**Software Design**

**Document**

**for**

**Smart Eye Drops**

**Version 2.1 approved**

**Prepared by**

**Marco Aguilar, Aaron Alvarez, Ana Cortes, Jonathan Nunez,**

**Feng You, Ivan Yu**

**Vodafone**

**March 18, 2020**

Table of Contents

[Revision History 4](#_Toc39847869)

[1. Introduction 5](#_Toc39847870)

[1.1 Purpose 5](#_Toc39847871)

[1.2 Intended Audience and Reading Suggestions 5](#_Toc39847872)

[1.3 System Overview 5](#_Toc39847873)

[2. Design Considerations 6](#_Toc39847874)

[2.1 Assumptions and Dependencies 6](#_Toc39847875)

[2.2 General Constraints 6](#_Toc39847876)

[2.2.1 Hardware Constraints 6](#_Toc39847877)

[2.2.2 Software Constraints 7](#_Toc39847878)

[2.2.3 End User Environment 7](#_Toc39847879)

[2.3 Goals and Guidelines 7](#_Toc39847880)

[2.4 Development Methods 7](#_Toc39847881)

[3. Architectural Strategies 8](#_Toc39847882)

[4. System Architecture 9](#_Toc39847883)

[5. Policies and Tactics 11](#_Toc39847884)

[5.1 Choice of which specific products used 11](#_Toc39847885)

[5.2 Plans for ensuring requirements traceability 11](#_Toc39847886)

[5.3 Plans for testing the software 11](#_Toc39847887)

[6. Detailed System Design 12](#_Toc39847888)

[6.1 Data Transmission (Module) 12](#_Toc39847889)

[6.1.1 Responsibilities 12](#_Toc39847890)

[6.1.2 Constraints 12](#_Toc39847891)

[6.1.3 Composition 12](#_Toc39847892)

[6.1.4 Uses/Interactions 12](#_Toc39847893)

[6.2 App (Module) 12](#_Toc39847894)

[6.2.1 Responsibilities 12](#_Toc39847895)

[6.2.2 Constraints 13](#_Toc39847896)

[6.2.3 Composition 13](#_Toc39847897)

[6.2.4 Uses/Interactions 13](#_Toc39847898)

[6.3 Server (Module) 13](#_Toc39847899)

[6.3.1 Responsibilities 13](#_Toc39847900)

[6.3.2 Constraints 14](#_Toc39847901)

[6.3.3 Composition 14](#_Toc39847902)

[6.3.4 Uses/Interaction 14](#_Toc39847903)

[6.4 Machine Learning (Module) 14](#_Toc39847904)

[6.3.1Responsibilities 14](#_Toc39847905)

[6.3.2 Constraints 14](#_Toc39847906)

[6.3.3 Composition 15](#_Toc39847907)

[6.3.4 Uses/Interaction 15](#_Toc39847908)

[7. Detailed Lower level Component Design 16](#_Toc39847909)

[8. Database Design 17](#_Toc39847910)

[9. User Interface 18](#_Toc39847911)

[9.1 Overview of User Interface 18](#_Toc39847912)

[9.1.1 Patient Interface 18](#_Toc39847913)

[9.1.2 Doctor Interface 18](#_Toc39847914)

[9.2 Screen Frameworks or Images 18](#_Toc39847915)

[9.3 User Interface Flow Model 19](#_Toc39847916)

[10. Requirements Validation and Verification 21](#_Toc39847917)

[11. Glossary 22](#_Toc39847918)

[12. References 23](#_Toc39847919)

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Date | Reason for Changes | Version |
| Feng You, Marco Aguilar | 11/24/19 | Initial creation | 1.0 |
| Ivan Yu, Jonathan Nunez | 02/26/20 | First revision | 2.0 |
| Aaron Alvarez, Ana Cortes | 03/18/2020 | Final Revision | 2.1 |

# 1. Introduction

## 1.1 Purpose

This document outlines the Software Design Specifications as part of the design plan and specifications for developing Smart Eye Drops.

This document goes over the application in its entirety, including the software and hardware components described by the features in the Software Requirements Specifications (SRS) v2.0. Each feature discussed will describe the existing functionality of Smart Eye Drops.

## 1.2 Intended Audience and Reading Suggestions

This document is intended for software developers, project managers, users, testers, and documentation writers. The rest of the SRS will go over the application’s structure, business logic, database design, and implementation. It will also go over the data flow of the system to help future developers gain a better understanding of the application. Those who wish to have a better understanding of the tools used in the Smart Eye Drops system, such as the MMC and AWS, should refer to the documentation provided in the Glossary and References section.

## 1.3 System Overview

Sensors are placed on an eye drop container to collect live data from the patient who is applying the eye drop. The sensors collect raw accelerometer, gyroscope, and pressure data, which is then sent to the backend of the system. The backend system will then clean up the data and use machine learning to determine if an eye drop was applied or not. Once the data has been classified it will be sent to the cloud service and stored in a database. The cloud service also provides specific endpoints that will be accessible to the mobile application. The mobile application provides the main user interface for both the physician and patient. The app will allow physicians to view their patient’s adherence rates and adjust their patient’s regimen if needed. Another feature of the mobile application will allow patients to record the data when they apply their eye drops.

# 2. Design Considerations

Research on the necessary hardware components and the assembly of these components need to be performed to a significant extent to ensure that we have a working prototype of Smart Eye Drops before any coding begins. This is done to prevent spending a considerable amount of time creating software for a machine that may not work as intended.

## 2.1 Assumptions and Dependencies

The dependencies of Smart Eye Drops include, but are not limited to, the Android mobile operating system, Amazon Web Services, MBientLab sensors and supporting software, Interlink FSR 406, Python, and a few machine learning algorithms. This application was built under the assumption that the user will not misuse the eye drop container in any way that may cause the system to record false information. It is also assumed that AWS will be able to provide any backend functionality needed to maintain and improve any features or requirements of the Smart Eye Drops application.

## 2.2 General Constraints

The system will be dependent on at least one of the following factors to ensure successful operation of Smart Eye Drops. These are Hardware Constraints, Software Constraints, and End User Environment.

### 2.2.1 Hardware Constraints

* Force Sensitive Resistor: the FSR, or pressure sensor, needs to be flexible and large enough to wrap around a generic eye drop bottle and can be attached to a main sensor module to which it will pass pressure data. Software is designed to use pressure data to ensure that eye drops were applied by the squeezing of Smart Eye Drops.
* Main sensor module: the main sensor module will need an independent power source and have wireless data transmission functionality. Software is designed to send data recorded wirelessly to a mobile phone, which is then relayed to cloud storage whilst requiring minimal processing to prolong battery life.

### 2.2.2 Software Constraints

* The software to be build should take advantage of open source libraries and supporting software, such as sensor manufacturer APIs and cloud databases, unless an adequate open source product is not available for use.

### 2.2.3 End User Environment

* Internet Connection: if internet connection is not available during time of eye drop application, then the software will store relevant information to send to cloud storage when internet connection is available.

## 2.3 Goals and Guidelines

One of the main goals and priority for the Smart Eye Drops is to collect data from the eye drop container with minimized noise. This is an important aspect of the system since the collected data will be used in conjunction with some machine learning algorithms to determine if the user is applying the eye drops or not. Another goal the application is to classify the data retrieved from the sensors in order to send the correct information to the physician. This is crucial because it will give the physician valuable information about patient’s behavior. The information can then be used by the physician to adjust the patient’s regimen or eye drop dosage.

## 2.4 Development Methods

The Waterfall Development method will be used for the development of Smart Eye Drops. However, at the time of creation of this document, we were only beginning the Design stage of the Waterfall Development method due to the significant amount of time spent on researching and building the hardware necessary from the ground-up.

# 3. Architectural Strategies

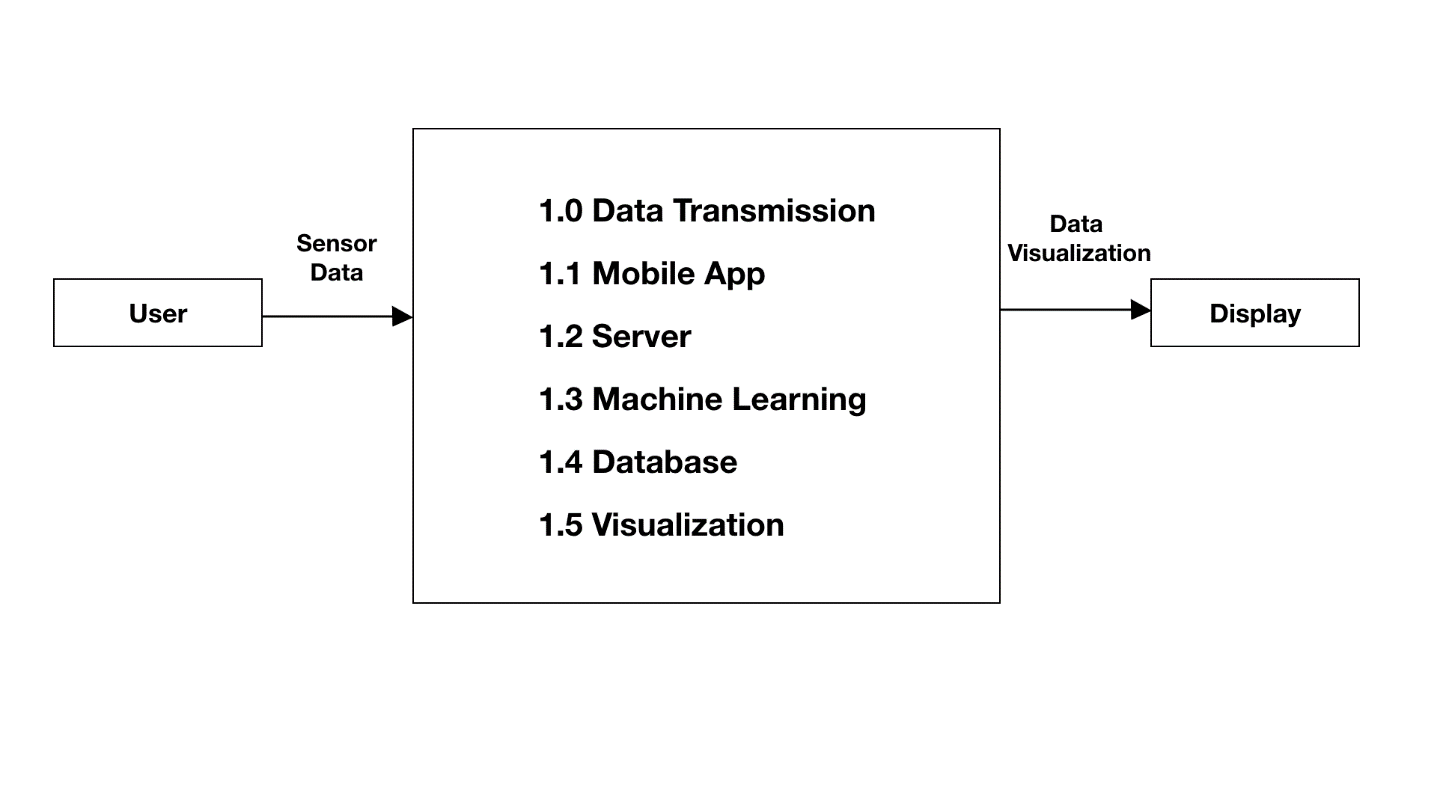
The machine learning aspect of the Smart Eye Drops system will be implemented using the Python programming language. This design decision was made because the Python programming language provides all the necessary libraries needed to work with various machine learning algorithms. There are other programming languages that can also be used to incorporate machine learning, such as Java, C++, JavaScript, and R. However, Python is very simple to use and has a lot of support from the machine learning community.

Another design decision that was made was using Amazon Web Services since it provides many services, such as storage and machine learning capabilities. There are other technologies that provide these two features, but AWS also allows one to create lambda functions, API endpoints, and gives access to a server via EC2. The lambda functions execute code in response to events like data being stored or processed. These functions provide an easier way to handle events in the backend. Overall, AWS will supply our system with the functionality needed to allow all our modules to communicate properly.

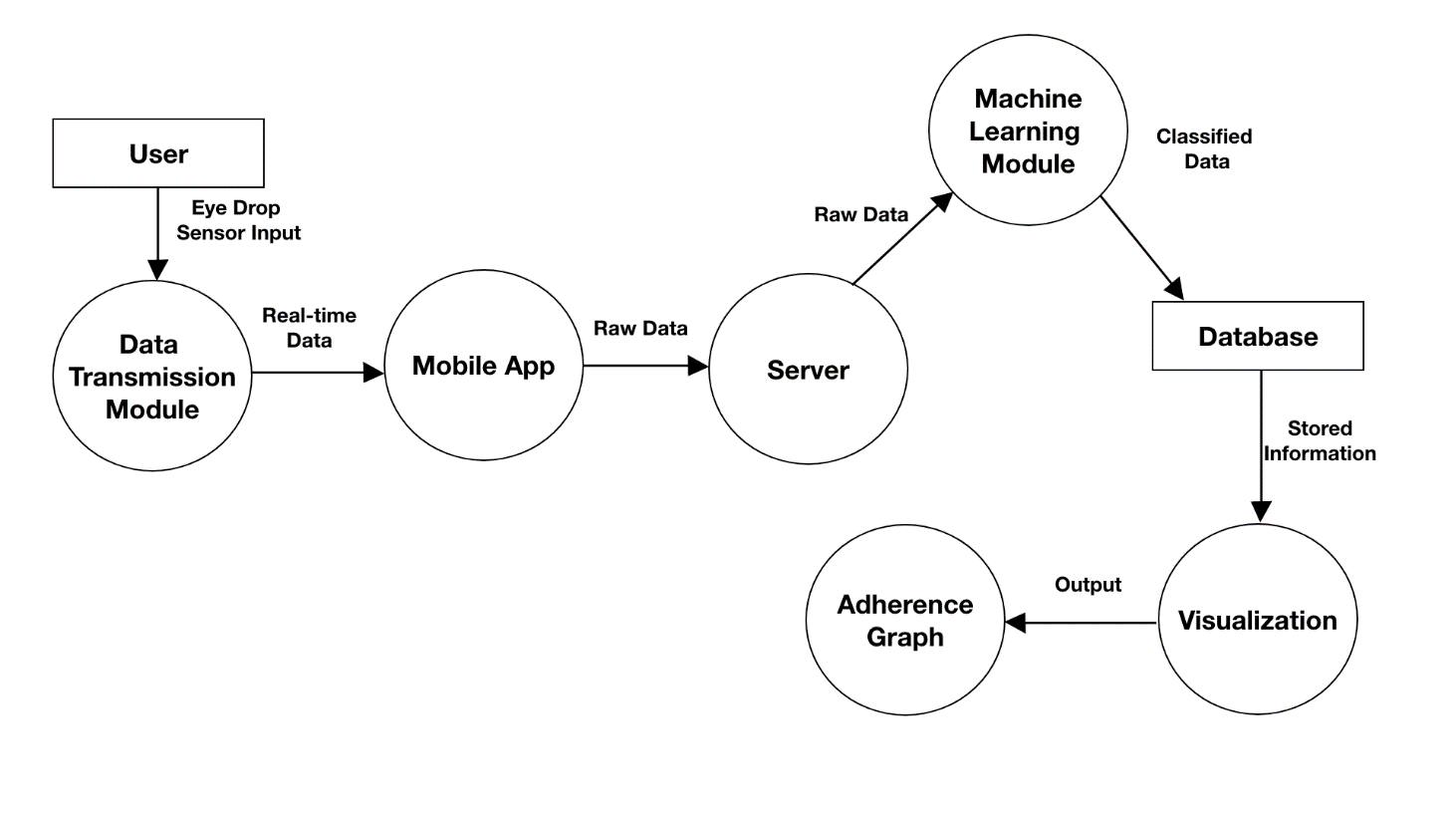
# 4. System Architecture

Our system will gather sensor input that is collected from the interaction between a user and an eye drop container. The sensor data will then be sent over to the set of modules that make up the main functionality of the Smart Eye Drops. From the set of modules, the adherence analytics will be sent over to the physician and displayed via the mobile application, also known as the display.

**DFD 0**



**DFD 1**



# 5. Policies and Tactics

This section includes both hardware and software design policies and/or tactics that do not have sweeping architectural implications, but which nonetheless affect the details of the interface and/or implementation of various aspects of the system.

## 5.1 Choice of which specific products used

**Interlink FSR 406**

The Interlink FSR 406 force sensitive resistor was chosen by virtue of its physical flexibility and pressure sensitivity range. By attaching it to MBientLab MetaMotion C, our main sensor module, we can retrieve a reading of a physical pressure applied to it. Interlink FSR 406 can read force with high accuracy, provided that adequate tuning is performed using software. However, the tactic of leaving the reading as is was implemented since the precise value of force applied is unnecessary in Smart Eye Drops. The sensor has fulfilled its function by verifying that the bottle was squeezed.

# 5.2 Plans for ensuring requirements traceability

Interlink FSR 406 is a hardware component so there will be no on-going refinement.

## 5.3 Plans for testing the software

Testing will be done by connecting Interlink FSR 406 to MMC and checking to see if a reading can be retrieved. Software must be modularized so that debugging is kept at a minimum. The cloud service also provides their own set of tests so that the software for each service can be tested before placed into production

# 6. Detailed System Design

## 6.1 Data Transmission (Module)

### 6.1.1 Responsibilities

This component oversees transferring all the information collected from the sensors. The raw data will be collected by the mobile application.

### 6.1.2 Constraints

One of the constraints of this module is the fact that it relies on a coin cell battery to perform its function. Another constraint of this component is that it will collect raw data. The raw data that is collected will need to be cleaned up and preprocessed before it is sent to the server module to be processed.

### 6.1.3 Composition

This component is composed of MbientLab’s MetaMotion C sensor, a coin-cell battery, an eye drop container, and a pressure sensor. The coin-cell battery is used to power the MetaMotion C sensor and the pressure sensor will be connected to the MMC to transfer data. The eye drops container will be used to attach all the sensors to it.

### 6.1.4 Uses/Interactions

This component will be used in conjunction with the main app module. The app module is responsible for collecting the raw data from the sensors and then transmitting that data over to the EC2 server.

## 6.2 App (Module)

### 6.2.1 Responsibilities

This module is responsible for receiving the data that is collected from the data transmission module and relaying that information to the cloud service. This module is a crucial component that will be used to transfer data from the sensors to the server module. This module is also responsible for making requests to AWS API Gateway and retrieving the data that was classified by the machine learning module. Once it receives the information it will then display it in a chart to show the patient’s eye drop adherence.

### 6.2.2 Constraints

This module will be built under the assumption that it will be able to handle all the traffic that passes through it. One of the limitations of this module is that it is only meant to send and receive data from other modules. This means that the data storage of this module is volatile and will need to handle any errors to prevent any data from being lost during communication. Another constraint of this module is that it is built for the Android OS platform. This means that phones with the iOS platform won't be able to use this module.

### 6.2.3 Composition

This module is made up of the Android operating system and the software libraries required to accurately receive the raw data coming from the sensors.

### 6.2.4 Uses/Interactions

This component will be used by all other modules to allow data to flow throughout the whole system. This entity will be used by AWS, which will be responsible for classifying, storing, and transmitting the data. It will also be used by the visualization module to obtain the data that was classified in the machine learning module.

## 6.3 Server (Module)

### 6.3.1 Responsibilities

This module is responsible for receiving the data from the App module. The data that is received is then processed into features. These features are then sent over to the machine learning module for classification.

### 6.3.2 Constraints

One of the constraints is that it can process data one at a time

### 6.3.3 Composition

This module is made using Java with sockets. The Java program will run as a server and run a python script to convert the data it received into features.

### 6.3.4 Uses/Interaction

This component will be used to collect raw data and process it into features and send those features to the machine learning module.

## 6.4 Machine Learning (Module)

### 6.3.1Responsibilities

The machine learning module is accountable for preprocessing and classifying the real-time data that is sent to it. Once the data has been properly categorized, the machine learning module will send the information over to the database module.

### 6.3.2 Constraints

One of the constraints of this module is the processing power used to manipulate the real-time data. This module is also limited by the size of the data gathered to train the data. If the data size is not large enough it’s prediction won’t be as accurate.

### 6.3.3 Composition

This module is made with python using software libraries to create a neural network and update the database.

### 6.3.4 Uses/Interaction

This module will communicate with the database module and the server module. This module will predict the type of application based on the features provided by the server module. If it was predicted to be an actual use, then the database module will be updated.

# 7. Detailed Lower level Component Design

**Physician Profile**

The physician’s profile uses an Android Activity to display the physician’s information and a list of patients that he/she oversees. The patient items are displayed with the help of a RecyclerView, where each patient’s data is retrieved from the AWS cloud service.

API Gateway is responsible for returning patient and physician information to the mobile application once the appropriate endpoint is reached. The physician’s profile Activity makes a GET request to the physician endpoint on API Gateway, and the lambda function responsible for retrieving data from the database executes. Once the data has been retrieved from the database, the lambda function packages the data in the form of JSON and sends it the API Gateway. Finally, the JSON data will be sent to the client as a response.

**Patient Adherence Graph**

The patient adherence graph uses an open-source library called MPAndroidChart to display the adherence analytics for a patient. More specifically, the Smart Eye Drops application uses a scatter chart and populates the graph by requesting data from the cloud service. The request is sent to an endpoint on API Gateway via a GET request.

Once the mobile application retrieves all the records for a user, it then stores it locally in a HashMap. The HashMap is then used to retrieve data points for a month of the year. The physician can then use the navigation buttons to view the adherence analytics for other months of the year.

# 8. Database Design

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Date and Time | Force Applied | X-axis | Y-axis | Z-axis |
|  |  |  |  |  |
|  |  |  |  |  |

The database will have five columns; “Date and Time,” “Force Applied,” “X-axis,” “Y-axis,” and “Z-axis.”

Date and Time records the time at which the data was recorded.

Force Applied ensures that Smart Eye Drops was used.

X, Y, and Z axes record the orientation of Smart Eye Drops at a given time.

|  |  |  |  |
| --- | --- | --- | --- |
| Physician Id | First Name | Last Name | Patients |
|  |  |  |  |
|  |  |  |  |

This database table keeps track of all the registered Physicians. The primary key for the table is the physician id, which contains integers. It also consists of the physician’s first name, and last name which are string values. Finally, the database table has a list of patients that the specific physician oversees.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Patient Id | Email | First Name | Last Name | Image | Physician Id | Records |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

This database table keeps track of all the registered Patients. The primary key for this table is the patient id, which holds integers. It also consists of string values for the patient’s email, first name, last name, and URL link to their image. The table also has a column for the id of their physician, which is an integer value. Finally, it contains a section dedicated to storing the patient’s adherence records.

# 9. User Interface

## 9.1 Overview of User Interface

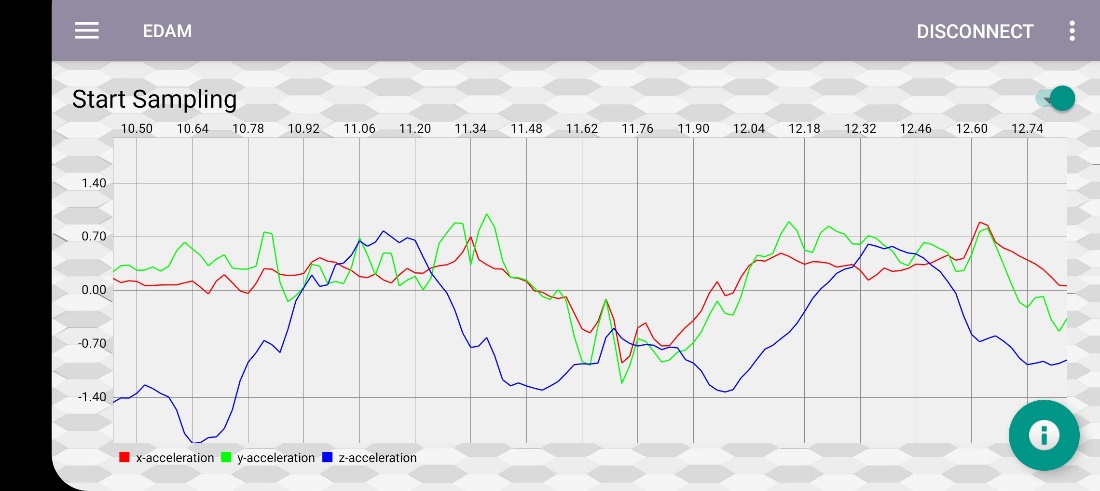
### 9.1.1 Patient Interface

The mobile application will have a simple layout for the patients which shows them times at which they are expected to apply their medication, missed application times, and a button that allows them to connect their mobile phones to Smart Eye Drops. The mobile application will also send reminders to patients when it is time to use their medication.

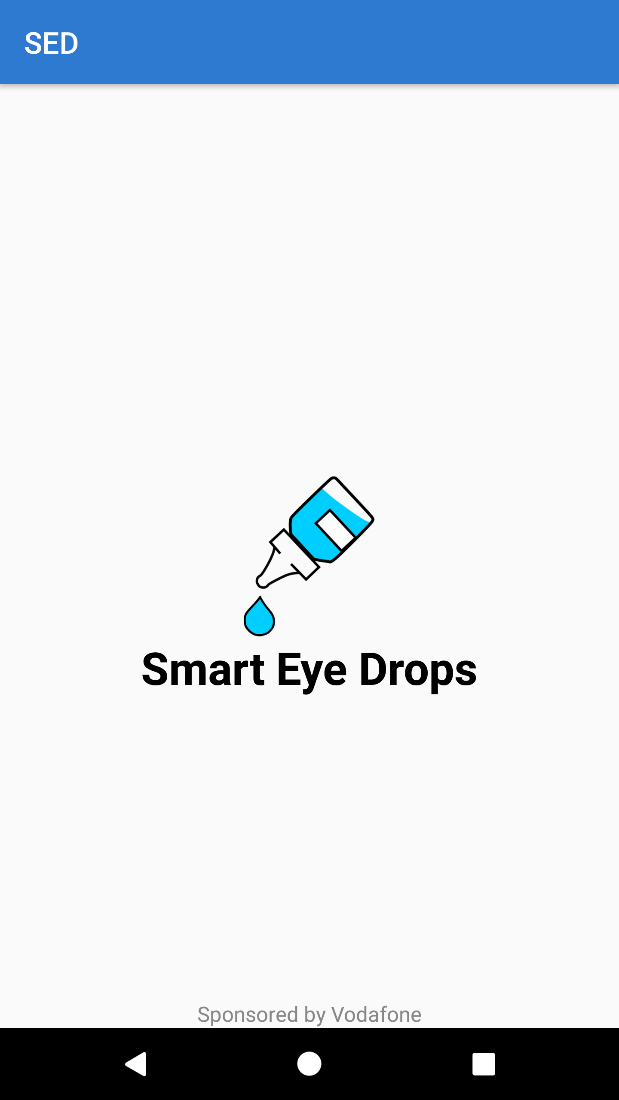
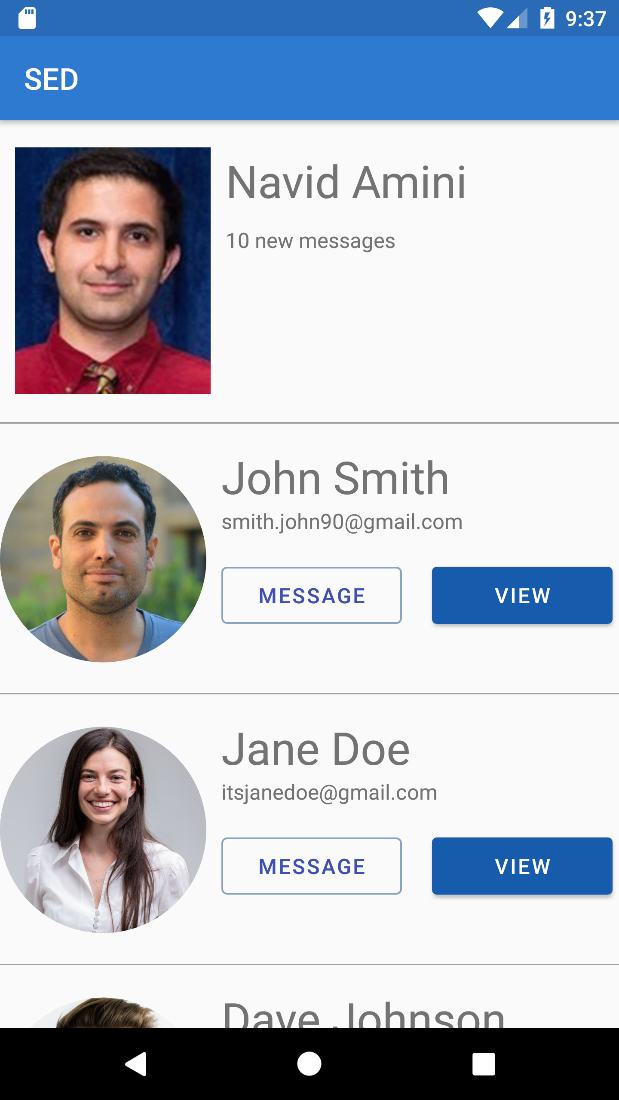
### 9.1.2 Doctor Interface

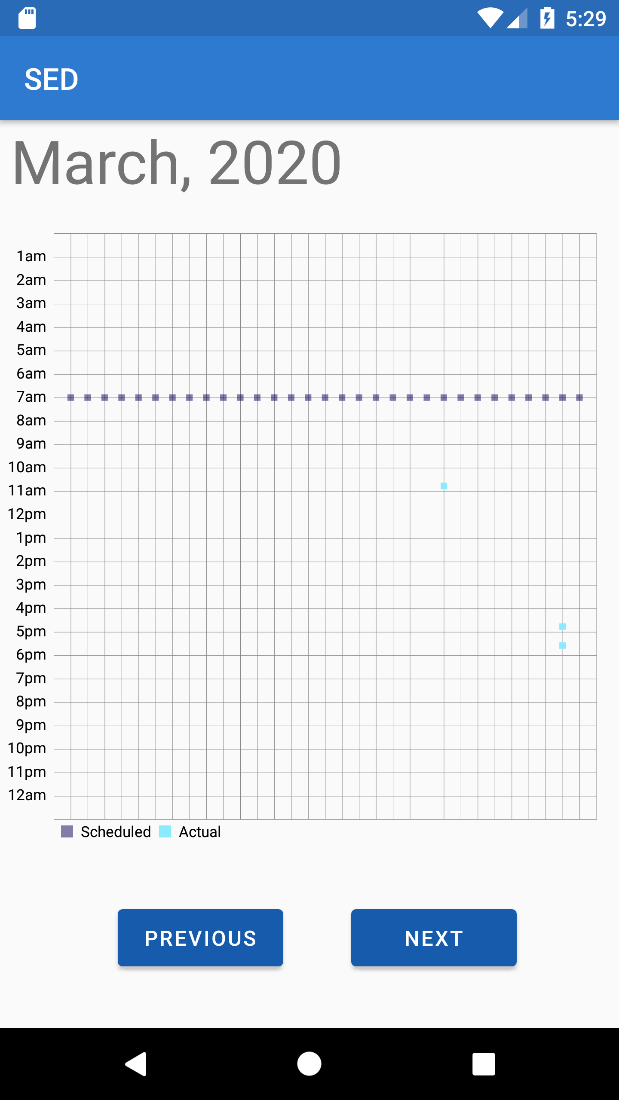
The physician will be able to view their own profile and have a list of their patients listed. The doctor will have the option to message their patients or view their adherence rates via a scatter chart. Once the physician clicks “View”, they will be able to view their patient’s adherence chart for the current month. They will also have the option to look at the adherence chart for previous months.

## 9.2 Screen Frameworks or Images



## 9.3 User Interface Flow Model



# 10. Requirements Validation and Verification

TBD.

# 11. Glossary

1. **AWS**: Amazon Web Services which provides on-demand cloud computing platforms and APIs. <https://aws.amazon.com/>
2. **AWS Elastic Compute Cloud (EC2)**: A service provided by AWS that allows one to create their own server. <https://aws.amazon.com/ec2/>
3. **AWS DynamoDB:** A NoSQL database provided by AWS. <https://aws.amazon.com/dynamodb/>
4. **AWS Lambda:** Allows one to run their code so that the AWS services can communicate. <https://aws.amazon.com/lambda/>
5. **AWS API Gateway:** Allows developers to create, publish, maintain, and secure APIs at any scale. <https://aws.amazon.com/api-gateway/>
6. **FSR**: Force Sensitive Resistor which allows for detection of physical pressure from squeezing. <https://learn.adafruit.com/force-sensitive-resistor-fsr/overview>
7. **MMC**: MBientLab MetaMotion C sensor. <https://mbientlab.com/metamotionc/>

# 12. References

Ada, Lady. “Force Sensitive Resistor (FSR).” *Adafruit Learning System*, <https://learn.adafruit.com/force-sensitive-resistor-fsr/overview>.

*Amazon Web Services (AWS) - Cloud Computing Services*. https://aws.amazon.com/.

“MMC.” *MbientLab*, 28 Feb. 2019, https://mbientlab.com/metamotionc/.