EasyGlucose

CMPT 276 – Group 01 – Glucinators
Assignment 5 – Design Document

Website: https://sites.google.com/view/cmpt276-summer2018/

Anmol Bajaj
Faisal Atif
Zhixin Huang
Tony Liu
Henry Yip

August 1st, 2018

1. Table of Contents

2. Revision History

Revision	Status	Publication/Revision Date	By
0.0	Rough draft created with requirement for each category	July 11, 2018	Henry Yip
1.0	Design document created	July 12, 2018	Everyone
1.1	Added rough writing with each part	July 12, 2018	Henry Yip
1.2	Assigned sections of plan for each group members	July 13, 2018	Everyone
2.0	Added “Guidelines”	July 13, 2018	Henry Yip
2.1	Added “System Diagram”	July 15, 2018	Faisal Atif
3.0	Modified “Guidelines” & “System Diagram”	July 15, 2018	Anmol Bajaj
3.1	Added “System Diagram”	July 16, 2018	Faisal Atif
3.2	Added “Data Requirements”	July 16, 2018	Henry Yip
3.3	Added “Feature Priority”	July 17, 2018	Everyone
4.0	Modified the report	July 18, 2018	Everyone
4.1	Formatted the file	July 18, 2018	Zhixin Huang
5.0	Changed to fit version 3 design.	July 30, 2018	Henry Yip

3. Development Guidelines:

The developers will abide the following technical and legal/ethical guidelines in the development process for EasyGlucose.

Technical guidelines:

1. Developers should use Xcode 9 as their main IDE to develop EasyGlucose.
2. Developers should use Git as their version control software.
3. Developers must write the source code for EasyGlucose in the programming language Swift.
4. Developers should follow the following naming style guidelines for code consistency

Subject	**Style**
Variable names	Lower case camel case Eg. camelCase
Method names	Upper case camel case Eg. CamelCase
Constants	All caps with underscores Eg. ALL_CAPS

5. Developers should use meaningful and descriptive words when following the above styling guideline to improve code readability.
6. Developers should reduce code complexity by modularizing functions.
7. Developers should produce concise and sufficient documentation for each file, function, class, and class method.

Ethical/legal guidelines:

1. Developers must not infringe existing copyrights when developing EasyGlucose or writing related documents.
2. Developers must not plagiarize existing information when developing EasyGlucose or writing related documents.
3. Developers must give credit to referenced work and external help where appropriate.
4. Developers should follow the IEEE code of ethics when developing EasyGlucose or writing related documents.
5. Developers should follow App Store requirements when developing EasyGlucose.

4. System diagrams

The **Log** Class

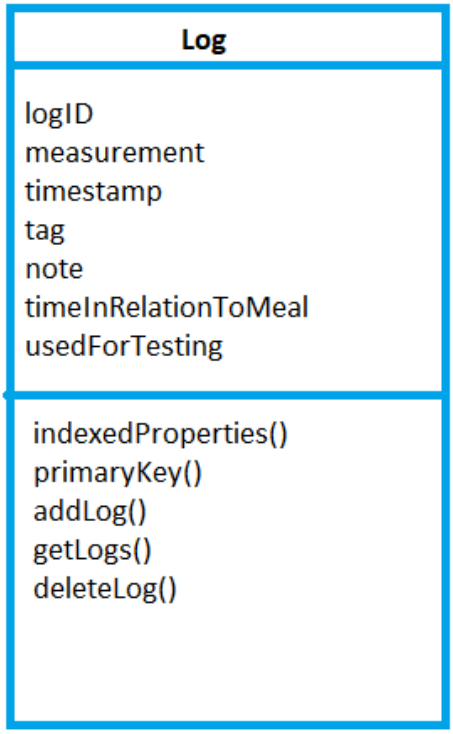

Figure 1.

This is a new class we have added in **Version 2** of our application. This **Log** class is the database of our application. The objects on top are created with Realm.io Mobile Database and used to store the data within the app. Below are the descriptions for each object and function.

Objects:

- logID: creates a random ID that is associated with each glucose log.
- measurement – used to store the glucose reading of the user.
- timestamp – automatically logs the date and time associated with each log
- note – used to store a note from the user
- timeInRelationToMeal – stores the fact if the log was before or after meal
- usedForTesting – this Boolean object is initialized true if the log is associated with automatic testing

Functions:

- indexedProperties() – in-built function from Realm, explicit declaration of a database
- primaryKey() – in-built function from Realm, the return value
- addLog() – adds the glucose measurement and writes it to memory
- getLogs() – converts the logs into an array and displays them
- deleteLog() – delete a log from the database

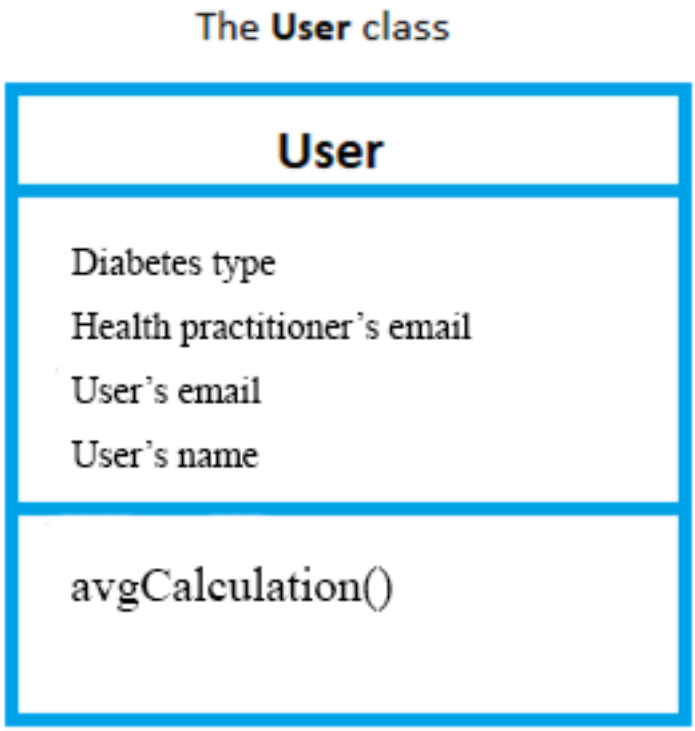

Figure 1.a

This class is fully implemented in version 3 with some changes from the original class and combining the previously named DiabetesData class into this one. The User class is defined in Figure 1.a. This is a critical component of our application as it handles all user details. The data we intend to store is listed on the first half and the methods for the class are listed on the bottom half. Below is the explanation for the purpose of each method we intend to create:

avgCalculation() – Function to calculate the average of blood glucose levels, carbs, and blood pressure.

Activity Model of the User Flow

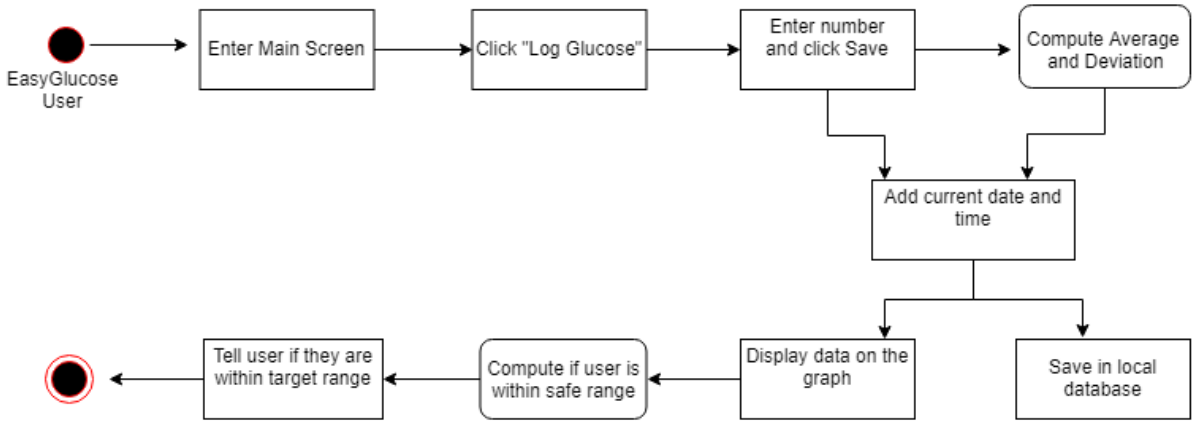

Figure 1.c

This activity model of the User Flow in Figure 1.c aims to visualize the typical actions that the user will take on the daily basis and how our application will react to their actions. The typical EasyGlucose user will want to log their blood glucose level as often as recommended by their physician or dietician.

They will start by clicking on the app icon on their iOS device and entering the main screen of our app. We plan to make the “Log Glucose” button large and minimize distractions before the glucose is logged. The user will enter the reading from their personal Diabetes Glucose meter into the text field in our application.

Our algorithm will then compute the average and range, including the newly input data. We will also associate their glucose data to the current data and time and save it to the local database. Our app will then call functions to visualize the data. If the user is within the safe range, we will display a “You are within range” message. However, if the user has abnormal levels of blood glucose, we will alert the user with a “You are not within range” message and prompt them to email the glucose report to their dietician and/or physician.

Sequence Diagram of Report Creation

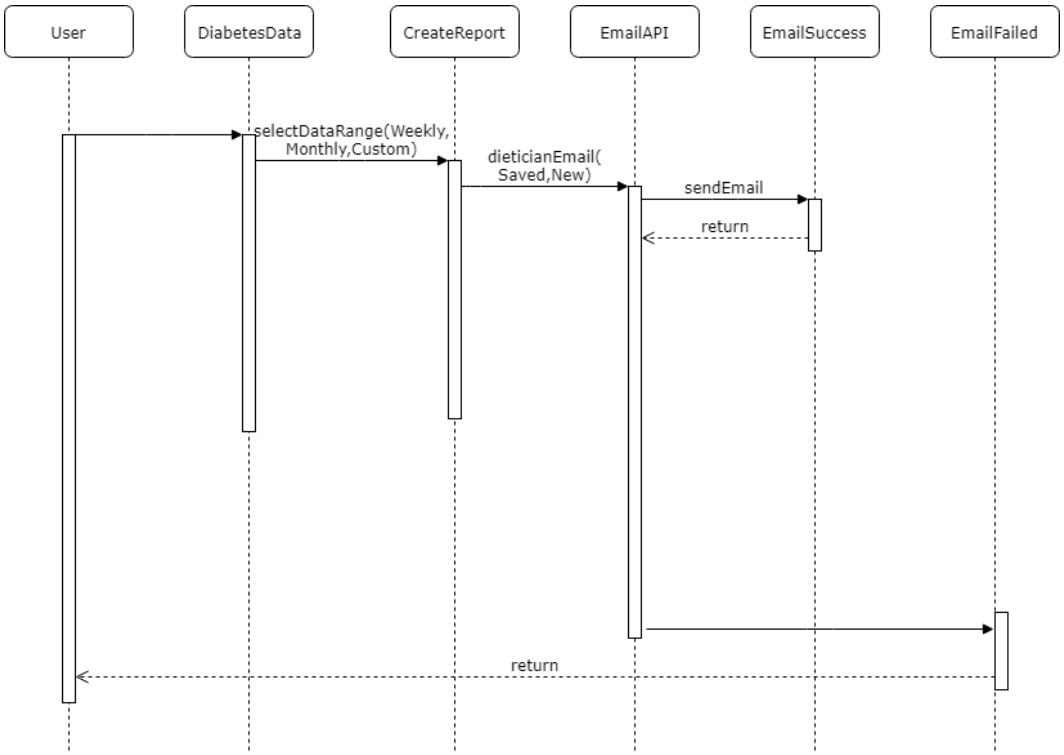

Figure 1.d

The purpose of Sequence Diagrams is to highlight interactions between users and the system and their system components. In Figure 1.d, we highlight the objects and methods that need to be concurrently running for the user to email their Blood Glucose Report to their dietician and/or physician.

This feature was not implemented into the app. The diagram starts with the User and continues to DiabetesData where the user is prompted to select the range of data that that they wish to include in their report. After they select ‘Weekly’, ‘Monthly’, or a ‘Custom’ range, the Create Report method is called. This method prepares the report in a PDF file format and calls the function send Email. If the email is valid and no network connection is lost, the email gets sent and EmailSuccess is called. However, if the network connection is lost during the time the email was being sent, the diagram reaches the EmailFailed state. This state returns the use back to the original window and prompts them to try again.

5. Data requirements

The main input method for EasyGlucose will be using the iPhone touchscreen, which provide software input methods such as built-in keyboard and drop-down selection forms. Other input methods will include built-in camera, access from system gallery, and get requests from the system clock. The application will collect data for purpose according to the following list of tables, with the list title being an app feature.

IO overview:

- EasyGlucose will use the following iPhone hardware for input: touchscreen display, microphone.
- Software input methods include built-in keyboard, swipe gestures, drop down selection forms, data access to system image gallery, and get requests from the system clock.
- EasyGlucose will use the following iPhone hardware for output: touchscreen display, on-board speakers, network card.
- The external system that EasyGlucose will interact is the Email system.

Detailed input and output methods in relation to features:

- The following list of tables are description of input and output methods based on features of EasyGlucose

Glucose level entries:

Type of data collected	**Input method and details**
Glucose level measurement	Built-in numeric keyboard
Time and date of entry	Automatically recorded.

Diary log with pictures:

Type of data collected	**Input method and details**
Title field	Built-in alphabetic keyboard
Time and date of entry	Automatically recorded.
Diary entry	Built-in alphabetic keyboard
Picture	Picture from built-in camera or from system image gallery
Related tags	Selection menu of before/after meal.
Notes	Operation initiated by tapping a “+” button.

Blood glucose level analysis graph settings

Type of data collected	**Input method and details**
Graph time-period adjusting command	Changed in settings page.
Graph time frame finetune	Selection menu with pre-entered time frames.
Diary entry access	Tap displayed diary entry thumbnail.

Profile initialization/modification

Type of data collected	**Input method and details**
Date of birth	Drop down menu with date options.
Type of diabetes	Selection menu via touch screen
Name	Built-in alphabetic keyboard
Email	Built-in alphabetic keyboard
Dietitian email	Built-in alphabetic keyboard

Scrollable blood glucose level entry table and diary entry table

Type of data collected	**Input method and details**
Table positioning commands	Vertical swipe gestures from touchscreen.
Entry view selection	Tap gestures from touchscreen.

Database Format Definition:

Data inputted for glucose level entries, diary log with database and profile initialization/modification will be saved using Realm database with format shown in figure below.

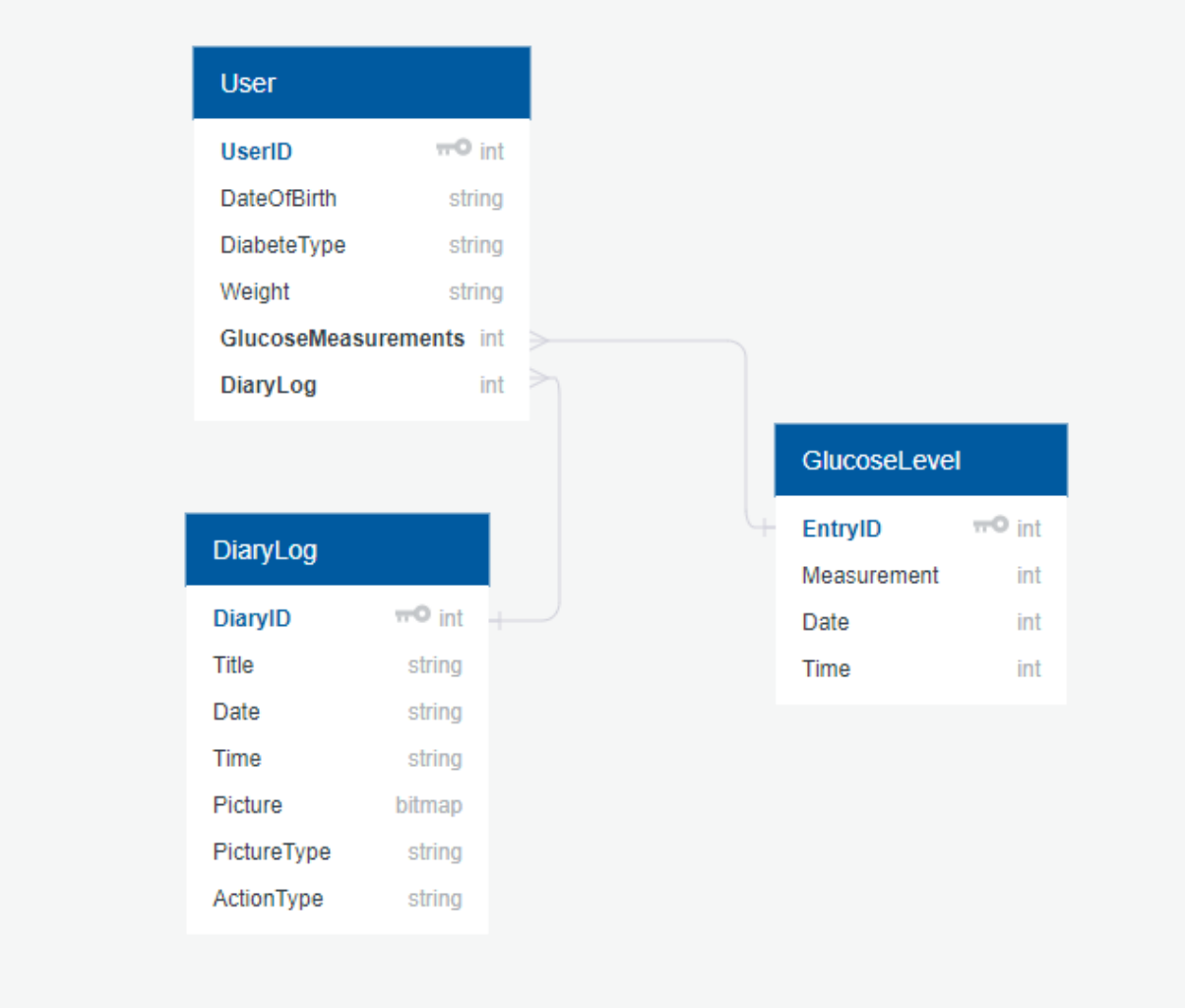

Figure 2. Diagram creation tool credit: QuickDBD

6. Feature priority

EasyGlucose shall provide users with these features that are ranked based on their priority. We are on track for version one and version two. We do not have the voice input implemented as planned for the version 2. We will be working on this on version 3.

Version 1:

- 1- Manual input: Allow the user to manually input their glucose levels into the app.
- 2- Database: Create a local database to store all the raw data and the tables/graphs after analysis.
- 2.5- Analysis: Analyze the raw data and add it back to the database.
- 3- Startup screen: Create a startup screen that asks the user for their type of diabetes, age, gender, and weight.

Version 2:

- 1- Document food: Allow the user to log and to take pictures of their food.
- 2- Log notes: Allow the user to log notes or add tags along with their glucose input.
- 3- Improve aesthetics: Improve the user interface and pick a colour scheme.

Version 3:

- 1- Create a settings page: Allow the user to update their weight and other personal information.
- 2- Unit conversion: The user can choose to use the imperial or metric system.
- 3- Create a profile page: The user will have the ability to customize their account more and add more information to it.
- 4- Change language: Give the user the option to change the application’s language to Chinese (Simplified).