

# Deliverable 2: Test Plan

Project: Glucosio

By: Dwayne Ferguson, Jason Adler, and Shefali Emmanuel

## The Testing Process

The goal is to eventually test 25 cases for the Glucosio Software. The team will identify five methods to test five times each. We will be writing a script to open our driver. The driver will send arguments through the main method, which will create an instance of the class to test. The arguments will be run through the method being tested and then output to a file on the computer. The driver will then locate the file, open it, and compare the outcome with the oracle. Finally, the console will display the expected and actual outcomes.

## Requirements Traceability

We do not know the requirements the original developer was trying to meet but as Glucosio was created as a diabetes management tool we have developed a core requirement that will likely envelope all of our test cases.

- Function: Compute Glucose & A1C levels
- Description: Compute glucose and A1C levels at regular intervals throughout the day and when the sensor transmits data to the application or the user inputs data.
- Inputs: User weight, blood sugar reading
- Source: Reading from sensor or user input.
- Outputs: The calculated blood sugar and recommended levels according to the ADA, AACE, and UKNICE.
- Destination: User Screen (MainPresenter)
- Action: Glucose readings are received by a sensor or input from a user. The user's self-entered weight is used with the glucose reading to calculate A1C levels according to the NGSP and IFCC. These values are then compared to the recommended levels by the ADA, AACE, and UKNICE and displayed on the user's screen.
- Requires: User or sensor input of weight and glucose levels.
- Precondition: None
- Postcondition: Read value is stored in a user's profile.
- Side effects: None.

## Tested Items

Methods:

glucosio-android -> app -> src -> main -> java -> org -> glucosio -> android -> tools -> ...

1. FormatDateTime.java -> convertDate(String datetime)
2. FormatDateTime.java -> convertRawTime(Date datetime)
3. GlucoseRanges.java -> colorFromReading(int reading)
4. GlucoseRanges.java -> stringToColor(String color)
5. GlucosioConverter.java -> kgToLb(int kg)
6. GlucosioConverter.java -> alcToGlucose(double alc)
7. GlucosioConverter.java -> glucoseToAlc(double mgDI)
8. GlucosioConverter.java -> glucoseToMmolL(double mgDI)
9. GlucosioConverter.java -> glucoseToMgDI(double mmolL)
10. GlucosioConverter.java -> round(double value, int places)
11. InputFilterMinMax.java -> isInRange(int a, int b, int c)
12. readingTools.java -> hourToSpinnerType(int hour)

## Test Cases

```
1  1
2
3  # When the input is 0 the output should result in 0 as well.
4
5  glucosio-android/app/src/main/java/org/glucosio/android/tools/GlucosioConverter.java
6
7  kgToLb(int kg)
8
9  0
10
11  0
```

---

```
1 2
2
3 # When the input is 0< the output should result in positive number as well.
4
5 glucosio-android/app/src/main/java/org/glucosio/android/tools/GlucosioConverter.java
6
7 kgToLb(int kg)
8
9 1
10
11 2.20462
```

```
1 3
2
3 # When the input is <0 the output should result in negative number as well.
4
5 glucosio-android/app/src/main/java/org/glucosio/android/tools/GlucosioConverter.java
6
7 kgToLb(int kg)
8
9 -1
10
11 -2.20462
```

```
1 4
2
3 # When the input is 99< the output should result in 99< as well.
4
5 glucosio-android/app/src/main/java/org/glucosio/android/tools/GlucosioConverter.java
6
7 kgToLb(int kg)
8
9 100
10
11 220.462
```

```
1 5
2
3 # When the input is 1500< the output should result in 1500< as well.
4
5 glucosio-android/app/src/main/java/org/glucosio/android/tools/GlucosioConverter.java
6
7 kgToLb(int kg)
8
9 1500
10
11 3306.934
```

## Testing Schedule

*October 1:* Deliverable #2 - Complete Test Plan including at least 5 of the eventual 25 test cases that will be developed

*October 29:* Deliverable #3 - Design and build an automated testing framework that you will use to implement your test plan per the Project Specifications document.

*November 12:* Deliverable #4 - Complete the design and implementation of your testing framework as specified in Deliverable #3. You are now ready to test your project in earnest. You will create 25 test cases that your framework will automatically use to test your H/FOSS project. Report: This is Chapter 4 of your Final Report. Report on your experiences and provide the details produced in the task above. This is your opportunity to get your report organized into a professional-quality deliverable. Document progress on the wiki.

*November 19:* Deliverable #5 - Design and inject 5 faults into the code you are testing that will cause at least 5 tests to fail, but hopefully not all tests to fail. Exercise your framework and analyze the results.

## Test Recording Procedures

As outlined in "The testing process" above, the team will write a script that will execute the methods with test values. We will identify five different methods to test five different times each. The results of these executions will be outputted to a file on the user's computer. The file will contain the following test cases:

1. Test number or ID
2. Requirement being tested
3. Component being tested
4. Method being tested

5. Test input(s) including command-line argument(s)
6. Expected outcome(s)

## **Hardware and Software Requirements**

VirtualBoxVM 6.0.10

Linux Ubuntu (64-bit) 18.04

Android Studio 3.5

Gradle 4.4.1

Java 1.8.0\_222

## **Constraints**

- lack of people
- the learning curve with android development
- lack of familiarity with the correct version of the project to utilize from HFOSS Developers
- running into problems

## **Experiences**

We have been having trouble building the entirety of the program on each team member's individual machine. Our hypothesis is that this is a version issue in relation to our project. The creators did not design mandatory version requirements for the project to run efficiently. We are currently performing a brute force approach to this problem. Since the project built on one computer, we were able to find twelve test cases in one sitting, which helped us on our path to completing deliverable 2.