the virtual reality experience, as shown in Figure 6.1. All system processes are placed into separate components such that all of the data and functions inside each component are semantically related. Within the experience, the user will be given four tasks in total—one task per 20 seconds, and the user has two minutes to finish them. For example, when the second task is given to the user, and the user has not finished the first task yet, the user will do both task at the same time until the third task is given. The difficulty can be adjusted by shortening or extending the time frame.

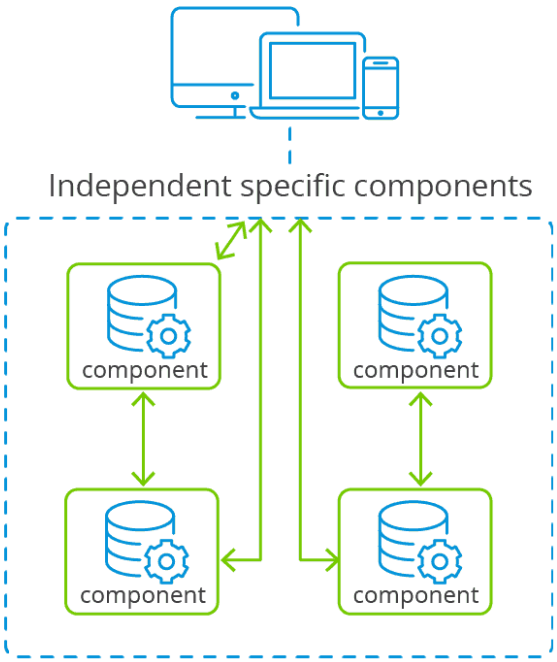

Figure 6.1: Component-Based Architecture

Figure 6.2 depicts the data-centric architecture used for the report functionality. A client, the user's smartphone,

sends a PHP request to the web server. The web server queries the database to retrieve the user's data. It then sends the data back to the client via PHP, where it is displayed within the user's application.

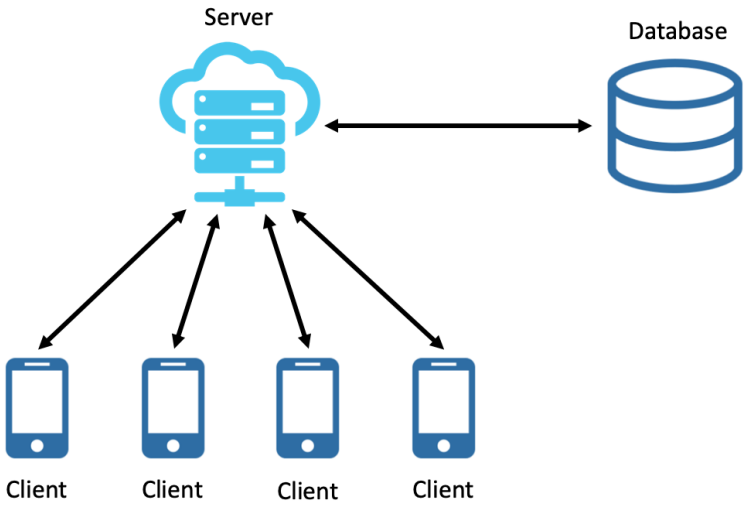

Figure 6.2: Data-Centric Architecture

Chapter 7

Design Rationale

7.1 Biofeedback

Biofeedback, a training technique that monitors bodily functions, teaches users to increase awareness of their physiological functions. This technique increases the user's feeling of virtual "presence." ResearchKit is an open source software framework that simplifies application creation for medical research or for other research projects. It will allow us to track the user's heart rate natively in iOS.

7.2 C#

We chose C# as our main scripting language as Unity Engine offers a primary scripting API in C#. Also, our group's familiarity with C and C++ will make learning this new technology more efficient.

7.3 Objective-C

We chose Objective-C as our main language for iOS development due to its similarity to other C-based languages. We opted for this over Swift due to compatibility issues that arose between Swift and third-party frameworks.

7.4 MySQL

We chose MySQL as our relational database system because of our previous experience in other projects. This eliminated time needed to setup and familiarize ourselves with a new database system.

7.5 PHP

Like with MySQL, we chose PHP as our bridge between our database and our mobile application because of our previous experience in other projects. PHP also integrated easily within our Objective-C code.

7.6 Component-Based Architecture

By utilizing an component-based architecture for the virtual reality experience, we will promote the separation of concerns in regard to the system's wide-ranging functionality. This architecture is suitable for the nature of our design, which encompasses a loosely coupled structure.

7.7 Data-Centric Architecture

We decided to use a data-centric architecture for the report functionality as it allows for a single point of access for user's information.

7.8 GitHub

We chose GitHub as our version control system due to our previous knowledge regarding its functionality. It served as the platform we use to collaborate and store code and core assets.

7.9 Google Cardboard

Google Cardboard technology was chosen due to the low cost of usage. It provides a more cost-effective solution to virtual reality hardware than many of the major competitors, while still giving a quality experience. By offloading the software requirements to a smartphone and keeping the hardware down to a cardboard frame, Google Cardboard lets the product scale more effectively among users and increases accessibility.

7.10 Google VR SDK for Unity

Google VR provides SDKs for many popular development environments such as Android, iOS, Unity, and Unreal. These SDKs provide native APIs for key VR features like user input, controller support, and rendering. These can be used to build new VR experiences on Google Cardboard.

7.11 Unity Engine

Unity Engine was our cross-platform game engine driving our application. It is the most widely-used VR development platform. Unity's highly optimized rendering pipeline will expedite our development process. Additionally, Unity supports the low-level APIs Metal on iOS and Vulkan on Android. Its build settings allow applications to be built for iOS as an Xcode project.

7.12 Virtual Reality

We chose a virtual reality interface due to the immersive experience it provides. Users of the system are practicing for real-life situations. In order to get the most out of the experience, the system must be as close to real life as possible. Virtual reality provides the closest experience to reality, without any keyboard controls to interface through. Moving through a virtual environment provides valuable experience for reacting to the real-life situations this program hopes to train people in.

7.13 Xcode

Xcode functioned as our IDE. It includes a suite of development tools for creating iOS applications. Unity's support for building to Xcode will make our development process seamless.

Chapter 8

Test Plan

Testing is fundamental for ensuring that our product works effectively for its users. Throughout this project, we utilized testing to verify our product works and validate that we were building the right product.

8.1 White Box Testing

As we developed this project, we constantly tested and debugged our solution. Since we used GitHub for reasons mentioned in Section 5, this simplified deploying our system and verifying if changes are functioning correctly.

8.1.1 Unit Testing

We divided our functionality into modules to implement unit testing. We created a testing directory that includes test cases for each of these respective modules.

8.1.2 Integration Testing

After unit testing was complete, we incorporated multiple modules to test together. We tested to verify our navigation functionality, allowing for a more seamless experience for black box testers.

Chapter 9

Societal Issues

9.1 Ethical

This project utilizes virtual reality to promote empathy in users with their children by simulating real life situations. Ethical considerations were made in regard to security and exposing users to stress. We opted to prioritize user data confidentiality by proactively preventing outside attacks to our database. Users are also allowed to decline the option of sending data for research. Our project also refrained from deliberately inducing stress in our users.

9.2 Social

This project was designed to act as a supplement to Dr. Burns' Resilient Families Program. It will allow parents to train in emotional resilience, even when they are unable to physically attend these workshops.

9.3 Economic

One of the main design constraints for this project was low-cost and accessibility. By utilizing Google Cardboard, and assuming the user already possesses a smartphone, the project can be made for less than $15.

9.4 Usability

Given the constraints of virtual reality headsets, our project utilizes only one button. The rest of navigation within the application relies on the user "looking" at the object they would like to interact with in the virtual world. Within the iOS application itself, we simplified the interface to only allow for two options: partake in the experience or view a report of your previous sessions.

9.5 Lifelong Learning

Most of our team was unfamiliar with the technologies and concepts we would be using in this project. While we studied new material to familiarize ourselves with the new technologies we would be using, we consulted our advisor, Dr. Amer, for assistance in the guidance and direction of our project.

9.6 Compassion

With the addition of our project, the Resilient Families Program will now allow parents to train off-site and independently. Parents with busy schedules can now practice on their own time, with little to no impact on their day-to-day lives. Our project helps this program by extending the outreach of parental empathy training, thus raising more awareness toward the importance of mental health of our youth.

Chapter 10

Conclusion

10.1 Evidence

We created an accessible parental empathy training tool using Google Cardboard and a smartphone. Using only the phone's camera, we can record the user's estimated heart rate in real time. This collected data can be used to track progress from previous sessions. Based on audience feedback during our presentation, we concluded that our virtual reality experience may need to guide our users more explicitly in order to create a more seamless experience. However, after informing the audience on why certain decisions are made in the experience, they felt more comfortable in understanding the bigger picture. Because of this, our experience has the potential to successfully function as a supplement to the Resilient Families Program.

10.2 Lessons Learned

During the design process, our team took note of several important lessons that groups or individuals seeking to replicate or continue this project would find beneficial. We found that testing early and often helped us reduce the number of defects and avoid spending too much time reworking previous modules. In addition to this, we acknowledged that our communication with end-users could be improved. This would allow us as developers to gain a better understanding what aspects about the application could be modified to better suit the needs of our users.

10.3 Advantages

Within our project, we were able to successfully integrate our database within the iOS application. This reduced the complexity of our system by allowing for a single point of access to all functionality. Included in this is the implementation of the integrated heart rate sensor, which also reduces the hardware complexity and overall cost.

10.4 Disadvantages

At this time, our solution lacks a smooth transition between the virtual reality experience and the report functionality. Some users may find it inconvenient to navigate between the two interfaces.

10.5 Future Works

10.5.1 System Testing

We can enlist testers to test the integrated system to evaluate the systems compliance with our specified requirements. They can test our mobile application on different hardware to ensure portability. However, prior permission is needed from the SCU Institutional Review Board to conduct research with human participants.

10.5.2 Acceptance Testing

Final end-user acceptance testing can be performed with Dr. Burns and parents in the Resilient Families Program to ensure our implementation meets their criteria.

10.5.3 Auto Layout

By implementing Apple's constraint-based layout system within our iOS app, we can support devices with varying screen sizes. At this time, we only had access to physically test on the devices our team members owned.

10.5.4 Multiple Users

Our current implementation differentiates users based on the phone's UUID. In the future, multiple users could be support on a single device using Apple's LocalAuthentication API.

Chapter 11

References

[1] ”Resilient Families Program — At Santa Clara University.” Santa Clara University. May 2019. [Online]. Available: https://resilientfamiliesscu.wordpress.com/

[2] Barbara Burns, Juanita Escamilla and Marlen Monroy. ”Resilient families - Ignation Center - Santa Clara University.” Santa Clara University. May 2019. [Online]. Available: https://www.scu.edu/ic/programs/thriving-neighbors/current-programs/resilient-families/

Chapter 12

Appendices

12.1 iOS Application Source Code

Listing 12.1: ViewController.h

```
1  //
2  //  ViewController.h
3  //  mainScreen
4  //
5  //  Created by Jake Day on 5/5/19.
6  //  Copyright  2019 Jake Day. All rights reserved.
7  //
8
9  #import <UIKit/UIKit.h>
10 #import <ResearchKit/ResearchKit.h>
11 #import <ResearchKit/ORKHrCaptureStepViewController.h>
12 #import <ResearchKit/ORKHrCaptureStep.h>
13 #import <ResearchKit/ORKHrCaptureView.h>
14 #import <ResearchKit/ORKFFTUtils.h>
15 #import "PHPController.h"
16
17 @interface ViewController : UIViewController <ORKStepViewControllerDelegate>
18
19 @end
```

Listing 12.2: ViewController.m

```
1  //
2  //  ViewController.m
3  //  mainScreen
4  //
5  //  Created by Jake Day on 5/5/19.
6  //  Copyright    2019 Jake Day. All rights reserved.
7  //
8
9  #import "ViewController.h"
10
11 @interface ViewController ()
12
13 @property (weak, nonatomic) IBOutlet UIButton *gameButton;
14 @property (weak, nonatomic) IBOutlet UIButton *reportButton;
15 @property (strong, nonatomic) ORKHrCaptureStepViewController *hrViewController
       ;
16 - (IBAction)backButton:(id)sender;
17
18 @end
19
20 @implementation ViewController
21
22 - (void)viewDidLoad {
23     [super viewDidLoad];
24     //[self configPHP];
25 }
26
27 - (void)configPHP {
28     PHPController *php = [[PHPController alloc] init];
29     BOOL isCalibrated = [php calibrationSet];
30
31     if (!isCalibrated) {
```

```
32          NSLog(@"Not calibrated...initializing HR capture");
33          // present ORKHrCaptureStepViewController
34          ORKHrCaptureStep *myStep = [[ORKHrCaptureStep alloc]
                    initWithIdentifier:@"hr"];
35          [myStep setDuration:@120];
36
37          _hrViewController = [[ORKHrCaptureStepViewController alloc]
                    initWithStep:myStep];
38          _hrViewController.delegate = self;
39          ORKHrCaptureView *hrCaptureView = [_hrViewController getCaptureView];
40          UIBarButtonItem *skipButton = [[UIBarButtonItem alloc] initWithTitle:@
                    "CAPTURE_BUTTON_SKIP" style:UIBarButtonItemStylePlain target:
                    self action:@selector(goToHomeScreen)];
41
42          [hrCaptureView setSkipButtonItem:skipButton];
43          dispatch_async(dispatch_get_main_queue(), ^{
44              [self presentViewController:_hrViewController animated:YES
                        completion:nil];
45          });
46          // acquire hr
47          NSMutableArray *hrData = [_hrViewController getCollectedHrValues];
48          NSLog(@"Array: %@", hrData);
49          // update calibration column in UUID's row
50      } else {
51          NSLog(@"User calibrated");
52      }
53  }
54
55  - (void)goToHomeScreen {
56      NSLog(@"pop");
57      [[self presentingViewController] dismissViewControllerAnimated:NO
                completion:nil];
```

```
58      //[_hrViewController dismissViewControllerAnimated:NO completion:nil];
59  }
60
61  -(IBAction)toggleUIButtonImage:(id)sender{
62      if ([sender isSelected]) {
63          [sender setBackgroundColor:[UIColor orangeColor]];
64          [sender setSelected:NO];
65      } else {
66          [sender setBackgroundColor:[UIColor blueColor]];
67          [sender setSelected:YES];
68      }
69  }
70
71  - (IBAction)backButton:(id)sender {
72
73  }
74
75  - (void)capturePressed:(void (^ _Nullable)())handler {
76
77  }
78
79  - (void)retakePressed:(void (^ _Nullable)())handler {
80
81  }
82
83  - (void)setHrLbl:(nonnull UILabel *)hrLbl {
84
85  }
86
87  - (void)stopCapturePressed:(void (^ _Nullable)())handler {
88
89  }
```

```
90
91  - (void)stopHrEstimation {
92
93  }
94
95  - (void)videoOrientationDidChange:(AVCaptureVideoOrientation)videoOrientation
        {
96
97  }
98
99  - (void)encodeWithCoder:(nonnull NSCoder *)aCoder {
100
101 }
102
103 - (void)traitCollectionDidChange:(nullable UITraitCollection *)
        previousTraitCollection {
104
105 }
106
107 - (void)preferredContentSizeDidChangeForChildContentContainer:(nonnull id<
        UIContentContainer>)container {
108
109 }
110
111 - (CGSize)sizeForChildContentContainer:(nonnull id<UIContentContainer>)
        container withParentContainerSize:(CGSize)parentSize {
112     return CGSizeZero;
113 }
114
115 - (void)systemLayoutFittingSizeDidChangeForChildContentContainer:(nonnull id<
        UIContentContainer>)container {
116
```

```
117 }
118
119 - (void)viewWillTransitionToSize:(CGSize)size withTransitionCoordinator:(
        nonnull id<UIViewControllerTransitionCoordinator>)coordinator {
120
121 }
122
123 - (void)willTransitionToTraitCollection:(nonnull UITraitCollection *)
        newCollection withTransitionCoordinator:(nonnull id<
        UIViewControllerTransitionCoordinator>)coordinator {
124
125 }
126
127 - (void)didUpdateFocusInContext:(nonnull UIFocusUpdateContext *)context
        withAnimationCoordinator:(nonnull UIFocusAnimationCoordinator *)
        coordinator {
128
129 }
130
131 - (void)setNeedsFocusUpdate {
132
133 }
134
135 - (BOOL)shouldUpdateFocusInContext:(nonnull UIFocusUpdateContext *)context {
136     return NO;
137 }
138
139 - (void)updateFocusIfNeeded {
140
141 }
142
143 - (void)stepViewController:(nonnull ORKStepViewController *)stepViewController
```

```
        didFinishWithNavigationDirection:(
        ORKStepViewControllerNavigationDirection)direction {
144
145 }
146
147 - (void)stepViewController:(nonnull ORKStepViewController *)stepViewController
        recorder:(nonnull ORKRecorder *)recorder didFailWithError:(nonnull
        NSError *)error {
148
149 }
150
151 - (void)stepViewControllerDidFail:(nonnull ORKStepViewController *)
        stepViewController withError:(nullable NSError *)error {
152
153 }
154
155 - (void)stepViewControllerResultDidChange:(nonnull ORKStepViewController *)
        stepViewController {
156
157 }
158
159 @end
```

Listing 12.3: PHPController.h

```
1 //
2 //  PHPController.h
3 //  mainScreen
4 //
5 //  Created by Jake Day on 5/5/19.
6 //  Copyright  2019 Jake Day. All rights reserved.
7 //
8
9 #import "ASIFormDataRequest.h"
```

```
10 #import "PHPControllerDelegate.h"
11
12 @interface PHPController : NSObject <ASIHTTPRequestDelegate>
13
14 @property (nonatomic, strong) NSString *uuid;
15 @property (nonatomic, strong) NSString *hr; // remove
16 @property (nonatomic) BOOL calibrated;
17
18 enum SESSIONS {
19     SESSION1,
20     SESSION2,
21     SESSION3,
22     SESSION4,
23     SESSION5
24 };
25
26 -(id)init;
27 -(void)startCalibration;
28 -(BOOL)calibrationSet;
29 -(NSString *)loadUUID;
30 -(NSString *)loadHR;     // remove
31 -(NSString *)loadNullSession;
32 -(NSString *)loadSession:(NSString *)session;
33 -(NSString *)getUUID;
34 -(NSString *)getHR;     // remove
35 -(NSString *)getTimeAsString;
36 -(NSArray *)parseArrayFromSession:(NSString *)session;
37 -(void)test;             // remove
38 -(NSDictionary *)parseDictionary;
39 -(int)getFilledSessions;
40 @end
```

Listing 12.4: PHPControllerDelegate.h

```
1  //
2  //  PHPControllerDelegate.h
3  //  mainScreen
4  //
5  //  Created by Jake Day on 5/5/19.
6  //  Copyright  2019 Jake Day. All rights reserved.
7  //
8
9  @class PHPController;
10
11 @protocol PHPControllerDelegate <NSObject>
12
13 @optional
14
15 -(void)startCalibration:(PHPController *)request;
16 -(BOOL)calibrationSet:(PHPController *)request;
17 -(NSString *)getUUID:(PHPController *)request;
18 -(NSString *)getHR:(PHPController *)request;
19
20 @end
```

Listing 12.5: PHPController.m

```
1  //
2  //  PHPController.m
3  //  mainScreen
4  //
5  //  Created by Jake Day on 5/5/19.
6  //  Copyright  2019 Jake Day. All rights reserved.
7  //
8
9  #import "PHPController.h"
10
```

```
11 @interface PHPController ()
12
13 @end
14
15 @implementation PHPController
16
17 -(id)init {
18     self = [super init];
19     _uuid = [self loadUUID];
20     _hr = [self loadHR];     //remove
21     [self checkIfUserExists];
22     return self;
23 }
24
25 -(void)startCalibration {
26     // present ORKHrCaptureStepViewController
27     // acquire hr
28     // update calibration column in UUID's row
29 }
30
31 -(void)checkIfUserExists {
32     NSLog(@"calling checkIfUserExists");
33     // PHP script to check if user UUID exists
34     NSURL *requestURL = [NSURL URLWithString:@"http://students.engr.scu.edu/~
           jday/php/HEART/exists.php"];
35
36     //The actual request
37     ASIFormDataRequest *request = [ASIFormDataRequest requestWithURL:
           requestURL];
38     [request setPostValue:[self getUUID] forKey:@"uuid"];
39     [request setDelegate:self];
40
```

```
41      // start request
42      [request startSynchronous];
43      BOOL userExists = [[request responseString] isEqualToString:@"true"];
44
45      if (!userExists) {
46          [self addUser];
47      }
48  }
49
50  -(void)addUser {
51      NSLog(@"calling addUser");
52      // PHP script to check if user UUID exists
53      NSURL *requestURL = [NSURL URLWithString:@"http://students.engr.scu.edu/~
                jday/php/HEART/insertUser.php"];
54
55      //The actual request
56      ASIFormDataRequest *request = [ASIFormDataRequest requestWithURL:
                requestURL];
57      [request setPostValue:[self getUUID] forKey:@"uuid"];
58      [request setDelegate:self];
59
60      // start request
61      [request startSynchronous];
62  }
63
64  -(BOOL)calibrationSet {
65      NSLog(@"calling calibrationSet");
66      // PHP script to check if user UUID exists
67      NSURL *requestURL = [NSURL URLWithString:@"http://students.engr.scu.edu/~
                jday/php/HEART/calibrationSet.php"];
68
69      //The actual request
```

```
70      ASIFormDataRequest *request = [ASIFormDataRequest requestWithURL:
                requestURL];
71      [request setPostValue:[self getUUID] forKey:@"uuid"];
72      [request setDelegate:self];
73
74      // start request
75      [request startSynchronous];
76      return [[request responseString] isEqualToString:@"true"];
77  }
78
79  - (void)requestFinished:(ASIHTTPRequest *)request{
80      NSLog(@"request finished");
81      NSString *response = [request responseString];
82      NSLog(@"response: %@",response);
83  }
84
85  // Get device UUID to anonymously manage users
86  -(NSString *)loadUUID {
87      NSLog(@"calling loadUUID");
88      return [[[UIDevice currentDevice] identifierForVendor] UUIDString];
89  }
90
91  -(NSString *)getUUID {
92      return _uuid;
93  }
94
95  // outdated
96  -(NSString *)getHR {
97      return _hr;
98  }
99
100 // outdated
```

```
101 -(NSString *)loadHR {
102     NSLog(@"calling loadHR");
103     NSString *uuid = [self getUUID];
104     // PHP script to get user's hr
105     NSURL *requestURL = [NSURL URLWithString:@"http://students.engr.scu.edu/~
            jday/php/HEART/loadHR.php"];
106     //The actual request
107     ASIFormDataRequest *request = [ASIFormDataRequest requestWithURL:
            requestURL];
108     [request setPostValue:uuid forKey:@"uuid"];
109     [request setDelegate:self];
110
111     // start request
112     [request startSynchronous];
113     return [request responseString];
114 }
115
116 -(NSString *)loadNullSession {
117     NSLog(@"calling loadNullSession");
118     NSString *uuid = [self getUUID];
119     // PHP script to get user's hr
120     NSURL *requestURL = [NSURL URLWithString:@"http://students.engr.scu.edu/~
            jday/php/HEART/loadNullSession.php"];
121     //The actual request
122     ASIFormDataRequest *request = [ASIFormDataRequest requestWithURL:
            requestURL];
123     [request setPostValue:uuid forKey:@"uuid"];
124     [request setDelegate:self];
125
126     // start request
127     [request startSynchronous];
128     return [request responseString];
```

```
129 }
130
131 // change to NSData
132 -(NSString *)loadSession:(NSString *)session {
133     NSString *uuid = [self getUUID];
134     NSURL *requestURL = [NSURL URLWithString:@"http://students.engr.scu.edu/~
            jday/php/HEART/loadSession.php"];
135     ASIFormDataRequest *request = [ASIFormDataRequest requestWithURL:
            requestURL];
136     [request setPostValue:uuid forKey:@"uuid"];
137     [request setPostValue:session forKey:@"session"];
138     [request setDelegate:self];
139
140     // start request
141     [request startSynchronous];
142     return [request responseString];
143 }
144
145 -(void)updateSession:(NSString *)session {
146     NSString *uuid = [self getUUID];
147     NSString *hr = [self getHR];
148     NSURL *requestURL = [NSURL URLWithString:@"http://students.engr.scu.edu/~
            jday/php/HEART/updateSession.php"];
149     ASIFormDataRequest *request = [ASIFormDataRequest requestWithURL:
            requestURL];
150     [request setPostValue:uuid forKey:@"uuid"];
151     [request setPostValue:hr forKey:@"hr"];
152     [request setPostValue:session forKey:@"session"];
153     [request setDelegate:self];
154
155     // start request
156     [request startSynchronous];
```

```
157 }
158
159 -(void)addToSession:(NSString *)session WithHR:(NSString *)hr {
160     NSString *uuid = [self getUUID];
161
162     NSURL *requestURL = [NSURL URLWithString:@"http://students.engr.scu.edu/~
            jday/php/HEART/updateSession.php"];
163     ASIFormDataRequest *request = [ASIFormDataRequest requestWithURL:
            requestURL];
164     [request setPostValue:uuid forKey:@"uuid"];
165     [request setPostValue:hr forKey:@"hr"];
166     [request setPostValue:session forKey:@"session"];
167     [request setDelegate:self];
168
169     // start request
170     [request startSynchronous];
171 }
172
173 // outdated
174 -(NSString *)updateHR {
175     NSString *uuid = [self getUUID];
176     NSString *hr = [self getHR];
177
178     // PHP script to update user's hr
179     NSURL *requestURL = [NSURL URLWithString:@"http://students.engr.scu.edu/~
            jday/php/HEART/updateHR.php"];
180     // The actual request
181     ASIFormDataRequest *request = [ASIFormDataRequest requestWithURL:
            requestURL];
182     [request setPostValue:uuid forKey:@"uuid"];
183     [request setPostValue:hr forKey:@"hr"];
184     [request setDelegate:self];
```

```
185
186     [request startSynchronous];
187     return [request responseString];
188 }
189
190 -(NSString *)getTimeAsString {
191     NSDateFormatter *dateFormatter=[[NSDateFormatter alloc] init];
192     [dateFormatter setDateFormat:@"MM-dd-yyyy HH:mm:ss a"];
193     return [dateFormatter stringFromDate:[NSDate date]];
194 }
195
196 -(void)test {
197     NSArray *hrArray = [[NSArray alloc] initWithObjects:@58,@58,@57,@57,@71,
            @71,@50,@50,@57,@57,@50,@50,@106,@106,@92,@85,@85,@78,@78,@78,@50,
            @50,@50,@50,@50,@50,@50,@50,@50,@64,@57,@57,@57,@64,@64,@50,@50,@50
            ,@57,@57,@50,@50,@50,@50,@50,@50,@50,@50,@50,@50,@50,@50,@50,@50,@50,
            @50,@50,@50,@50,@50,@57, nil];
198     NSString *time = [self getTimeAsString];
199     NSMutableDictionary *dict = [[NSMutableDictionary alloc] init];
200     NSData *data = [NSJSONSerialization dataWithJSONObject:dict options:0
            error:nil];
201     NSString *hrString = [[NSString alloc] initWithData:data encoding:
            NSUTF8StringEncoding];
202     dict[@"time"] = time;
203     dict[@"hr"] = hrArray;
204
205     NSString *nullSession = [self loadNullSession];
206     [self addToSession:nullSession WithHR:hrString];
207 }
208
209 -(NSArray *)parseArrayFromSession:(NSString *)session {
210     NSString *requestString = [self loadSession:session];
```

```
211
212     NSRange range = NSMakeRange(1, requestString.length-2);
213     NSString *newStr = [requestString substringWithRange:range];
214
215     NSData* newData = [newStr dataUsingEncoding:NSUTF8StringEncoding];
216
217     NSError *error = nil;
218     NSArray *jsonArray = [NSJSONSerialization JSONObjectWithData:newData
            options:NSJSONReadingAllowFragments error:&error];
219
220     if (error != nil) {
221         NSLog(@"Error parsing JSON.");
222     } else {
223         NSString *className = NSStringFromClass([jsonArray class]);
224         NSLog(@"class: %@", className);
225     }
226
227     return jsonArray;
228 }
229
230 -(NSDictionary *)parseDictionary {
231     NSString *uuid = [self getUUID];
232     // PHP script to get user's hr
233     NSURL *requestURL = [NSURL URLWithString:@"http://students.engr.scu.edu/~
            jday/php/HEART/loadSession.php"];
234     //The actual request
235     ASIFormDataRequest *request = [ASIFormDataRequest requestWithURL:
            requestURL];
236     [request setPostValue:uuid forKey:@"uuid"];
237     [request setPostValue:@"SESSION5" forKey:@"session"];
238     [request setDelegate:self];
239
```

```
240     // start request
241     [request startSynchronous];
242     NSData *requestData = [request responseData];
243     NSError *error = nil;
244     id json = [NSJSONSerialization JSONObjectWithData:requestData options:
                NSJSONReadingAllowFragments error:&error];
245
246     if (error != nil) {
247         NSLog(@"Error parsing JSON, %@", [error description]);
248     } else {
249         NSLog(@"parsed: %@", json);
250         NSString *className = NSStringFromClass([json class]);
251         NSLog(@"class: %@", className);
252     }
253
254
255     return json;
256 }
257
258 -(int)getFilledSessions {
259     NSString *nullSession = [self loadNullSession];
260
261     NSLog(@"UUID: %@, nullSession: %@", [self getUUID], nullSession);
262
263     NSString *lastChar = [nullSession substringFromIndex: [nullSession length]
                - 1];
264     int lastValue = [lastChar intValue];
265
266     int filled = 5;
267
268     switch (lastValue) {
269         case 1:
```

```
270                 filled = 0;
271                 break;
272             case 2:
273                 filled = 1;
274                 break;
275             case 3:
276                 filled = 2;
277                 break;
278             case 4:
279                 filled = 3;
280                 break;
281             case 5:
282                 filled = 4;
283                 break;
284             default:
285                 break;
286         }
287
288         NSLog(@"filled: %d", filled);
289
290         return filled;
291 }
292
293 @end
```

Listing 12.6: GraphView.h

```
1 //
2 //  GraphView.h
3 //  mainScreen
4 //
5 //  Created by Jake Day on 5/6/19.
6 //  Copyright    2019 Jake Day. All rights reserved.
7 //
```

```
8
9  #import <UIKit/UIKit.h>
10 #import "PHPController.h"
11 //#import "ReportCellViewController.h"
12
13 #define kGraphHeight 394
14 #define kDefaultGraphWidth 2955
15 #define kOffsetX 0
16 #define kOffsetY 10
17 #define kStepX 50
18 #define kStepY 50
19 #define kGraphBottom 394
20 #define kGraphTop 20
21 #define kCircleRadius 3
22
23 #define hrLow 40
24 #define hrHigh 200
25
26 NS_ASSUME_NONNULL_BEGIN
27
28 @interface GraphView : UIView {
29     float pointerX;
30     float pointerY;
31     BOOL drawPointer;
32     float *data;
33     NSString *dataStr;
34     int dataLength;
35     float scale;
36 }
37
38 @property (strong, nonatomic) NSString *session;
39
```

```
40 @end
41
42 NS_ASSUME_NONNULL_END
```

Listing 12.7: GraphView.m

```
1  //
2  //  GraphView.m
3  //  mainScreen
4  //
5  //  Created by Jake Day on 5/6/19.
6  //  Copyright  2019 Jake Day. All rights reserved.
7  //
8
9  #import "GraphView.h"
10
11 @implementation GraphView
12
13 - (float *)initData {
14
15     PHPController *php = [[PHPController alloc] init];
16     NSArray *plotData = [php parseArrayFromSession:_session];
17     NSLog(@"Session: %@, Array: %@", _session, plotData);
18
19     dataLength = (int)[plotData count];
20     scale = [self getScaleFromMaxOfArray:plotData];
21
22     return [self rangeToPercentageFromArray:plotData WithLow:(int)hrLow
           andHigh:(int)hrHigh];
23 }
24
25 - (void)drawLineGraphWithContext:(CGContextRef)context {
26     data = [self initData];
27
```

```
28      // prepare gradient
29      CGGradientRef gradient;
30      CGColorSpaceRef colorspace;
31      size_t num_locations = 2;
32      CGFloat locations[2] = {0.0, 1.0};
33      CGFloat components[8] = {1.0, 0.5, 0.0, 0.2,  // Start color
34          1.0, 0.5, 0.0, 0.8}; // End color
35      colorspace = CGColorSpaceCreateDeviceRGB();
36      gradient = CGGradientCreateWithColorComponents(colorspace, components,
                locations, num_locations);
37
38      CGPoint startPoint, endPoint;
39      startPoint.x = kOffsetX;
40      startPoint.y = kGraphHeight;
41      endPoint.x = kOffsetX;
42      endPoint.y = kOffsetY;
43
44      // specify fill color
45      CGContextSetFillColorWithColor(context, [[UIColor colorWithRed:1.0 green
                :0.5 blue:0 alpha:0.5] CGColor]);
46      int maxGraphHeight = kGraphHeight - kOffsetY;
47
48      CGContextBeginPath(context);
49      CGContextMoveToPoint(context, kOffsetX, kGraphHeight);
50      CGContextAddLineToPoint(context, kOffsetX, kGraphHeight - maxGraphHeight *
                data[0]);
51      for (int i = 1; i < dataLength; i++) {
52          CGContextAddLineToPoint(context, kOffsetX + i * kStepX, kGraphHeight -
                    maxGraphHeight * data[i]);
53      }
54      CGContextAddLineToPoint(context, kOffsetX + (dataLength - 1) * kStepX,
                kGraphHeight);
```

```
55      CGContextClosePath(context);
56
57      //CGContextDrawPath(context, kCGPathFill);
58      CGContextSaveGState(context);
59      CGContextClip(context);
60      CGContextDrawLinearGradient(context, gradient, startPoint, endPoint, 0);
61
62      // cleanup
63      CGContextRestoreGState(context);
64      CGColorSpaceRelease(colorspace);
65      CGGradientRelease(gradient);
66
67      CGContextSetLineWidth(context, 2.0);
68      CGContextSetStrokeColorWithColor(context, [[UIColor colorWithRed:1.0 green
            :0.5 blue:0 alpha:1.0] CGColor]);
69
70      CGContextBeginPath(context);
71      CGContextMoveToPoint(context, kOffsetX, kGraphHeight - maxGraphHeight *
            data[0]);
72
73      for (int i = 1; i < dataLength; i++) {
74          CGContextAddLineToPoint(context, kOffsetX + i * kStepX, kGraphHeight -
                maxGraphHeight * data[i]);
75      }
76
77      CGContextDrawPath(context, kCGPathStroke);
78
79      // emphasize data points
80      CGContextSetFillColorWithColor(context, [[UIColor colorWithRed:1.0 green
            :0.5 blue:0 alpha:1.0] CGColor]);
81
82      for (int i = 1; i < dataLength - 1; i++) {
```

```
83          float x = kOffsetX + i * kStepX;
84          float y = kGraphHeight - maxGraphHeight * data[i];
85
86          CGRect rect = CGRectMake(x - kCircleRadius, y - kCircleRadius, 2 *
                kCircleRadius, 2 * kCircleRadius);
87          CGContextAddEllipseInRect(context, rect);
88      }
89
90      CGContextDrawPath(context, kCGPathFillStroke);
91  }
92
93  // Only override drawRect: if you perform custom drawing.
94  // An empty implementation adversely affects performance during animation.
95  - (void)drawRect:(CGRect)rect {
96      CGContextRef context = UIGraphicsGetCurrentContext();
97
98      // define thickness and color of grid lines
99      CGContextSetLineWidth(context, 0.6);
100     CGContextSetStrokeColorWithColor(context, [[UIColor lightGrayColor]
            CGColor]);
101     CGFloat dash[] = {2.0, 2.0};
102     CGContextSetLineDash(context, 0.0, dash, 2);
103
104     // draw vertical grid lines
105     int vLines = (kDefaultGraphWidth - kOffsetX) / kStepX;
106
107     for (int i = 0; i <= vLines; i++) {
108         CGContextMoveToPoint(context, kOffsetX + i * kStepX, kGraphTop);
109         CGContextAddLineToPoint(context, kOffsetX + i * kStepX, kGraphBottom);
110     }
111
112     // draw horizontal grid lines
```

```
113
114     int hLines = (kGraphBottom - kGraphTop - kOffsetY) / kStepY;
115
116     for (int i = 0; i <= hLines; i++) {
117         CGContextMoveToPoint(context, kOffsetX, kGraphBottom - kOffsetY - i *
                kStepY);
118         CGContextAddLineToPoint(context, kDefaultGraphWidth, kGraphBottom -
                kOffsetY - i * kStepY);
119     }
120
121     // commit drawing
122     CGContextStrokePath(context);
123
124     // disable dash
125     CGContextSetLineDash(context, 0, NULL, 0);
126
127     [self drawLineGraphWithContext:context];
128
129     // Drawing text
130     CGContextSetTextMatrix(context, CGAffineTransformMake(1.0, 0.0, 0.0, -1.0,
            0.0, 0.0));
131     CGContextSelectFont(context, "Helvetica", 18, kCGEncodingMacRoman);
132     CGContextSetTextDrawingMode(context, kCGTextFill);
133     CGContextSetFillColorWithColor(context, [[UIColor colorWithRed:0 green:0
            blue:0 alpha:1.0] CGColor]);
134
135     // display x labels
136     for (int i = 1; i < dataLength; i++) {
137         NSString *theText = [NSString stringWithFormat:@"%d", i];
138         CGSize labelSize = [theText sizeWithFont:[UIFont fontWithName:@"
                Helvetica" size:18]];
139         CGContextShowTextAtPoint(context, kOffsetX + i * kStepX - labelSize.
```

```
                width/2, kGraphBottom - kOffsetY/2, [theText
                cStringUsingEncoding:NSUTF8StringEncoding], [theText length]);
140
141     }
142
143     // display y labels
144     NSArray *yLabels = [[NSArray alloc] initWithObjects:@65, @85, @105, @125,
            @145, @165, @185, @205, nil];
145
146     for (int i = 1; i < [yLabels count]; i++) {
147         int offset = (int)[yLabels count] - i - 1;
148         NSString *yLabelText = [NSString stringWithFormat:@"%@", yLabels[
                offset]];
149         CGSize labelSize = [yLabelText sizeWithFont:[UIFont fontWithName:@"
                Helvetica" size:18]];
150         CGContextShowTextAtPoint(context, kOffsetX, kOffsetY + i * kStepY -
                labelSize.height, [yLabelText cStringUsingEncoding:
                NSUTF8StringEncoding], [yLabelText length]);
151     }
152
153 }
154
155 - (float *)rangeToPercentageFromArray:(NSArray *)array WithLow:(int)low
        andHigh:(int)high {
156     NSInteger arrayCount = [array count];
157     float *convertedArray = (float *)malloc(sizeof(float) * arrayCount);
158
159     for (int i = 0; i < arrayCount; i++) {
160         int obj = (int)[(NSNumber *)[array objectAtIndex:i] integerValue];
161         float range = high - low;
162         float offset = obj - low;
163         // TODO: implement scaling
```

```
164             float percentage = offset / range;
165             convertedArray[i] = percentage;
166         }
167 
168         return convertedArray;
169 }
170 
171 - (float)getScaleFromMaxOfArray:(NSArray *)array {
172         // sort array
173         [array sortedArrayUsingSelector:@selector(compare:)];
174 
175         // get max
176         int length = (int)[array count];
177         float max = (float)[(NSNumber *)[array objectAtIndex:length-1]
                integerValue];
178 
179         if (max > 200) {
180             return 1.0;
181         } else if (max > 150) {
182             return 1.25;
183         } else if (max > 100) {
184             return 1.5;
185         } else if (max > 50) {
186             return 2;
187         } else {
188             return 2.25;
189         }
190 }
191 
192 @end
```

Listing 12.8: PointerView.h

```
1 //
```

```
2  //  PointerView.h
3  //  mainScreen
4  //
5  //  Created by Jake Day on 5/6/19.
6  //  Copyright  2019 Jake Day. All rights reserved.
7  //
8
9  #import <UIKit/UIKit.h>
10 #import "GraphView.h"
11 #import "PHPController.h"
12
13 NS_ASSUME_NONNULL_BEGIN
14
15 @interface PointerView : UIView {
16     float pointerX;
17     float pointerY;
18     BOOL drawPointer;
19     NSString *session;
20 }
21
22 @property (nonatomic, strong) NSString *session;
23
24 @end
25
26 NS_ASSUME_NONNULL_END
```

Listing 12.9: PointerView.m

```
1  //
2  //  PointerView.m
3  //  mainScreen
4  //
5  //  Created by Jake Day on 5/6/19.
6  //  Copyright  2019 Jake Day. All rights reserved.
```

```
7  //
8
9  #import "PointerView.h"
10
11 @implementation PointerView
12
13
14 // Only override drawRect: if you perform custom drawing.
15 // An empty implementation adversely affects performance during animation.
16 - (void)drawRect:(CGRect)rect {
17     if (drawPointer) {
18         CGContextRef context = UIGraphicsGetCurrentContext();
19         CGRect frame = self.frame;
20
21         CGContextTranslateCTM(context, 0, kGraphHeight);
22         CGContextScaleCTM(context, 1, -1);
23
24         CGContextSetLineWidth(context, 2.0);
25         CGContextSetStrokeColorWithColor(context, [[UIColor colorWithRed:0.4
                green:0.8 blue:0.4 alpha:1.0] CGColor]);
26         CGContextMoveToPoint(context, pointerX, 0);
27         CGContextAddLineToPoint(context, pointerX, frame.size.height);
28         CGContextStrokePath(context);
29
30         [self drawHRLabels:context];
31     }
32 }
33
34 - (void)drawHRLabels:(CGContextRef)context {
35     // Drawing text
36     CGContextSelectFont(context, "Helvetica", 18, kCGEncodingMacRoman);
37     CGContextSetTextDrawingMode(context, kCGTextFill);
```

```
38      CGContextSetFillColorWithColor(context, [[UIColor colorWithRed:0 green:0
            blue:0 alpha:1.0] CGColor]);
39
40      // display hr label
41      float xValue = [self getXValueFromXCoordinate:pointerX];
42      float yValue = [self getYValueFromYCoordinate:pointerY];
43      int lower = [self getLowerBoundFromX:xValue];
44      int upper = [self getUpperBoundFromX:xValue];
45
46      // get data from session
47      PHPController *php = [[PHPController alloc] init];
48      NSArray *data = [php parseArrayFromSession:_session];
49
50      int lowerObj = (int)[(NSNumber *)[data objectAtIndex:lower] integerValue];
51      int upperObj = (int)[(NSNumber *)[data objectAtIndex:upper] integerValue];
52
53      // draw lower hr value
54      NSString *lowerText = [NSString stringWithFormat:@"%d BPM", lowerObj];
55      CGSize lowerLabelSize = [lowerText sizeWithFont:[UIFont fontWithName:@"
            Helvetica" size:18]];
56      CGContextShowTextAtPoint(context, pointerX - 1.25*lowerLabelSize.width,
            kGraphHeight - pointerY, [lowerText cStringUsingEncoding:
            NSUTF8StringEncoding], [lowerText length]);
57      // draw upper hr value
58      NSString *upperText = [NSString stringWithFormat:@"%d BPM", upperObj];
59      CGSize upperLabelSize = [upperText sizeWithFont:[UIFont fontWithName:@"
            Helvetica" size:18]];
60      CGContextShowTextAtPoint(context, pointerX + 0.25*upperLabelSize.width,
            kGraphHeight - pointerY, [upperText cStringUsingEncoding:
            NSUTF8StringEncoding], [upperText length]);
61  }
62
```

```
63 - (void)touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event
64 {
65     UITouch *touch = [touches anyObject];
66     CGPoint point = [touch locationInView:self];
67     pointerX = point.x;
68     pointerY = point.y;
69     NSLog(@"x: %f\t y: %f", point.x, point.y);
70
71     float xValue = [self getXValueFromXCoordinate:pointerX];
72     float yValue = [self getYValueFromYCoordinate:pointerY];
73     int lower = [self getLowerBoundFromX:xValue];
74     int upper = [self getUpperBoundFromX:xValue];
75     [self printPercentageFromLower:lower AndUpper:upper WithValue:xValue];
76     NSLog(@"xValue: %f\tyValue: %f\tlower: %d\tupper: %d", xValue, yValue,
           lower, upper);
77     NSArray *data = [[NSArray alloc] initWithObjects:@58,@58,@57,@57,@71,@71,
           @50,@50,@57,@57,@50,@50,@106,@106,@92,@85,@85,@78,@78,@78,@50,@50,
           @50,@50,@50,@50,@50,@50,@50,@64,@57,@57,@57,@64,@64,@50,@50,@50,@57
           ,@57,@50,@50,@50,@50,@50,@50,@50,@50,@50,@50,@50,@50,@50,@50,@50,
           @50,@50,@50,@50,@57, nil];
78     [self getYCoordinateFromData:data AtIndex:lower];
79     [self getYCoordinateFromData:data AtIndex:upper];
80
81     drawPointer = YES;
82     [self setNeedsDisplay];
83 }
84
85 // TODO: get hr value based on x y coordinates pressed
86 // idea:     get x coordinate of button pressed
87 //           find which interval that corresponds to
88 //           find out how far along that is between the two to get offset
89 //           get the slope of the 2 values with offset
```

```
90  - (float)getXValueFromXCoordinate:(float)x {
91      float intervals = kDefaultGraphWidth / kStepX;
92      return (x / kDefaultGraphWidth) * intervals;
93  }
94
95  - (float)getYValueFromYCoordinate:(float)y {
96      float intervals = kGraphHeight / kStepY;
97      return (y / kGraphHeight) * intervals;
98  }
99
100 - (int)getLowerBoundFromX:(float)x {
101     return floorf(x);
102 }
103
104 - (int)getUpperBoundFromX:(float)x {
105     return ceilf(x);
106 }
107
108 // percentage between lower and upper x values
109 - (void)printPercentageFromLower:(int)lower AndUpper:(int)upper WithValue:(
        float)value {
110     NSArray *plotData = [[NSArray alloc] initWithObjects:@58,@58,@57,@57,@71,
            @71,@50,@50,@57,@57,@50,@50,@106,@106,@92,@85,@85,@78,@78,@78,@50,
            @50,@50,@50,@50,@50,@50,@50,@50,@64,@57,@57,@57,@64,@64,@50,@50,@50
            ,@57,@57,@50,@50,@50,@50,@50,@50,@50,@50,@50,@50,@50,@50,@50,@50,
            @50,@50,@50,@50,@50,@57, nil];
111
112     float percentage = (value - lower) / (upper - lower);
113     NSLog(@"percentage: %f, data[lower] = %@, data[upper] = %@", percentage,
            plotData[lower], plotData[upper]);
114 }
115
```

```
116 - (float)getSlopeFromX1:(float)x1 Y1:(float)y1 AndX2:(float)x2 Y2:(float)y2 {
117     float slope = (y2 - y1) / (x2 - x1);
118     NSLog(@"slope: %f", slope);
119     return slope;
120 }
121
122 - (float)getXCoordinateFromData:(NSArray *)data {
123     return 0.0;
124 }
125
126 - (void)getYCoordinateFromData:(NSArray *)data AtIndex:(int)index {
127     int obj = (int)[(NSNumber *)[data objectAtIndex:index] integerValue];
128
129     float oldValue = [self getYValueFromYCoordinate:pointerY];
130     float oldRange = kGraphHeight / kStepY;
131
132     float range = hrHigh - hrLow;
133     float newValue = (((oldValue - 0) * range) / oldRange) + hrLow;
134     NSLog(@"oldValue: %f, newValue: %f", oldValue, ceilf(hrHigh + hrLow -
            newValue));
135 }
136
137 @end
```

Listing 12.10: ReportViewController.h

```
1 //
2 //  ReportViewController.h
3 //  mainScreen
4 //
5 //  Created by Jake Day on 5/5/19.
6 //  Copyright    2019 Jake Day. All rights reserved.
7 //
8
```

```
9  #import <UIKit/UIKit.h>
10 #import "PHPController.h"
11 #import "ReportCellViewController.h"
12 #import <ResearchKit/ResearchKit.h>
13
14 NS_ASSUME_NONNULL_BEGIN
15
16 @interface ReportViewController : UIViewController <
       UINavigationControllerDelegate, UITableViewDelegate,
       UITableViewDataSource>
17
18 - (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView;
19 - (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(
       NSInteger)section;
20 - (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath
       :(NSIndexPath *)indexPath;
21 - (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(
       NSIndexPath *)indexPath;
22 - (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView;
23
24 - (NSString *)getSession;
25 @end
26
27 NS_ASSUME_NONNULL_END
```

Listing 12.11: ReportViewController.m

```
1 //
2 //  ReportViewController.m
3 //  mainScreen
4 //
5 //  Created by Jake Day on 5/5/19.
6 //  Copyright    2019 Jake Day. All rights reserved.
7 //
```

```
8
9  #import "ReportViewController.h"
10
11 @interface ReportViewController ()
12
13 @property (weak, nonatomic) IBOutlet UITableView *reportTableView;
14 @property (strong,nonatomic) NSMutableArray *tableData;
15 @property (weak, nonatomic) IBOutlet UIBarButtonItem *backButton;
16 @property (weak, nonatomic) IBOutlet UINavigationItem *navigationItem;
17 @property (strong, nonatomic) NSString *session;
18
19 @end
20
21 @implementation ReportViewController
22
23 - (void)viewDidLoad {
24     [super viewDidLoad];
25     [self configTableView];
26     // Do any additional setup after loading the view.
27     PHPController *php = [[PHPController alloc] init];
28     int size = [php getFilledSessions];
29     NSLog(@"size: %d", size);
30     self.tableData = [NSMutableArray arrayWithCapacity:size];
31
32     for (int i = 0; i < size; i++) {
33         NSString *sessionStr = @"Session ";
34         sessionStr = [sessionStr stringByAppendingString:[NSString
                stringWithFormat:@"%d", i+1]];
35         [self.tableData insertObject:sessionStr atIndex:i];
36     }
37 }
38
```

```
39 -(void)configTableView {
40     self.reportTableView = [[UITableView alloc] initWithFrame:self.view.bounds
               style:UITableViewStylePlain];
41     self.reportTableView.delegate = self;
42     self.reportTableView.dataSource = self;
43     [self.view addSubview:self.reportTableView];
44 }
45
46 - (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView {
47     return 1;
48 }
49
50 - (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(
        NSInteger)section {
51     return _tableData.count;
52 }
53
54 - (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath
        :(NSIndexPath *)indexPath {
55     static NSString *cellID = @"cellID";
56     UITableViewCell *cell = [self.reportTableView
            dequeueReusableCellWithIdentifier:cellID];
57
58     if (cell == nil) {
59         cell = [[UITableViewCell alloc] initWithStyle:
                UITableViewCellStyleDefault reuseIdentifier:cellID];
60     }
61
62     cell.textLabel.text = [_tableData objectAtIndex:indexPath.row];
63     return cell;
64 }
65 - (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(
```

```
        NSIndexPath *)indexPath {
66      NSLog(@"title of cell %@", [_tableData objectAtIndex:indexPath.row]);
67
68      NSString *sessionStr = @"SESSION";
69      _session = [sessionStr stringByAppendingString:[NSString stringWithFormat:
            @"%ld", indexPath.row+1]];
70
71      [self performSegueWithIdentifier:@"ReportCellSegue" sender:self];
72  }
73
74  - (NSString *)getSession {
75      return _session;
76  }
77
78
79  #pragma mark - Navigation
80
81  // In a storyboard-based application, you will often want to do a little
        preparation before navigation
82  - (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender {
83
84      if ([segue.identifier isEqualToString:@"ReportCellSegue"]) {
85          ReportCellViewController *destinationViewController = segue.
                destinationViewController;
86
87          // pass session variable to child UIView
88          destinationViewController.session = _session;
89      }
90
91      //...
92  }
93
```

```
94 @end
```

Listing 12.12: ReportCellViewController.h

```
1  //
2  //  ReportCellViewController.h
3  //  mainScreen
4  //
5  //  Created by Jake Day on 5/6/19.
6  //  Copyright  2019 Jake Day. All rights reserved.
7  //
8
9  #import <UIKit/UIKit.h>
10 #import "GraphView.h"
11 #import "PointerView.h"
12 #import "ReportViewController.h"
13
14 NS_ASSUME_NONNULL_BEGIN
15
16 @interface ReportCellViewController : UIViewController
17
18 @property (weak, nonatomic) IBOutlet UIScrollView *scroller;
19 @property (weak, nonatomic) IBOutlet GraphView *graphView;
20 @property (weak, nonatomic) IBOutlet PointerView *pointerView;
21 @property (strong, nonatomic) NSString *session;
22
23 - (NSString *)getSession;
24
25 @end
26
27 NS_ASSUME_NONNULL_END
```

Listing 12.13: ReportCellViewController.m

```
1  //
```

```
2  //  ReportCellViewController.m
3  //  mainScreen
4  //
5  //  Created by Jake Day on 5/6/19.
6  //  Copyright  2019 Jake Day. All rights reserved.
7  //
8
9  #import "ReportCellViewController.h"
10
11 @interface ReportCellViewController ()
12
13 @end
14
15 @implementation ReportCellViewController
16
17 - (void)viewDidLoad {
18     [super viewDidLoad];
19     // Do any additional setup after loading the view.
20     _scroller.contentSize = CGSizeMake(kDefaultGraphWidth, kGraphHeight);
21
22     NSLog(@"the session is %@", _session);
23     self.pointerView.session = _session;
24     self.graphView.session = _session;
25 }
26
27 - (BOOL)shouldAutorotate {
28     return YES;
29 }
30
31 - (NSString *)getSession {
32     return _session;
33 }
```

```
34
35 /*
36 #pragma mark - Navigation
37
38 // In a storyboard-based application, you will often want to do a little
       preparation before navigation
39 - (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender {
40     // Get the new view controller using [segue destinationViewController].
41     // Pass the selected object to the new view controller.
42 }
43 */
44
45 @end
```

12.2 Database Querying Source Code

Listing 12.14: Database.php

```
1  <?php
2
3  class Database {
4     public static $connection;
5
6     public static function db_connect() {
7        if (!isset($connection)) {
8           $config = parse_ini_file('./private/config.ini');
9           self::$connection = mysqli_connect(
10             $config['host'],
11             $config['user'],
12             $config['pass'],
13             $config['name']
14          );
15       }
16
```

```
17          if (self::$connection->connect_error) {
18              die("Connection failed: " .
19                  self::$connection->connect_error
20              );
21              return FALSE;
22          }
23
24          return TRUE;
25      }
26
27      public static function db_close() {
28          self::$connection->close();
29          return !is_resource(self::$connection);
30      }
31  }
```

Listing 12.15: Entry.php

```
1   <?php
2
3   require_once "Database.php";
4
5   class Entry {
6       public static function selectUUID($value) {
7       Database::db_connect();
8
9           $query = "SELECT UUID FROM heart WHERE UUID=? LIMIT 1";
10
11          $stmt = Database::$connection->prepare($query);
12          $stmt->bind_param("s", $value);
13          $stmt->execute();
14          $stmt->store_result();
15          $stmt->bind_result($uuid);
16
```

```
17          while ($stmt->fetch()) {
18              echo $uuid . '<br>';
19          }
20
21          Database::db_close();
22      }
23
24      public static function selectOne($key, $value) {
25          Database::db_connect();
26
27                      $query = "SELECT UUID FROM heart WHERE ?=? LIMIT 1";
28          echo $query . '<br>';
29          $stmt = Database::$connection->prepare($query);
30
31          if (!$stmt) {
32              echo "fail";
33          }
34
35          $stmt->bind_param("ss", $key, $value);
36                      $stmt->execute();
37          $stmt->store_result();
38          $stmt->bind_result($uuid);
39
40          echo "key: " . $key . ", value: " . $value . "<br>";
41          $numRows = $stmt->num_rows;
42          echo $numRows;
43          while ($stmt->fetch()) {
44              echo 'UUID: ' . $uuid . '<br>';
45          }
46
47
48          $stmt->free_result();
```

```
49          $stmt->close();
50
51          Database::db_close();
52      }
53
54      public static function selectAll() {
55          Database::db_connect();
56
57          $query = "SELECT * FROM heart";
58          // TODO: finish
59      }
60  }
```

Listing 12.16: User.php

```
1   <?php
2
3   require_once "Database.php";
4
5   class User {
6       public static function selectOne($column, $value) {
7                   Database::db_connect();
8
9                   $query = "SELECT $column FROM heart
10                          WHERE UUID=? LIMIT 1";
11
12                  $stmt = Database::$connection->prepare($query);
13                  $stmt->bind_param("s", $value);
14                  $stmt->execute();
15                  $stmt->store_result();
16                  $stmt->bind_result($uuid);
17
18          $result = null;
19
```

```
20                  while ($stmt->fetch()) {
21                          $result = $uuid;
22                  }
23
24                  $stmt->free_result();
25                  $stmt->close();
26
27                  Database::db_close();
28
29                  return $result;
30          }
31
32          public static function selectUUID($value) {
33           return self::selectOne("UUID", $value);
34      }
35
36      public static function selectCalibration($value) {
37          return self::selectOne("CALIBRATION", $value);
38      }
39
40      public static function selectSession($session, $value) {
41          return self::selectOne($session, $value);
42      }
43
44      public static function selectNullSession($uuid) {
45          Database::db_connect();
46
47          $sessions = array(
48              "SESSION1",
49              "SESSION2",
50              "SESSION3",
51              "SESSION4",
```

```
52              "SESSION5"
53          );
54
55
56          for ($index = 0; $index < count($sessions); $index++) {
57              $query = "SELECT (
58                      IF($sessions[$index] IS NULL, 1, 0)
59                  ) as isNull
60                  FROM heart WHERE UUID=?";
61              $stmt = Database::$connection->prepare($query);
62              $stmt->bind_param("s", $uuid);
63              $stmt->execute();
64              $stmt->bind_result($result);
65
66              $isNull = 0;
67
68              while ($stmt->fetch()) {
69                  $isNull = $result;
70              }
71
72              if ($isNull) {
73                  $stmt->free_result();
74                  $stmt->close();
75                  Database::db_close();
76                  return $sessions[$index];
77              }
78
79          }
80
81          $stmt->free_result();
82          $stmt->close();
83
```

```
84          Database::db_close();
85
86          return "NONNULL";
87      }
88
89      public static function selectHR($value) {
90          return self::selectOne("HR", $value);
91      }
92
93      public static function exists($uuid) {
94          return self::selectUUID($uuid) !== null;
95      }
96
97      public static function calibrationSet($uuid) {
98          if (!self::exists($uuid)) {
99              return false;
100         }
101         return self::selectCalibration($uuid) !== null;
102     }
103
104     public static function insertUser($uuid) {
105         Database::db_connect();
106
107         $query = "INSERT INTO heart (
108             UUID
109         ) VALUES (
110             ?
111         )";
112
113         $stmt = Database::$connection->prepare($query);
114         $stmt->bind_param("s", $uuid);
115                 $stmt->execute();
```

```
116                 $stmt->close();
117
118                 Database::db_close();
119     }
120
121     public static function updateHR($uuid, $hr) {
122         Database::db_connect();
123
124         $query = "UPDATE heart SET HR=? WHERE UUID=?";
125
126         $stmt = Database::$connection->prepare($query);
127         $stmt->bind_param("ss", $hr, $uuid);
128         $stmt->execute();
129         $stmt->close();
130
131         Database::db_close();
132     }
133
134     public static function updateSession($uuid, $hr, $session) {
135         Database::db_connect();
136
137         $query = "UPDATE heart SET $session=? WHERE UUID=?";
138
139         $stmt = Database::$connection->prepare($query);
140         $stmt->bind_param("ss", $hr, $uuid);
141         $stmt->execute();
142         $stmt->close();
143
144         Database::db_close();
145     }
146
147     public static function add() {
```

```
148         Database::db_connect();
149
150         $uuid = $_POST["uuid"];
151         $hr = $_POST["hr"];
152
153         // TODO: return if _POST is empty
154
155         $query = Database::$connection->prepare("INSERT INTO heart (
156                         UUID,
157                         HR
158                 ) VALUES (
159                         ?,
160             ?
161                 )");
162
163         $query->bind_param("ss", $uuid, $hr);
164         $query->execute();
165
166         Database::db_close();
167     }
168 }
```

Listing 12.17: calibrationSet.php

```
1  <?php
2
3  require_once("User.php");
4
5  if (!isset($_POST["uuid"])) {
6      echo "NOT SET";
7  } else {
8      $uuid = $_POST["uuid"];
9      echo json_encode(User::calibrationSet($uuid));
10 }
```

Listing 12.18: exists.php

```
<?php

require_once("User.php");

if (!isset($_POST["uuid"])) {
    echo "NOT SET";
} else {
    $uuid = $_POST["uuid"];
    echo json_encode(User::exists($uuid));
}
```

Listing 12.19: inserUser.php

```
<?php

require_once("User.php");

$uuid = $_POST["uuid"];

User::insertUser($uuid);
```

Listing 12.20: loadHR.php

```
<?php

require_once("User.php");

if (!isset($_POST["uuid"])) {
    echo "NOT SET";
} else {
    $uuid = $_POST["uuid"];
    echo json_encode(User::selectHR($uuid));
}
```

Listing 12.21: loadNullSession.php

```
<?php

require_once("User.php");

$uuid = $_POST["uuid"];

echo User::selectNullSession($uuid);
```

Listing 12.22: loadSession.php

```
<?php

require_once("User.php");

if (!isset($_POST["uuid"]) || !isset($_POST["session"])) {
    echo "NOT SET";
} else {
    $uuid = $_POST["uuid"];
    $session = $_POST["session"];
    echo json_encode(User::selectSession($session, $uuid));
}
```

Listing 12.23: updateHR.php

```
<?php

require_once("User.php");

$uuid = $_POST["uuid"];
$hr = $_POST["hr"];

User::updateHR($uuid, $hr);
```

Listing 12.24: updateSessions.php

```
1 <?php
2
3 require_once("User.php");
4
5 $uuid=$_POST["uuid"];
6 $session=$_POST["session"];;
7 $hr=$_POST["hr"];
8
9 User::updateSession($uuid, $hr, $session);
```