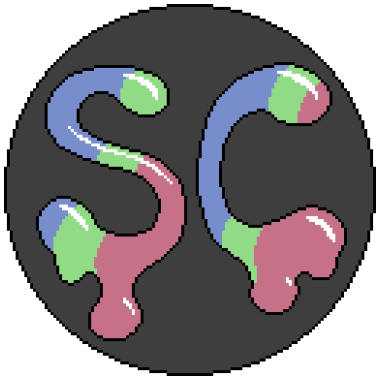
Sugar Coded



Software Testing Plan

April 1st, 2018

Project: Prediabetes Intervention Mobile Application

Sponsor: Dr. Natalia Dmitrieva

Mentor: Dr. Eck Doerry

Team Members:

Chantz Spears

John Bassler

Julian Shak

Alfonso Martinez

Version 1.0

Table of Contents

1.0 Introduction

Diabetes mellitus is a chronic and frequently irreversible blood condition that affects over 30 million Americans, and as of 2015 was the 7th leading cause of death in the US. There is a growing call to awareness of those who are at risk of becoming diabetic via Diabetes prevention programs to reduce the rate of cases. To keep these prevention programs successful across all affected ethnicities, it is imperative to identify reasons for participants to drop from programs. Dr. Natasha Dmitrieva, a developmental and health psychology researcher at Northern Arizona University, believes reasons for participants dropping out of programs can be found through Ecological Momentary Assessment (EMA), a research technique focusing on frequently asking participants to record what they are feeling in the moment in order to find disparities. Our team is tasked with creating SCHEMA (Sugar Coded's Handheld Ecological Momentary Assessment); a web portal for researchers to create EMA studies, and a mobile application the studies can be deployed on.

At this point in development to see if both the web portal and the mobile application are functioning as intended we have designed a software testing plan. The goal of our testing is threefold, to ensure each of our applications modules are functioning correctly with unit testing, to ensure interaction and data exchanges between are modules are correct, and to ensure a smooth end user experience with usability testing.

Due to SCHEMA being heavily geared towards the frequent use of end users, much of our testing will be devoted to usability. This document will go into detail on unit tests, how we will ensure the functionality of our projects functions, integration tests, how we check that all interactivity and data exchange between modules is accurate, and finally our most important piece usability testing. The usability section will detail plans for extensive user testing with Dr.Dmitrieva's research group along with nonspecific NAU students. Usability testing will include both moderated and unmoderated interaction testing. In-person usability testing will focus on user's ease of use and ability to complete questionnaires, while unmoderated long term testing will focus on user completion frequency for both automated and user initiated questions. From here, our unit testing methods and individual tests will be detailed in section two.

2.0 Unit Testing

To ensure SCHEMA's modules and procedures perform correctly the team has designed unit tests for each of our application's main functions. Since many of our functions make use of minor helper functions we opted to focus our tests on main subsets of our web and mobile applications. We have written the majority of our code in TypeScript and to help expedite our unit testing we will be leveraging the following tools, Mocha a testing framework, and Chai an assertion library. These tools both have first-class support for TypeScript which is our main motivation for choosing them, they will make writing unit tests far easier than without using any external libraries or testing functions. The modules listed below will be accompanied by tables providing the name of the module or function being tested, the equivalence partitions, boundary values, sample output, and expected output for multiple unit tests. To start off we will examine the web portal's tests before detailing the mobile application's tests later on.

Web Portal

Account Handling -
This module handles researchers logging into our web application, creating new accounts to log in from, and logging out. Creating a new account requires the user to enter a valid email address, and password of minimum length of 6 standard input english keyboard characters. To sign in if a user has already created their account they just need to enter their email and password upon entering our webportal.

Unit Test	Equivalence Partitions	Boundary Values	Sample Input / Action	Expected Output / Effect
Valid Log In	Any valid email and password stored in the database	Email:email@site.com Any valid email format Password: >= 6 characters & Ascii characters 33 -136	Email: css258@nau.edu Password: abc123	User's email and password checked against the database and they are let inside the app
Invalid Log In	Any input of email and password that is not stored in the database	Email: any string not containing '@' Password: < 6 characters & ! Ascii characters 33 - 136	Email: cats Password: 123	Message pops up telling user their email / password combination is invalid
Valid Account Creation	Validly formatted email Password which meets requirements	Email: any string containing '@' Password: >= 6 characters & Ascii characters 33 - 136	Email: css258@nau.edu Password: abc123	Stores new email and password combination into the database

Invalid Account Creation	Invalid email, Password which does not meet character and length requirements	Email: any string not containing '@' Password: < 6 characters & not Ascii characters 33 - 136	Email: cats Password: 123	Prompts user to enter a valid email and password
Log Out	Being logged into the app and pushing the logout button	N/A	Click logout button	User is logged off and taken back to login page

Participants -
A user can create a participant inside of our web portal who is then linked to the participant as an owner. When creating a participant a user has the option to add additional information but if they would just like to generate a unique user ID that is possible as well. Participants are updated very similar to how they are created and of course participants may be deleted as well.

Unit Test	**Equivalence Partitions**	**Boundary Values**	**Sample Input / Action**	**Expected Output / Effect**
Participant creation	Filling out optional information or leaving it blank	Blank strings or any character can be used for all the fields, which are optional	Clicking create user without filling out any information, which is optional	New participant is created in the database with current user as their owner and given a unique ID
Participant Update	Updating the optional information of a participant	Blank strings or any character can be used for all the fields, which are optional	Clicking update having made or not made changes to the participant's fields	The database is updated with the new or same information for the participant
Delete Participant	There is a participant stored in the database	N/A	Clicking the delete button next to the participant to delete	The participant is found in the database and removed

Study, Question, and Module Creation, Updates, and Deletion -
User's may add, update and delete numerous studies, modules, and questions within the web portal. When a user creates a new study, module, or question or updates an existing one all fields must be filled out with valid alphanumeric characters.

Unit Test	**Equivalence Partitions**	**Boundary Values**	**Sample Input / Action**	**Expected Output / Effect**
Valid Creation	Filling out all fields before completing creation	Any character can be used for all the fields except for dates which must be filled using given tool	Clicking create study having filled out every field	A new study is created in the database with included fields
Invalid Creation	Failing to fill out all fields before trying to complete creation	Empty strings	Leaving fields blank and clicking create study	User is notified all fields must be filled out
Update	All fields must still be filled out upon clicking update	Any character can be used for all the fields except for dates which must be filled using given tool	Clicking update study having made changes to fields	The database is updated with the new information for the study
Deletion	There is a study stored in the database	N/A	Clicking the delete button after selecting the study	The study is found in the database and removed

Adding Modules to Studies and Questions to Modules -
After a user has created all of their study's components they will want to combine them to get their study into a deployable state. Studies need to contain modules which in turn contain questions before the mobile application can deploy the study.

Unit Test	**Equivalence Partitions**	**Boundary Values**	**Sample Input / Action**	**Expected Output / Effect**
Valid Adding	Adding a module to a study which does not already contain it, or any question to a module	N/A	Clicking on the module or questions name inside of the component they are to be added to	The question or module is now contained in the corresponding module or study
Invalid Adding	Trying to add a module to a study which already contains it	N/A	Clicking on add module inside a study that already contains it	User is notified the study already contains that module

Branching -
After questions are added to a module their order and branching paths need to be determined. If the question a user is branching is a radio button they will need to provide branch paths for each possible choice rather than just one path for a text question.

Unit Test	**Equivalence Partitions**	**Boundary Values**	**Sample Input / Action**	**Expected Output / Effect**
Valid Branching	Filling out a path for every question inside the module	The path value needs to either be another question or "end" from a drop down menu which signifies the end of that questionnaire	Clicking on the add branching button after selecting a branch path for each option on the current question	The branch paths are added or updated within the database and the user is taken back one page to the module screen
Invalid Branching	Leaving branching paths blank	Leaving drop down fields empty	Clicking on the add branching button after leaving any dropdown blank	User is prompted to fill each field

Data Download -
The user will want their participants' answers and data downloaded to files so they can be analyzed and manipulated. This data can be downloaded at anytime after a study has began even if the data is currently incomplete, but they will be warned.

Unit Test	**Equivalence Partitions**	**Boundary Values**	**Sample Input / Action**	**Expected Output / Effect**
Downloading a completed module's or study's answers	After participants have completed a study a researcher	There are answers in the database for the study in question	Clicking on the download data button after participants have completed the study or module	A JSON (soon to be .csv) file is downloaded to the local machine containing the requested data
Downloading an incomplete study's answers	Participants have yet to complete the module or study's questions	None or few answers have been submitted	Clicking on the download data button prior to study or module completion	User is notified the module is incomplete, if they continue anyway a .JSON file is downloaded to the local machine containing the requested data

Mobile Application

Connecting to studies -
This module handles the validation of unique User IDs that are auto-generated by our database for each participant that is registered within a study. Upon downloading our app, end-users are presented with a login page that prompts them for the unique User ID, which should be distributed by the study administrator. Any input is accepted into this field, however only when a valid unique User ID has been entered will the end-user be connected to a study. Invalid User IDs will present the user with an "Invalid Login" alert.

Unit Test	**Equivalence Partitions**	**Boundary Values**	**Sample Input / Action**	**Expected Output / Effect**
Valid Log In	Any valid unique User ID that is registered to a study.	Any typed input	User ID: 1234	User's unique ID checked against database and let inside app

Invalid Log In	Any input that is not a unique User ID registered to a study.	Any typed input	User ID: 456	This ID is not a unique User ID registered to any study in our database, so an “Invalid Login” alert appears.

Notifications -

This module creates all of the notifications that will be associated with the Time-Initiated Modules (questionnaires) of a study. The Time-Initiated Modules that we are allowing the study administrator to configure are daily recurring Modules, such as “every 30 minutes”. Upon initial login, the mobile application will prompt the user to enter their approximate “sleep times”. These are approximate times in which this participant sleeps and wakes up every day. The notifications that we configure should not trigger between these times.

Unit Test	**Equivalence Partitions**	**Boundary Values**	**Sample Input / Action**	**Expected Output / Effect**
Create Time-Initiated Module (every 30 min)	N/A	N/A	A Module is created on the web portal and is configured to trigger every 30 minutes.	The mobile application will notify the participant to complete this Module every 30 minutes, except when they are sleeping.

Backlog -

This module of our mobile application is related to the notifications module above. For each Time-Initiated Module that is missed (swiping away/ignoring notification) will be added to a “backlog”, which is essentially a collection of all modules that are pending submission. All Time-Initiated Modules will be added to this collection when their associated notification is triggered and will not leave until the user completes the Module.

Unit Test	**Equivalence Partitions**	**Boundary Values**	**Sample Input / Action**	**Expected Output / Effect**
User Ignores/Misses Notification of Module	N/A	N/A	User does not click on notification to complete Module.	The Module that is associated with this missed notification will be populated into the “backlog” and the user will be able to view this list of missed Modules in our app.
User Completes Time-Initiated Module	N/A	N/A	The User either clicks on the notification or clicks on the Module in the “backlog” page of the app.	This Time-Initiated Module will be removed from the “backlog” and will no longer appear on the page with all the missed modules.

Answer Submission -

Our mobile application is completely functional offline. The only time that the user will need Internet access is at initial login. Once the user logs in for the first time, we do a download of the whole study and store everything necessary to run the study locally. The user will be able to complete all of their logs/modules while not connected to the Internet, because all of their answers will be kept local until they connect to Internet. When the user connects to Internet, our mobile application pushes all data to our database, since the last completed sync.

Unit Test	Equivalence Partitions	Boundary Values	Sample Input / Action	Expected Output / Effect
Offline Submission of Module	N/A	N/A	A Module is submitted when the user is connected to the Internet, by clicking the “Submit Module” button at the end of a Module.	The database will be updated in real-time with the answers to that module when the user clicks “Submit Module”.
Offline Submission of Module	N/A	N/A	A Module is submitted when the user is not connected to the Internet, by clicking the “Submit Module” button at the end of a Module.	The database will not be updated with the answers to the module until the user connects to Internet.

Successful completion of the aforementioned unit tests will prove that our key functions and modules for both our mobile application and web portal function correctly. We have made an effort to handle erroneous input or actions as well so that our applications can stand both malicious and misguided use. Section three below will focus on the interactions between our application’s components and ensuring correct data exchanges.

3.0 Integration Testing

This application will be using a one time connection that begins with the web portal and goes to the Mobile application and a one way connection from Mobile application to the web portal. The researcher creates a study that will be downloaded by the user by logging into application with their unique id. Once the study has been downloaded the user will begin answering modules, and the answers to that module will be sent to the Firebase database, where the researchers can access using the web portal. Another application that will connect to our web portal is a third party application called Wearware. A researcher will be able to query Wearware in order to get fitbit data for a

specific user. This next section will go into the different tests to make sure that data is passed around correctly.

Study Upload/Mobile Download - As mentioned previously, a researcher will create a study. The researcher will give the users their unique id to login into the study. Once the user has logined to study using their unique id they will download the study to their mobile device.

Integration Test	**Sample Input / Action**	**Expected Output / Effect**
Study Download; Participant Login	Create Study and participant; Have Participant login to study	Participant will be welcomed to study; Baseline Module will be given to Participant if Baseline Module exist

Answers Upload and Download - Users will answer modules throughout the study and those answers will be uploaded to Firebase. Researchers will have access to the answers of any module in the study.

Integration Test	**Sample Input / Action**	**Expected Output / Effect**
Answer Module; Download Answers of Module	Participant answers module; Researcher downloads the module answers	Researcher will receive a CSV file of the answered module

Ownership/Display Permissions - The web portal includes ownership of each study. This is to prevent from anyone seeing seeing certain studies without the correct permissions. Studies, Modules, Questions, and Participants all have ownership permissions, which means only those with the correct permissions will be able to see these specific collections.

Integration Test	**Sample Input / Action**	**Expected Output / Effect**
Study permissions	Create two Researchers, one creates the study; Have the other look through their own studies	Researcher without the permission will not be allowed to view that specific study

Wearware Fitbit Query - The study that will use this application will be using Fitbit to gather more data from the users. The data received from fitbit will include data such as steps, heart rate, distance walked, etc. Wearware will be used to get the data from Fitbit, and the web portal will query Wearware for that data.

Integration Test	Sample Input / Action	Expected Output / Effect
Query Wearware data	Researcher will send query from web Portal to Wearware	Researcher will receive a .CSV file with fitbit data

The above integration tests will ensure our data is being grabbed and transferred correctly between modules and our database as well as make sure all interactions between our modules are correct. Section four, Usability Testing, will discuss our plan for ensuring a smooth end user experience.

4.0 Usability Testing

Our goal with usability testing is to ensure users of both of our applications, the web portal and mobile application, can effectively use and access the functionality we have provided. We need to make sure that our interfaces are understandable and actually perform how the end users desire them to perform. Before any serious testing begins we are planning some rapid iteration sessions with Dr.Dmitrieva where we present her everything we have in its extremely rough state without any user interface optimizations. Dr. Dmitrieva will then hone in on facets of our applications we can change relatively quickly to better suit her needs or provide end users with a more understandable experience. After the application has been smoothed out some from these sessions we can jump into usability testing for the web portal and mobile application.

WEB PORTAL USABILITY TESTING

Before usability testing can begin for our mobile application, we will start by testing our web portal. The target users for the web portal are researchers hosting studies, so both Dr. Dmitrieva and her students will be interacting with the portal for us to gather their feedback. Usability testing for the web portal will begin with moderating Dr. Dmitrieva and her students' use of the portal.

To moderate over the web portal usability testing, we will provide Dr. Dmitrieva and her students with a list of tasks asking them to perform various tasks from our integration testing, such as creating an account, creating a study, and creating questions. These tasks will be described somewhat vaguely to avoid over-instructing the researchers. Some task examples would include:

- Register an account on the web portal
- Create a study titled "My Research Study" along with questions, and modules
- Add the modules into the study and questions into the modules

By moderating over the researchers completing tasks, we can ask for their feedback on how they liked or disliked interacting with the interface. How they would prefer pages oriented, how buttons are mapped, and how it may compare to software they have previous experience with. We would also encourage the testers to “break” our code in edge cases we may not be aware of, such as certain data values that researchers wouldn’t use or formatting rules that must be followed within scientific studies.

After moderated testing and continuous improvements to our product based on feedback we can begin letting testers use the products on their own in any way they see fit and report back to us any critiques or findings. Simultaneously we will be conducting mobile testing, detailed below.

MOBILE USABILITY TESTING

For unmoderated mobile testing, the user will be on a “dummy” study. This will be a dry run of hosting users on a study created in our webportal. The users will also be given a task sheet with intentionally general/vague objectives to complete to interact with the app. For example, they will only be prompted “Visit the main screen and log your most recent meal” as opposed given step-by-step instructions on how to select, begin, complete, and submit a user initiated log.

The users will be invited to the dummy study and given a code. The user will download the app and be accepted into the study via their generated code. After the account setup, the only information users will have will be on their task sheet. During their testing period, they will act as if they are research participants

For in-person testing, we will give the testing participant a few small goals and oversee how they interact with our UI. Having each of these user go through the registration process is too time-consuming and we wish to focus on the app functionality, so we will give the user a phone to use with a test account attributed to it. We intend to moderate users by prompting them to do a task within the app and record their reactions and comments. Some questions that could give much-needed input from the end users may include the following:

- Is it always clear how the app wants you to complete a question? Why or why not?
- On a scale of one to five, how likely are you to select a push notification as soon as it arrives?
- Are the user initiated questionnaires easy to access?
- Would you be interested in seeing the results of previous questionnaires?
- Are any pages hard to navigate to?

Usability testing is a major part of discovering how functional and usable our product is, and as such, our testing methodology should reflect how in-depth and robust our product should be. Through our usability testing methods of both in-person moderated testing

and hands-off user interaction over multiple days, we are confident both our mobile app and our web portal will have the usability required to assist the researchers and the research participants with their needs.

5.0 Conclusion

Using the detailed testing plan discussed above we will perform rapid iterative development based on feedback or results from the three categories of testing. We are confident in our unit and integration tests completing successfully so most of our iteration cycle time will be put towards making changes to better suit the users wants and needs based on the usability testing. Following this plan we are confident that we can refine our project into something even more usable than what we currently have and and can rest assured that our product will perform as expected.