Deliverable #3

Team Members: Hannah Posch, Alex Thropp, Daniel Baczmaga, Daniel Lee

3 Deliverable 3

3.0 Architectural Description

We organized our testing framework into the following sections:
/TestAutomation

Joda-time-2.10.1.jar (Required Library)

Junit-4.12.jar (Required Library)

/project (Holds Glucosio repository)

/glucosio-android-develop

/scripts

runAllTests.sh

/testCases

testCase01.txt

testCase02.txt

...

/testCasesExecutables

/TeamCorrectTestExecutables

testCase01.java

testCase02.java

….

TestCasesInput.txt (All input for test cases)

Relevant Glucosio java files needed for compiling

/TeamCorrectFaultInjection

testCaseXX.java (test cases with faults injected)

….

/test

/temp

/oracles

/docs

README.md

/reports

testResults.html

3.1 Documentation

We are using the format specified on the class web page to document our test cases.

- test number or ID
- requirement being tested
- component being tested
- method being tested
- test input(s) including command-line argument(s)
- expected outcome(s)

3.2 Requirement List made by Team Correct

RS001: Record user input
Glucose
HbA1C reading
A1C reading
Cholesterol level
Blood pressure
Ketones
Body weight
RS002: Display last check

RS003: Provide a log of user inputs for the past days
RS004: Display graphically
RS005: Opt in option of sharing diabetes data with researchers (API)
RS006: Connect to other services (Google Fit, My Fitness Pal)
RS007: Export data to various formats and services
RS008: Sleep
RS009: Treatments
RS010: Medication
RS011: Exercise
RS012: Food and carbs
RS013: Set reminders
RS014: Export pictures of graphs to camera roll
RS015: Log diabetes type
RS016: Provide tips
RS017: Build User
RS018: Provide correct Glucose calculations based on User's data
RS019: Provide alerts

3.3 Instructions to run tests

1. Open Terminal
2. git clone https://github.com/csci-362-fall-2018-01/TeamCorrect
3. cd TeamCorrect/TestAutomation/scripts (May need additional navigation, depending on where the repository was cloned to)
4. ./runAllTests.sh
5. Output will be stored in an html file which should open automatically in the default browser.

*If any of the above commands return permission denied, use 'sudo' in front of the command.

*If the ./runAllTests.sh returns command not found, use 'sudo chmod 755 runAllTests.sh', then './runAllTests.sh'

3.4 Test Case Specifications

Test Number: 01
Requirement being tested: RS013 Set Reminders

Component being tested: Reminder.java

Method being tested: setMetric

Test input(s) including command-line argument(s):

0ml, 9ml, 10ml, 20ml

Expected outcome(s):

Metric returned = 0ml

Metric returned = 9ml

Metric returned = 10ml

Metric returned = 20ml

Test Number: 02

Requirement being tested: RS001 Record user input - Glucose

Component being tested: Glucose Converter

Method being tested: glucoseToA1C

Test input(s) including command-line argument(s):

10.0, 25.0, 50.0, 0.0

Expected outcome(s):

10.0 glucose level converted to 1.98

25.0 glucose level converted to 2.5

50.0 glucose level converted to 3.37

0.0 glucose level converted to 1.63

Test Number: 03

Requirement being tested: RS001 Record user input - Ketone

Component being tested: KetoneReading.java

Method being tested: setReading

Test input(s) including command-line argument(s):

2.0, 10.0, 50.0, 100.0

Expected outcome(s):

Reading returned = 2.0

Reading returned = 10.0

Reading returned = 50.0

Reading returned = 100.0

Test Number: 04

Requirement being tested: RS001 Record user input - Body Weight

Component being tested: WeightReading.java

Method being tested: setReading

Test input(s) including command-line argument(s):

100.0, 50.0, 10.0 0.0

Expected outcome(s):

Reading returned = 100.0

Reading returned = 50.0

Reading returned = 10.0

Reading returned = 0.0

Test Number: 05

Requirement being tested: RS001 Record user input - Blood Pressure

Component being tested: PressureReading.java

Method being tested: setMinReading

Test input(s) including command-line argument(s):

110.0, 30.0, 15.0, 2.0

Expected outcome(s):

Min Reading returned = 110.0

Min Reading returned = 30.0

Min Reading returned = 15.0

Min Reading returned = 2.0

3.5 Evaluation and Experiences

Deliverable 3's production was probably the most tricky as we were required to navigate between a shell script and java files, and ensure the syntax between them worked seamlessly. We were required to create a shell script in bash that would automate our java test files for our project, Glucosio. This required that our shell could navigate the correct directories and automatically run through our test cases. We also needed to ensure that our java test files could compile and execute from terminal as that is how the framework would run each file. These requirements required time and collaboration to complete and overall we are happy with the progress we made. The syntax of the script may have required a bit of trial and error, but once it was working, it made the testing of our project Glucosio much more efficient.