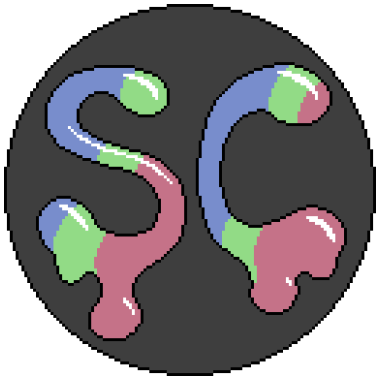
Sugar Coded



Software Design Document

February 7th, 2018

Project: Prediabetes Intervention Mobile Application

Sponsor: Dr. Natalia Dmitrieva

Mentor: Dr. Eck Doerry

Team Members:

Chantz Spears

Julian Shak

John Bassler

Alfonso Martinez

Version 1.2

Table of Contents

1.0 Introduction

Diabetes mellitus is a chronic condition characterized by elevated blood glucose levels. The reason for these elevated blood glucose levels is due to one's insulin, a hormone produced by the pancreas to filter sugar from your blood, not functioning properly. It is a disorder that currently affects over 30 million Americans. The vast majority of people with diabetes (~90%) have Type II diabetes. Those with Type II have grown resistant to their body's insulin over a long period of time (usually decades) and have a reduced capacity for blood sugar regulation. Causes of type 2 diabetes are often attributed to having a family history of diabetes, as well as a poor diet and exercise routine. Because of their high blood sugar levels, people with type 2 diabetes suffer from frequent hunger, increased thirst, fatigue, and blurred vision. Since diabetes is considered to be an irreversible condition, people with type 2 diabetes are left only to manage their condition for the remainder of their lives through various recommended strategies, including monitoring of blood sugar, carbohydrate intake, and medication use. However, type 2 diabetes prevention programs can successfully lower diabetes incidence.

One such program is the lifestyle arm of the Diabetes Prevention Program (DPP). This was designed to prevent future cases of diabetes by finding high-risk patients and offering moderate lifestyle and diet changes. These high-risk patients are usually overweight and have intermediate elevated blood sugar levels that do not yet meet the criteria for type 2 diabetes. This state describes the condition of prediabetes, and the US Department of Health and Human Services estimates that a third of US adults meet the criteria for prediabetes as of 2015. The DPP's goal is to reduce the patient's body weight by 7% and, in doing so, the DPP is successful in delaying and even preventing a future diabetes diagnoses. On average, a patient's risk of diabetes onset is reduced by 58%, as well as 45% of program participants self-identified as an ethnic or racial minority.

A problem these prevention programs are facing is the completion rates among their participants. Native Americans have the highest withdrawal rate from the DPP with approximately one-third prematurely dropping out of the 16-session program. Reasons for not completing the program or what can lead to someone dropping out are complex and hard to grasp. The current technique for finding out these reasons is through Ecological Momentary Assessment (EMA). In EMA, the methodology is to frequently ask the participants to record what they are feeling in the moment as a means for researchers to find . Since it is currently implemented through phone-surveys conducted by the researchers, EMA is expensive, time consuming, and intrusive for research participants. In order to improve retention, SugarCoded is deploying a mobile app and

web portal to aid with EMA research. This will allow surveys to be completed through participant's mobile phones and for researchers to prepare and deploy surveys to these participants via the web portal. By aiding researchers with our product, we can help increase the effectiveness of EMA research and in turn, help increase retention rates in diabetes prevention programs.

2.0 Implementation Overview

Team Sugar Coded has been working closely with Dr. Dmitrieva in order to design and implement a highly configurable mobile application and web portal. Figure 1, seen below, depicts a simple diagram of our solution, SCHEMA (Sugar Coded's Handheld Ecological Momentary Assessment). As displayed in Figure 1, we will be developing a web portal and mobile application that will have all of its central data stored in our backend database. You will also see that both the mobile application and web portal will be implemented using a server-client structure.

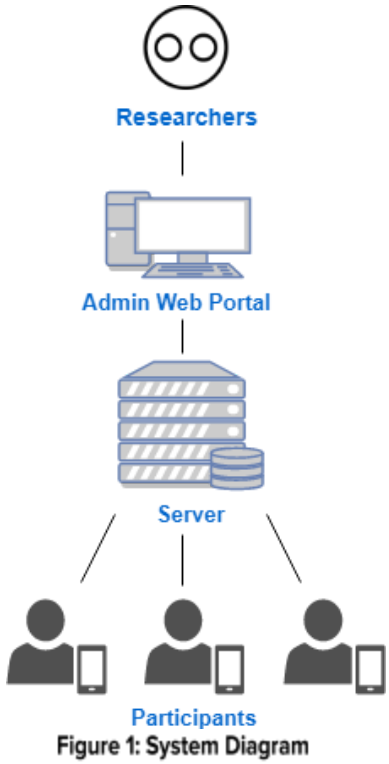


Figure 1: System Diagram

Our solution consists of two main components, we will go more in depth on the functionalities of both below.

1. **Web Portal**: The main functionality of the web portal is for the administrative owner of each study to configure the design and review the progress of each individual study. Configuration capabilities within the web portal will include study initialization with the subjects (participants) within each study as well as question and protocol creation. Each of these protocols will include different customizable settings to deploy to the research participants, such as different modules (questionnaires) and intervals in which these modules need to be taken. The web portal will be an easy way for the researcher to view the collected data in real time as the study progresses.
2. **Mobile Application**: The main functionality of the mobile application is to gather data from the research participants within each individual study. The mobile application that we are developing is designed to be a quicker, and cheaper alternative to gathering data compared to what is currently implemented; phone and paper surveys.

2.1 Tools & Technology

In the list below we go into more detail on each of the tools/technologies that we will be utilizing to implement this solution.

- Ionic Framework
    - Ionic is a complete open-source SDK for hybrid mobile app development. We chose Ionic as the framework to build both our web portal and mobile application because Ionic allows us to write an individual source code for both components that can be displayed on any device.
- Firebase Realtime Database
    - The Firebase Realtime Database is a cloud-hosted database. Data is stored as JSON and synchronized in realtime to every connected client. When you build cross-platform apps with our iOS, Android, and JavaScript SDKs, all of your clients share one Realtime Database instance and automatically receive updates with the newest data. We chose Firebase as our backend database because Ionic is highly supportive of this database and we are also interested in using Firebase's built-in authentication.
- Cordova Plugins
    - Cordova wraps your HTML/JavaScript app into a native container which can access the device functions of several platforms. These functions are exposed via a unified JavaScript API, allowing you to easily write one set of code to target nearly every phone or tablet on the market today and publish to their app stores. The cordova plugins that we are particularly looking into using are native plugins for local storage, push notifications, database communication, and any other native functionalities that we may be adding.

Section three will give a general overview of SCHEMA's architecture and how each component fits together.

3.0 Architectural Overview

Delving into a high level architectural overview of the project can provide a solid foundation for understanding how the system produces the desired behavior. In figure 2 featured below the systems four main components can be seen: the Mobile Application, the Web Portal, the Database, and the Wear Ware Portal.

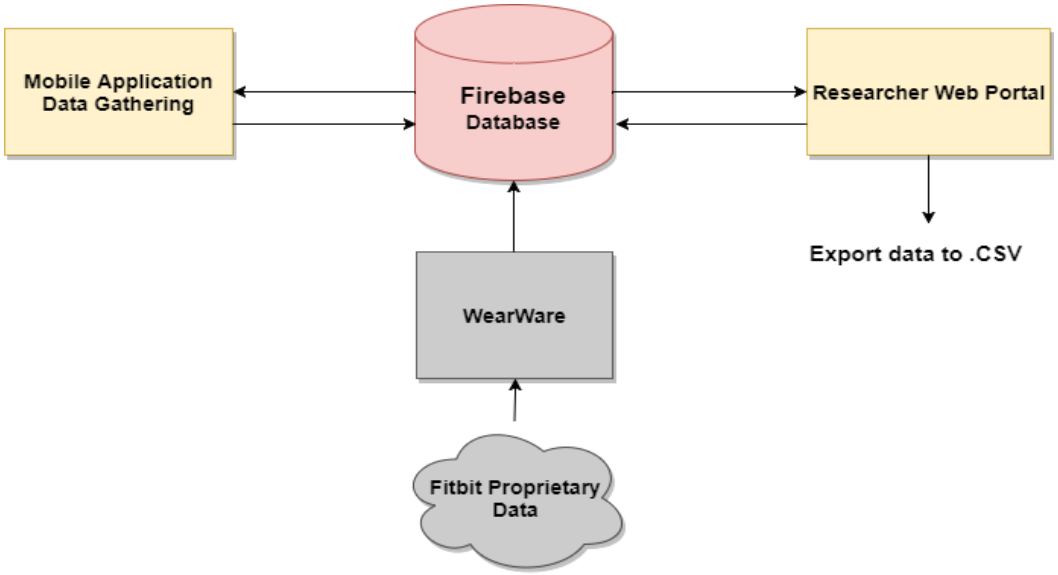


Figure 2: Architectural Diagram: The yellow components represent modules that Sugar Coded is building from the ground up. The red component is denoting the database that will link SCHEMA together and allow data collection and protocol administration between modules. The grey components are comprised of third party software provided by Dr. Kyle Winfree to collect participants Fitbit data.

Responsibilities and Features

The initial component that SCHEMA's users will interact with is the Web Portal. At a high level the Web Portal will allow researchers to:

- Create and publish studies
- Add participants to studies
- Create questions to be administered
- Control all time based protocols attached to the studies
- View study results
- Download results to a .csv

The Web Portal will control and provide all necessary functions a researcher would require to successfully administer a study. The Mobile Application will be what attaches study participants to a particular study and allows them to provide researchers with study results.

The Mobile Application will be utilized by study participants to:

- Attach themselves to studies
- Answer study questions (time based or user initiated responses)
- Receive study notifications

The Mobile Application will feature a question backlog section so if a user is unable to answer a time based question right away they can return at a convenient time to answer their questions. Starting the application for the first use will require participants to fill out a baseline questionnaire should the researcher desire and set up suitable times for the application to send them notifications.

The Wear Ware Portal will be used to gather participants' Fitbit data. Should a study issue participants Fitbits this portal will keep track of all Fitbit related data which can then be sent to the Web Portal for researchers to view.

The Firebase database will be the main connection for all of SCHEMA's individual modules, storing and pushing data between them when needed. The following section describes in finer detail the communication mechanisms the project utilizes that are heavily reliant on this centralized database.

Data Flow

Figure 2, above, shows how data will flow into and out of the database over the course of a study, the following describes a step by step of the process. After a study's creation its components are saved in the database and pulled into the Mobile Application. The answered questions in the Mobile Application and Fitbit data are uploaded to the database whenever a participant is connected to a network. These participant provided results are then delivered back to the researcher's dashboard on the Web Portal for their viewing needs. This is a cyclic process that continues through the duration of a study; questions are administered to the participants, they provide answers, and the researcher views the results. As mentioned earlier upon a studies completion the collected results may be downloaded to a .csv file.

SCHEMA's architecture is influenced heavily by the client server and component based architectural styles. Clients make requests to the server, where in this case the server is a database with application logic represented as stored procedures, that then directs the client how to proceed. SCHEMA is also sectioned into reusable functional and logical components that exhibit well-defined communication interfaces.
In section four an in depth view of each module and the roles associated with it are broken down, this will help provide greater clarity of the project in its entirety.

4.0 Module and Interface Descriptions

Section four, Module and Interface Descriptions, will provide an in depth overview of how each facet of SCHEMA will be composed, that is the classes each module will need, and the functions and variables within each class. The need and purpose of each class and how it furthers SCHEMA's abilities will be explained as well.

4.1 Web Portal Module

Below are UML diagrams depicting the different components of SCHEMA and the classes they are composed of. The UML diagrams are accompanied with a description of the components responsibilities and services each provide.

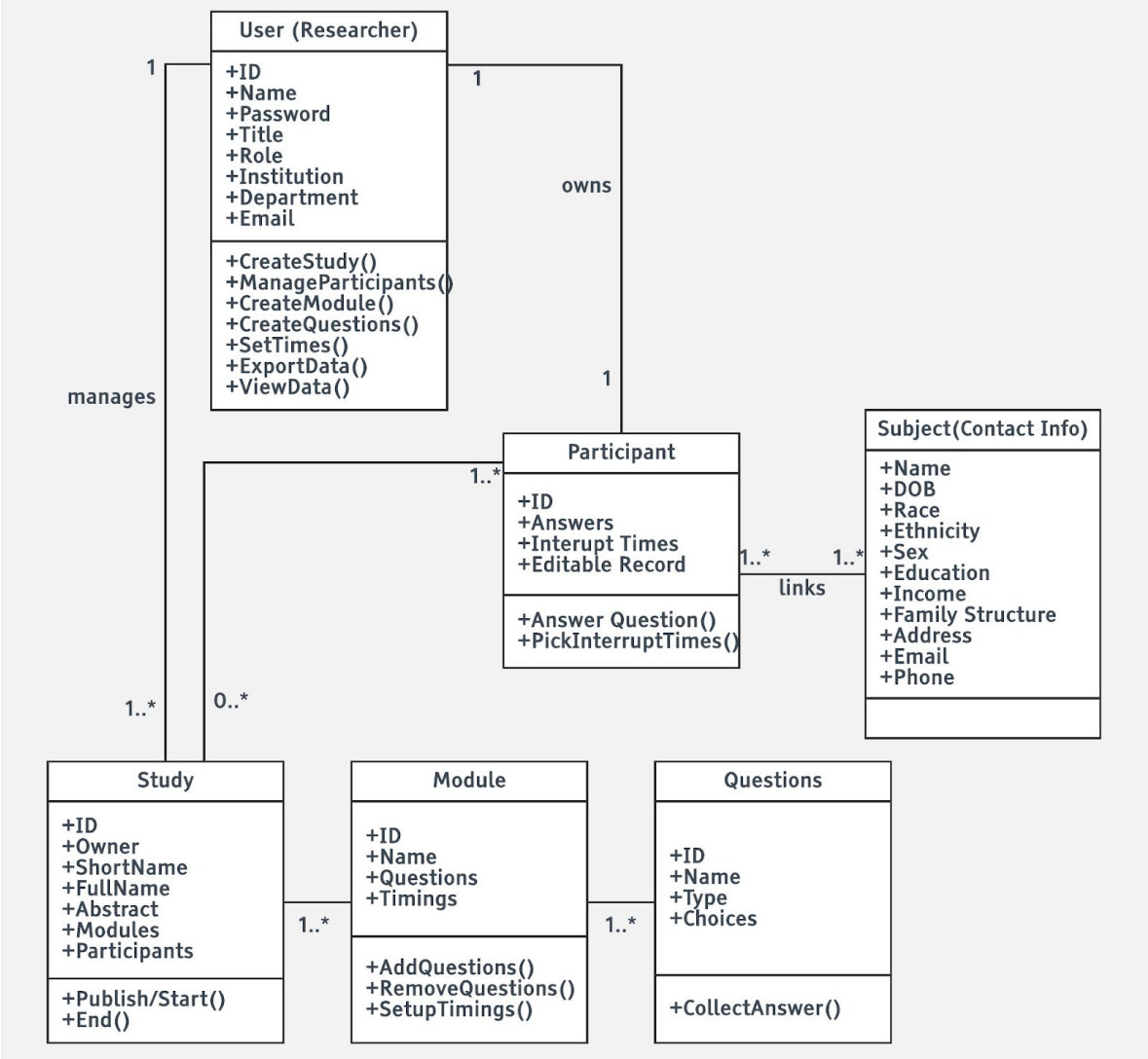


Figure 3: Web Portal UML

Figure 3 is a UML for the Web Portal module of SCHEMA which is composed of 6 actors/classes: User, Participant, Subject, Study, Module, and Questions. The Web Portal

will be the user's main hub for setting up studies and viewing the results. Below explanations will be given for any important variables and functions involved with the classes.

- User: This is the researcher or entity that will hold the most power in SCHEMA's hierarchy and the intended audience of Sugar Coded's product. Users will be able to create and manage multiple studies, view results in realtime, and export data after studies have concluded. The createStudy() function allows a user to set up a new study and fill in all the necessary information such as title, abstract, etc. ManageParticipants() allows a user to add, delete, or edit any participants information and change their status in the study. ViewData() allows a user to view all the currently gathered results from the study while exportData() allows the user to get the results in a .csv file. The createModule() and createQuestions() functions are what allows a user to customize a study to their liking. With these creation functions a user sets up what types of questions users will be asked, whether they will be notification based or self reported questions and with the setTimes() function they will be able to set the times for these questions to be asked. The following will describe any variables that are not implicitly inferred:
    - Title: How the user would like to be labeled i.e. Lead Researcher
    - Role: Whether or not the user is the owner of the study or just assisting with it.
    - Institution: The company or institute the user is associated with.
    - Department: The specific department within the users institution that they belong to.
- Study: A study is what the user creates and what participants are enrolled in. Studies have start/publish() and end() functions to mark when a study will begin pushing questions the participants and collecting data and when the study concludes and the participants are done answering questions. The study stores modules which control the protocols for questions, that is whether they are self reported questionnaires, or time based. The following will describe any variables related to a study that are not implicitly inferred:
    - Owner: The user in charge of this particular study.
    - Abstract: A description of what the study entails.
    - Modules: As mentioned above these are what hold the studies questions, protocols, and timings.
    - Participants: Studies keep track of which participants are associated with them.
- Module: Modules are able to addQuestions() to themselves or removeQuestions(). Adding a question allows a user to choose question type (multiple choice, slider, etc.), and if a time based question the user may use setupTimings() to schedule

when the participants should receive notifications to answer the question. Each module is individually set up by the user to either be a self reporting module or time based module, which then would only contain the corresponding questions.

- Questions: Questions are the actual data gathering tools setup by the user for participants to interact with. The Questions class will collectAnswers() of users after each their answers are submitted, sending them back to the database which the Web Portal will then pull from to display to the user. The following will describe any variables related to a study that are not implicitly inferred:
    - Type: Whether the question is requiring an answer in the form of a textbox, radio button, slider, etc.
    - Choices: If a question should be multiple choice for example the choices will be the options a participant has to choose from.
- Participant: These are who will be providing all the research data for a study via answering questions or providing information of their own accord. Participants will be able to answerQuestions() and pickInterruptTimes(). Picking interrupt times allows a participant to do things like set their typical sleep time so that no notifications will be sent to their mobile device in that time block (discussed further in the Mobile Application Module section below). The following will describe any variables related to a study that are not implicitly inferred:
    - Answers: All of the responses a user gives will be momentarily stored and sent to the database.
    - Editable Record: Each participant has a record that contains all of the demographic information they entered, they may change this at a later date should something change.
- Subject: This class is essentially all the identifying parts of a participant. Should a user whish to do a blind study simply hiding the subject class or breaking the link to participants will completely anonymize participants. The variables in this class are customizable as they could contain anything a user wishes to collect on participants.

The following section detailing the Mobile Application Module, goes in depth on the classes and functions that compose the mobile portion of SCHEMA and the goals it hopes to accomplish.

4.2 Mobile Application Module

SCHEMA's mobile portion will be the main form of communication study participants have that enables them to provide results to a researcher. Filling out self reported and time based questionnaires provide a variety of ways researchers can collect data and SCHEMA's notification system will help keep the participants participating.

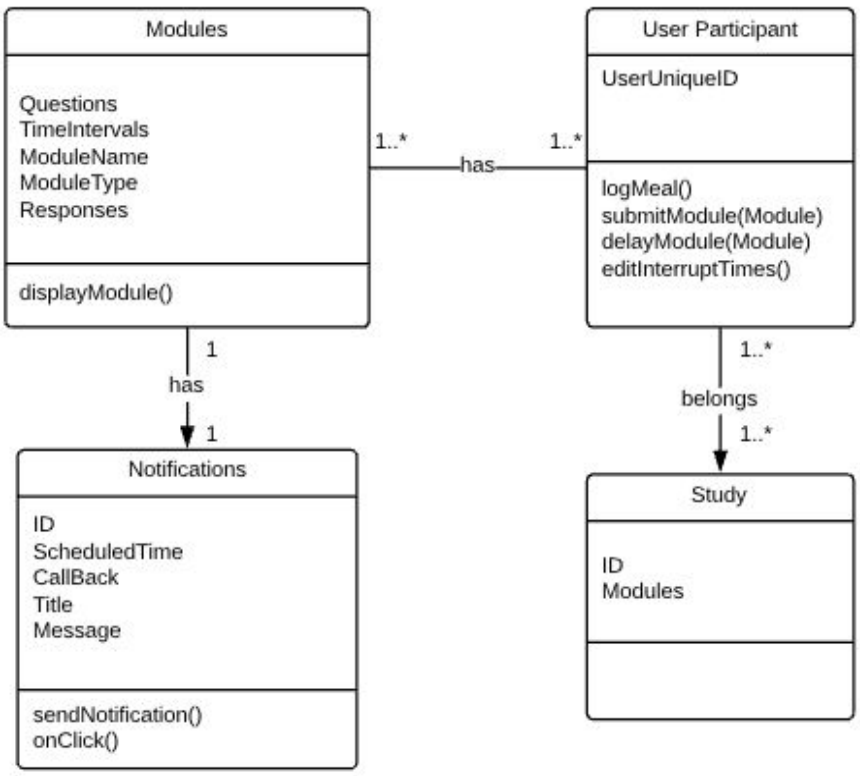


Figure 4 displays a UML diagram for the mobile application that we will be developing. There are four main actors/classes within our mobile application framework: Modules, Notifications, Users, and Studies. A study contains user participants which each have question modules associated to them. Each module in turn has researcher defined notifications associated with it. In the list below we go more in depth on the variables and functions that are associated with these four classes.

Figure 4: Mobile Application UML

- Module: an admin defined questionnaire/information gathering survey that is configured through the study admin web portal. Within this module class, our mobile application will be able to query data from our database and with the displayModule() function it will display the module according to the settings configured by the study owner. A module includes:
    - Questions: admin defined questions
    - Time Intervals: time intervals in which modules need to be completed
    - Module Type: self initiated or time interval
    - Responses: the user's response to questions in the module
- Notification: we will be utilizing a local notification system to alert the research participants of any modules that need to be completed. Upon initial login, our

mobile application will be gathering all of the time intervals in which each module needs to be completed and we will be storing these time intervals into the notification system of the mobile device. Our notifications will include two functions, the first function is sendNotification(), which will trigger based on the time intervals stored in the notification system. The second function is the onClick(), which is triggered once the notification is clicked on. This function will take the user directly to the correct module by looking at the variables associated to every notification listed below:

- ID: each notification will have an ID, which we will be using to associate different notifications to the type of module that they will be directed to upon click.
- ScheduledTime: this is the admin defined time interval in which this notification will be sent to the user.
- CallBack: this variable contains any additional information that we would like to associate to a notification to encapsulate any other information that will be sent to the module.
- Title: this is the bolded text at the top of the notification, with a short description of the action required.
- Message: this is a more detailed message to help the user identify which kind of module they are being requested to finish.

- User: a study participant is described in the above UML as a User. They are the main means of producing data for the researchers to collect through completing modules.
  - UserUniqueID: this generated ID will identify users while keeping their anonymity intact. This will also allow users to participate in multiple studies without creating new accounts.
  - logMeal(): the User uses the mobile app to complete a basic questionnaire for any meal they have during their time on the study. Some examples of information
  - submitModule(Module): upon completing a Module, a User can submit their completed Module through the mobile app to be stored in the database.
  - delayModule(Module): when the user receives a Push Notification to complete a module and they are currently unavailable to answer the Module (driving, receiving phone call, etc), they can delay answering the Module for a more convenient time.
  - editInterruptTimes(): a User will be allowed to edit the times the app will send notifications for their module. As each user will have different sleep

patterns, customizing interrupt times will allow for better user interaction for logging data through the app.

- Study: Studies are created by researches within the web portal. The Study has an ID and the Module(s) that execute while the study is active.
    - ID: each created Study will be given an ID that uniquely identifies it. An ID can be generated for a Study with no modules or study participants.
    - Modules: each study created by a researcher will contain one or more modules pertaining to their study participants (Users). Modules define the study, allowing the study creator(s) to edit and customize the questions to put in a module.

Next the Fitbit Module and its associated data will be examined.

Fitbit Module

The Fitbit Module will be what presents the researcher the data that was collected using Fitbit. The data will be collected from Dr. Kyle Winfree's Wearware Portal.

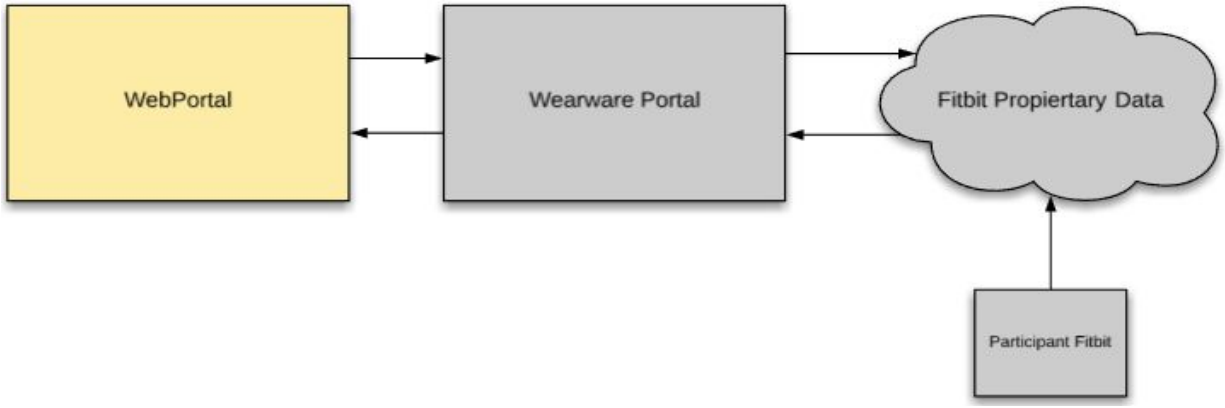


Figure 5: Fitbit diagram: The yellow component, the Web Portal, will be created as part of SCHEMA and be in direct interaction with the Wearware Portal to handle Fitbit data.

The WebPortal uses Wearware to collect data from Fitbit which comes from the participant. The data will be collected from the participant, which will then go to Fitbit's server. Wearware has access to Fitbit's data, which we will use in order to gather the physical and cardio data from the participant.

- WebPortal: The web application that a user will use in order to create and edit Studies. Within each study they will be able to pull information from Wearware in the form of .csv files.
- Wearware Portal: Dr. Kyle Winfree's web application that is using Fitbits api. Wearware receives updates from Fitbit, by receiving unique tokens. Wearware can use these unique tokens to pull data from Fitbit's server. It then stores this data in the background database. Our WebPortal will query from this database to show the researcher the information.

- Fitbit: Each participant will be given a Fitbit to wear during the study. The fitbit will record physical activity from the participant, which will be available to researchers. The data sets Fitbit stores are:
    - Heart Rate (bps)
    - Steps
    - Calories burned
    - Floors
    - Distance

Section five will give an overview on how SugarCoded plans on implementing the design from the prior sections.

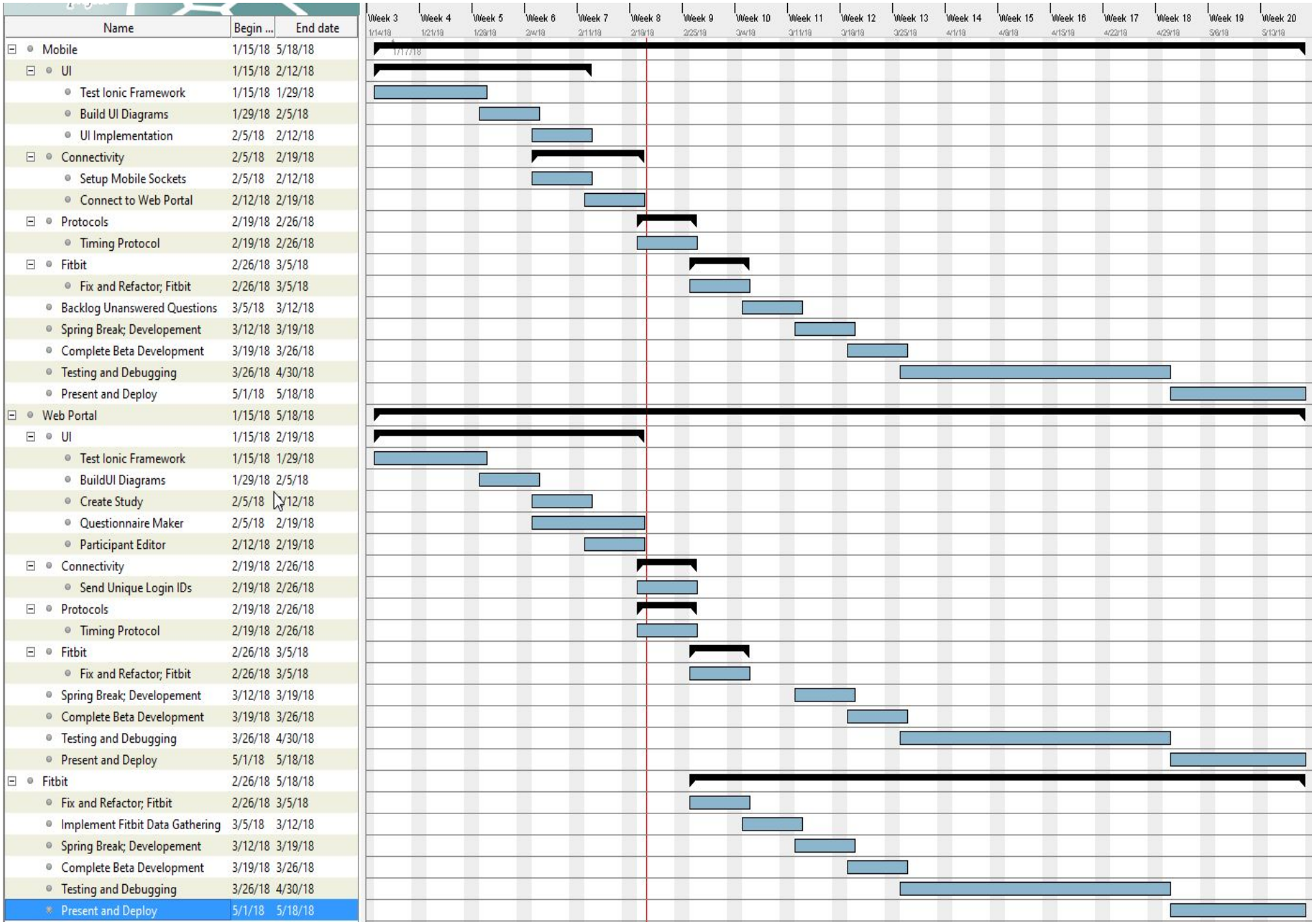
5.0 Implementation Plan

Next we take a look at our plan throughout the semester. The project will be split up into three different sections, the Web component, the Mobile component, and the Fitbit component. We will be working in pieces, breaking them down into smaller tasks that can be completed weekly. The first piece will be working on the UI for both web and mobile components. The mobile section will work on a basic user interface for participants to interact with. The web section will focus on being able to create studies and questions to populate the modules with.

The next piece will be Connectivity, joining the web and mobile components so that they are able to store data on the database. During this piece, we will begin creating and structuring the database that will be used by both web and mobile components. The web component will populate the questions and the mobile component will populate the answers for those questions in the database.

Once the components have been joined, the next piece will be implementing Protocols. The the timing protocols will be set into the web component. The researcher will create modules, which will hold a questionnaire and timings. The researcher can then add the module to their study. The module will send the questionnaire to the participant based on the timing that was chosen for it. The timings will be in the form of Day: (Once/ Weekly/Daily) Time: (xx:xx am/pm).

After the UI and backend has been implemented, Fitbit data will be added by using Wear Ware Portal to collect data from the participants. If Wearware is unable to provide the data, we will simulate Wearware into our application until a solution is found. Testing will be done once a beta version of the application is complete.


Name
Begin ...
End date
Mobile 1/15/18 5/18/18
UI 1/15/18 2/12/18
Test Ionic Framework 1/15/18 1/29/18
Build UI Diagrams 1/29/18 2/5/18
UI Implementation 2/5/18 2/12/18
Connectivity 2/5/18 2/19/18
Setup Mobile Sockets 2/5/18 2/12/18
Connect to Web Portal 2/12/18 2/19/18
Protocols 2/19/18 2/26/18
Timing Protocol 2/19/18 2/26/18
Fitbit 2/26/18 3/5/18
Fix and Refactor; Fitbit 2/26/18 3/5/18
Backlog Unanswered Questions 3/5/18 3/12/18
Spring Break; Developement 3/12/18 3/19/18
Complete Beta Development 3/19/18 3/26/18
Testing and Debugging 3/26/18 4/30/18
Present and Deploy 5/1/18 5/18/18
Web Portal 1/15/18 5/18/18
UI 1/15/18 2/19/18
Test Ionic Framework 1/15/18 1/29/18
BuildUI Diagrams 1/29/18 2/5/18
Create Study 2/5/18 2/12/18
Questionnaire Maker 2/5/18 2/19/18
Participant Editor 2/12/18 2/19/18
Connectivity 2/19/18 2/26/18
Send Unique Login IDs 2/19/18 2/26/18
Protocols 2/19/18 2/26/18
Timing Protocol 2/19/18 2/26/18
Fitbit 2/26/18 3/5/18
Fix and Refactor; Fitbit 2/26/18 3/5/18
Spring Break; Developement 3/12/18 3/19/18
Complete Beta Development 3/19/18 3/26/18
Testing and Debugging 3/26/18 4/30/18
Present and Deploy 5/1/18 5/18/18
Fitbit 2/26/18 5/18/18
Fix and Refactor; Fitbit 2/26/18 3/5/18
Implement Fitbit Data Gathering 3/5/18 3/12/18
Spring Break; Developement 3/12/18 3/19/18
Complete Beta Development 3/19/18 3/26/18
Testing and Debugging 3/26/18 4/30/18
Present and Deploy 5/1/18 5/18/18
Week 3 1/14/18
Week 4 1/21/18
Week 5 1/28/18
Week 6 2/4/18
Week 7 2/11/18
Week 8 2/18/18
Week 9 2/25/18
Week 10 3/4/18
Week 11 3/11/18
Week 12 3/18/18
Week 13 3/25/18
Week 14 4/1/18
Week 15 4/8/18
Week 16 4/15/18
Week 17 4/22/18
Week 18 4/29/18
Week 19 5/6/18
Week 20 5/13/18
1/17/18

6.0 Conclusion

Approximately 10% of Americans have diabetes mellitus (32.3 million), with the highest rates observed among individuals who self-identify as American Indian and Alaska Native (15.1%). A large body of research have examined effective approaches for reducing incidences of diabetes within this population. There currently is an effective sixteen-session program named Special Diabetes Program for Indians Diabetes Prevention (SDPI-DP) who successfully lower patient's risk of diabetes. However, the program's retention rate is suffering greatly with only 74% of participants remaining in the program by the sixteenth session, and 33% remaining by the three-year follow-up.

This gives us two questions that we attempt to answer though SCHEMA:

Q1: Why don't people complete the program?

- Reasons for dropping out are currently too numerous and ambiguous target and improve
- Using EMA, researchers can better identify when and what causes lapses in participant's discipline
- Create a means for more effective EMA research

Q2: What must we do to improve retention?

- Create a web portal for researchers to create and maintain a study based on the principles associated with EMA
- Allow researchers to collaborate and view data collected in the study
- Collect momentary data on various psychological, social, dietary, and physical activity experiences via mobile devices used by research participants

The work done in our Capstone will aid research related to psychosocial experiences and health behaviors among American Indians enrolled in SDPI-DP. Through raising retention rates within prediabetes prevention groups, our project aims to aid in reducing diabetes within Native American communities.