Health App

Software Requirements Specification

Version 1.6

December 01, 2017

Gayatri Patel

Melissa Heredia

Gopika Menon

Thanjila Uddin

# 

# **Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Description** | **Author** | **Comments** |
| 09/21/17 | Version 1.0 | Gayatri Patel Melissa Heredia Gopika Menon Thanjila Uddin | First Draft |
| 10/03/17 | Version 1.1 | Gayatri Patel Melissa Heredia Gopika Menon Thanjila Uddin | Modifications and updates made throughout the document. |
| 10/06/17 | Version 1.2 | Gayatri Patel Melissa Heredia Gopika Menon Thanjila Uddin | Included project updates (online version of application) |
| 10/07/17 | Version 1.3 | Gayatri Patel Melissa Heredia Gopika Menon Thanjila Uddin | Correcting any instances of “web application” to “server application” |
| 10/09/17 | Version 1.4 | Gayatri Patel Melissa Heredia Gopika Menon Thanjila Uddin | Updating functional requirements and Data Flow Diagram |
| 10/28/17 | Version 1.5 | Gayatri Patel Melissa Heredia Gopika Menon Thanjila Uddin | Updated functional requirements |
| 12/01/17 | Version 1.6 | Gayatri Patel Melissa Heredia Gopika Menon Thanjila Uddin | Updated according to final implementation of the application |

# **Document Approval**

The following Software Requirements Specification has been accepted and approved by the following:

|  |  |  |  |
| --- | --- | --- | --- |
| **Signature** | **Printed Name** | **Title** | **Date** |
|  | Gayatri Patel | Team Lead / Frontend Lead |  |
|  | Melissa Heredia | QA Lead / Backend Lead |  |
|  | Gopika Menon | Presentation Lead / Source Control Lead |  |
|  | Thanjila Uddin | Documentation / Architecture Planning Lead |  |
|  | Amrit Misra, MD | Client |  |

**Table of Contents**

[**Revision History**](#_kdk0cbfhcrjf)1

[**Document Approval**](#_v01y6p9fhjy9)1

[**1. Introduction**](#_of9ywv8sw7xu)4

[1.1 Purpose](#_8w1xyvqzfjvu) 4

[1.2 Scope](#_5edhukbu6nav) 4

[1.3 Definitions, Acronyms, and Abbreviations](#_ef112qweolkj) 4

[1.4. References](#_hc0dy0thvape) 6

[1.5 Overview](#_58h8dhyoxqgh) 6

[**2. General Description**](#_qanh3e360g4q)7

[2.1 Product Perspective](#_ldm88nk9skzq) 7

[2.2 Product Functions](#_pbp8z0c7uewj) 7

[2.3 User Characteristics](#_ksa9cqvoxmdt) 8

[2.4 General Constraints](#_m8mbi7tiwqkn) 8

[2.5 Assumptions and Dependencies](#_rismxsgncghs) 8

[2.5.1 Assumptions - Offline Application](#_k6bof1s1dzwh) 8

[2.5.2 Assumptions - Online Application](#_ecoklh38a1nj) 9

[**3. Specific Requirements**](#_rclyhxkax4yi)10

[3.1 External Interface Requirements](#_up3vjvyseka8) 10

[3.1.1 User Interfaces](#_lmzbl2m2yutb) 10

[3.1.2 Hardware Interfaces](#_exc2mcjwh09d) 26

[3.1.3 Software Interfaces](#_dl5rr8mhjzmi) 26

[3.1.4 Communications Interfaces](#_7pfcidqy9uv6) 27

[3.2 Functional Requirements](#_42ixp8tmf3ka) 27

[3.3 Non-Functional Requirements](#_ymck4ca0jmuq) 36

[3.3.1 Performance](#_xt1p4fybp7j2) 36

[3.3.2 Reliability](#_6ioixomqati6) 36

[3.3.4 Security](#_52gn37ap9pyy) 37

[3.3.5 Maintainability](#_oos8c78b1f4f) 37

[3.3.6 Portability](#_u0sl835fpfcl) 38

[3.4 Design Constraints](#_un03gajknztq) 38

[3.5 Logical Database Requirements](#_63zlheep6wpa) 38

[**4. Analysis Models**](#_jc5e0vkk51tl)39

[4.1 Data Flow Diagrams](#_kh8ejpn9vhv8) 39

[**Appendix**](#_ik9mj6uw4vid)42

[Requirements Traceability Matrix](#_phr6a77p7ckp) 42

# 1. Introduction

This chapter will introduce the Health App project by describing the issues that these offline and online applications aim to solve, the target audience, and abbreviations and references associated with the project.

## 1.1 Purpose

The purpose of this Software Requirements Specifications (SRS) document is to provide a detailed description of the Health App project requirements. This document contains the features that will be implemented in the offline and online applications. It will also contain a general description of each application, specific requirements, and visual diagrams associated with the Health App. All contents of this document have been agreed upon by the client, the professors, and the development team.

## 1.2 Scope

The scope of the Health App project is to create two iOS applications called “MyHealthKeeper”. One will be an offline application, and the other will be an online application. Both applications will allow patients between the ages of 16 to 25 with chronic medical conditions to track and store their medical history as they transition from their pediatrician to an adult physician. Patients will be able to enter their medical history, upload documents (medical reports and images), set reminders, print, share, and save their medical records.

To keep the patient’s information secure, these applications will include a Login feature consisting of a username and password. They will also include a Reminder feature that will notify the patient of their upcoming appointments. In addition to the appointment reminders, the patient will receive a monthly reminder that will notify the patient to update their medical history. The applications’ GUI will contain the following features:

* The applications’ display layout
* The applications’ Login feature
* The ability to input Medical Information (Personal Information, Diseases/Illnesses, Medications, Surgeries, Allergies, Vaccines, Additional Information, and Insurance Information)
* Upload Medical Reports (i.e. image of an EKG)
* Print Medical Information
* Share Medical Information
* Enable/Disable the Reminders feature
* Export Medical Information as a PDF

## 1.3 Definitions, Acronyms, and Abbreviations

|  |  |
| --- | --- |
| **Term** | **Definition** |
| User | An individual interacting with the iOS application. |
| HIPAA | Abbreviation for Health Insurance Portability and Accountability Act. Provides data privacy and security for patients’ medical records. |
| Physician | Healthcare professional trained and licensed to practice medicine. |
| Pediatrician | Physician who specializes in the medical care of children. |
| Patient | An individual who is receiving medical treatment or is under the care of a physician. |
| Transition Period | Time period between ages 16 - 25 where patients move from a pediatrician to an adult physician. |
| iOS | Operating system developed by Apple used on devices like the iPhone and iPad. |
| EKG | Abbreviation for Electrocardiogram. This medical test records electrical activity of the heart. |
| Chronic Medical Disease | A disease that lasts three months or more. Often cannot be cured through medication or prevented through vaccination. |
| GUI | Abbreviation for Graphical User Interface. This allows a user to interact with an electronic device through features like windows, icons, and buttons. |
| Server | Processes requests and delivers data to other computers (clients) over a local network or the internet. |
| Bluetooth | A wireless technology to exchange data from mobile phones, computers, and other electronic devices over short distances. |
| FMDB | Abbreviation for Flying Meat Database. It is used as a wrapper for SQLite database. |
| HTTPS | Abbreviation for Hypertext Transfer Protocol Secure. It is secure communication over a network between the iOS application and a server. |
| Objective-C | An Objective Oriented Programming Language used by Apple for the OSX and iOS operating systems. |
| AES | Abbreviation for Advanced Encryption Standard. It is a specification for the encryption of electronic data. |
| SQLCipher | A security extension to the SQLite database platform that facilitates the creation of encrypted databases. |
| AWS | Abbreviation for Amazon Web Services. It is subsidiary of Amazon.com that provides on-demand cloud computing platforms. |
| AWS EC2 Instance | A virtual server in Amazon’s Elastic Computing Cloud (EC2) for running Applications in the AWS infrastructure. |

## 1.4. References

[1] IEEE Software Engineering Standards Committee, “IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications”, October 20, 1998

[2] Apple, “Official Apple Support” support.apple.com/kb/SP751?locale=en\_US.

[3] Deelio Software Requirements Specification, 2016

## 1.5 Overview

This document will define the requirements for MyHealthKeeper. First, a general overview of the applications will be discussed. It will include Product Perspective and Functions, User Characteristics, General Constraints, and Assumptions and Dependencies. The document will then describe the specific requirements for the project including Interface Requirements, Functional Requirements, Non-Functional Requirements, Design Constraints, and Logical Database Requirements. This document will also include a Dataflow Diagram for the reader to get an idea of how information will flow throughout the iOS applications.

# 2. General Description

This section will provide an overview of the two applications the development team is creating. It will include information about the two iOS applications, the various functionalities in each application, information about who the end users will be, constraints the applications will adhere to, and finally, assumptions and dependencies of each application.

## 2.1 Product Perspective

MyHealthKeeper contains two interfaces for the user to interact with: an offline iOS application and an online iOS application. The patient can use both applications to enter, store, print, and share medical information. The applications will allow the patient to keep track of their medical history for their own personal use. Three main features that should be implemented into the iOS application are as follows: allow the patient to enter their medical history through different pages within the application, allow the patient to upload medical reports, and allow the patient to set an appointment reminder.

The offline application will communicate through an embedded database via SQLite. This is to keep the patient’s information private and secure on their iOS device. Additional features will allow the patient to save and export their data as a PDF, connect their iOS device to an AirPrint enabled printer, and print their medical information on an 8x11 paper.

The online application will perform all of the features implemented in the offline application (allow the patient to enter their medical history through different pages within the application, allow the patient to upload medical reports, allow the patient to set an appointment reminder, and allow the patient to save, export, and print their medical information). An additional feature will allow the patient to share their medical information to another iOS device via AirDrop. The main difference between the two applications is that the online version will utilize an AWS server to store and sync data that the patient wishes to access from any iOS device with the application installed.

## 2.2 Product Functions

The main purpose of these applications is to store and retrieve medical information to and from an iOS device. To operate these applications, the patient first needs to create an account with a username and password. After signing in with required and valid credentials, the patient has an option to either Enter their Medical Information, Upload a Document, or Set a Reminder. The ‘Enter Medical History’ feature will include the patient’s personal, medical, and insurance information including name, address, diseases/illnesses, surgeries, family medical history, medical insurance, etc. To enter their medical information, the patient needs to navigate through multiple pages within MyHealthKeeper. The ‘Upload Document’ feature will allow the patient to upload their medical reports such as an image of an EKG. The patient’s medical history and documents can be viewed or deleted after they have been saved. The data will be stored in an embedded database on the patient’s iOS device. The ‘Set Reminder’ feature will allow the patient to set a reminder for any upcoming appointments, and the monthly reminder will appear as a notification for the patient to update their medical history. The ‘Print’ feature will allow the patient to save their medical information as a PDF, and print it on an 8x11 paper.

The online application will have all of the functions of the offline application with a few additional features as follows: The ‘Share’ feature will allow the patient to share their medical information as a PDF, via Airdrop, to another iOS device. Another feature called ‘Sync’ will allows patients to access and edit their data through multiple iOS devices.

## 2.3 User Characteristics

The targeted audience for the offline and online applications are for patients between the ages of 16-25 with chronic medical conditions. Patients between this age range go through a transition period where a change from a pediatrician to an adult physician occurs. The offline and online applications will efficiently assist the patients with this transition. The patient will be able to store their past and current medical information through text, images, and reminders. The patient can save their medication information as a standard or detailed medical report, and save or print this report for the new physician. The Reminder feature of MyHealthKeeper will allow the patient to be prepared for upcoming appointments. Patients will also have the option to print and share their data to their physician. This decreases the waiting period where the physician attempts to gather medical information. It also allows the patient to be more involved with the information transmitted from a pediatrician to the new physician. Patients with chronic medical conditions may not remember all of the treatments or surgeries they underwent as a child. By utilizing MyHealthKeeper, the patient will be more aware of their medical history and the reasons for certain medications or treatments.

## 2.4 General Constraints

Since the application saves the user's medical information and because of a regulatory policy called HIPAA, which states that patient’s medical information should be kept private and secure, the application will have additional security constraints. These constraints will reduce the risk of sensitive information being stolen. The user will have the ability to store their medical information on their iOS device through the offline application’s embedded database. The offline application will also encrypt the information that is stored in that database using SQLCipher. The application has licensing constraints because it uses an Objective-C wrapper called FMDB that has an MIT license. This means that the license in the software’s source code must be clearly stated in MyHealthKeeper’s source code.

The user will also have the option to store the data on a server via the online application. This way of storing data can incur additional risks for the user, but will provide the user with the ability to access their information from any iOS device. To mitigate the potential risks of sending information to the server, the application will have security constraints including encrypting the information sent to the server using HTTPS. Both versions of the application should also have a Login feature so a user cannot access MyHealthKeeper without valid credentials.

## 2.5 Assumptions and Dependencies

### 2.5.1 Assumptions - Offline Application

It is assumed that the offline application will only be used on iOS devices which are iPhone 5s and above with screen resolutions of 1136 x 640 pixels, and up to iPad 5th Generation and above with screen resolutions of 2048 x 1536 pixels [2]. To download the application from the App Store, a network connection is required, and the user should have a valid Apple ID and password. The user’s iOS device is assumed to have a touch screen functionality to interact with MyHealthKeeper’s user interface. MyHealthKeeper should also connect to the iOS device’s camera, and allow the user to take a photo they wish to upload.

### 2.5.2 Assumptions - Online Application

It is assumed that when the user is using the online application, the information they create or edit will be stored on the server. Also, the most recent information can be accessed from any iOS device as long as the application is installed, the user has valid credentials, and the user has tapped on the ‘Sync’ button located in the side-menu. The ‘Share’ feature on the online application also requires a Bluetooth connection from one iOS device to another iOS device to send the patient’s medical information. As previously stated in the Offline Application Assumptions, it is assumed the online application will be used on iOS devices such as iPhone 5s and above with screen resolutions of 1136 x 640 pixels, and iPad 5th Generation and above with screen resolutions 2048 x 1536 pixels [2]. To download the application from the App Store, a network connection is required, and the user needs to have a valid Apple ID and password. If the user logs in or registers on a device or if the user decides to sync their data, a network connection is required as well. The user’s device is assumed to have a touch screen functionality to interact with the user interface.

# 

# 

# 3. Specific Requirements

This chapter will highlight all of the Functional and Non-Functional Requirements necessary for development of the offline and online applications. It will describe any external interfaces required as well as constraints.

## 3.1 External Interface Requirements

This section provides a detailed description of the applications’ display, and any hardware, software, and communications interfaces.

### 3.1.1 User Interfaces

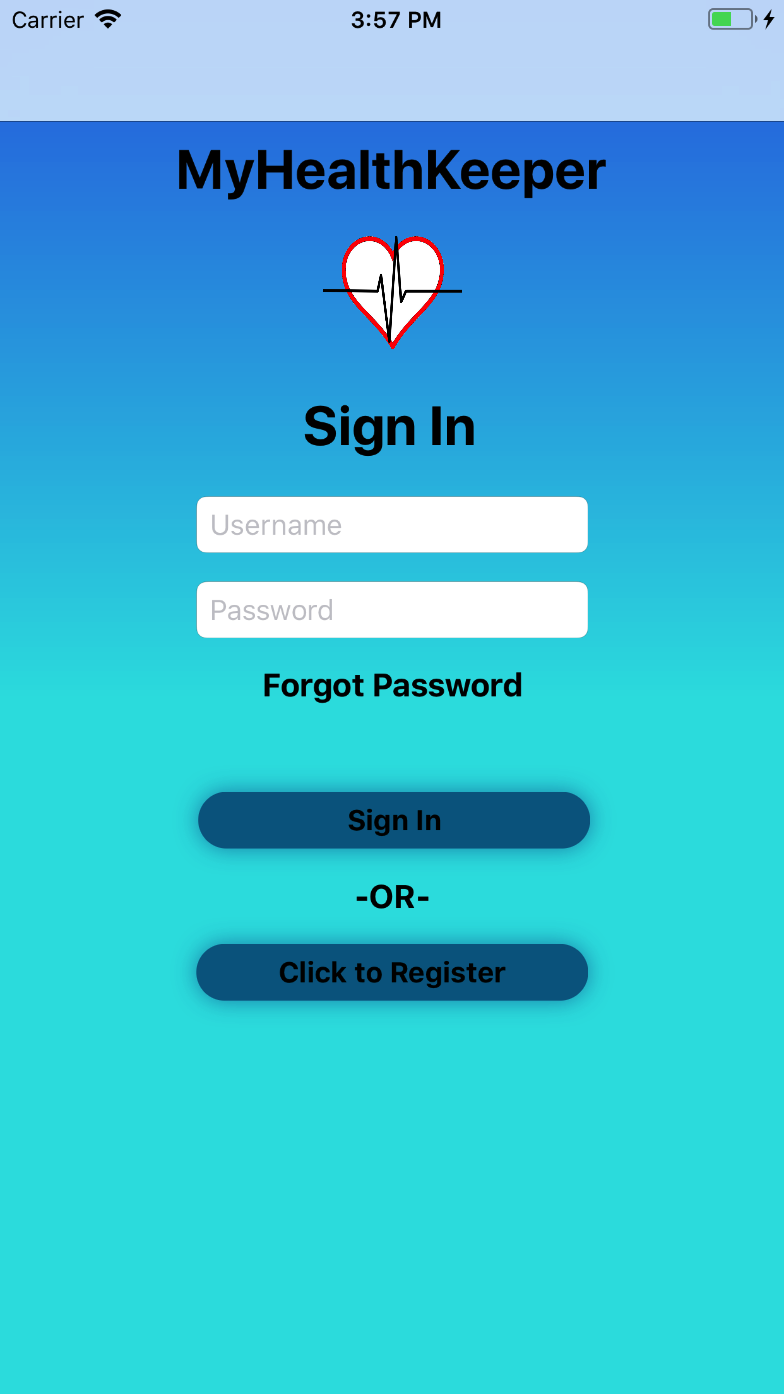
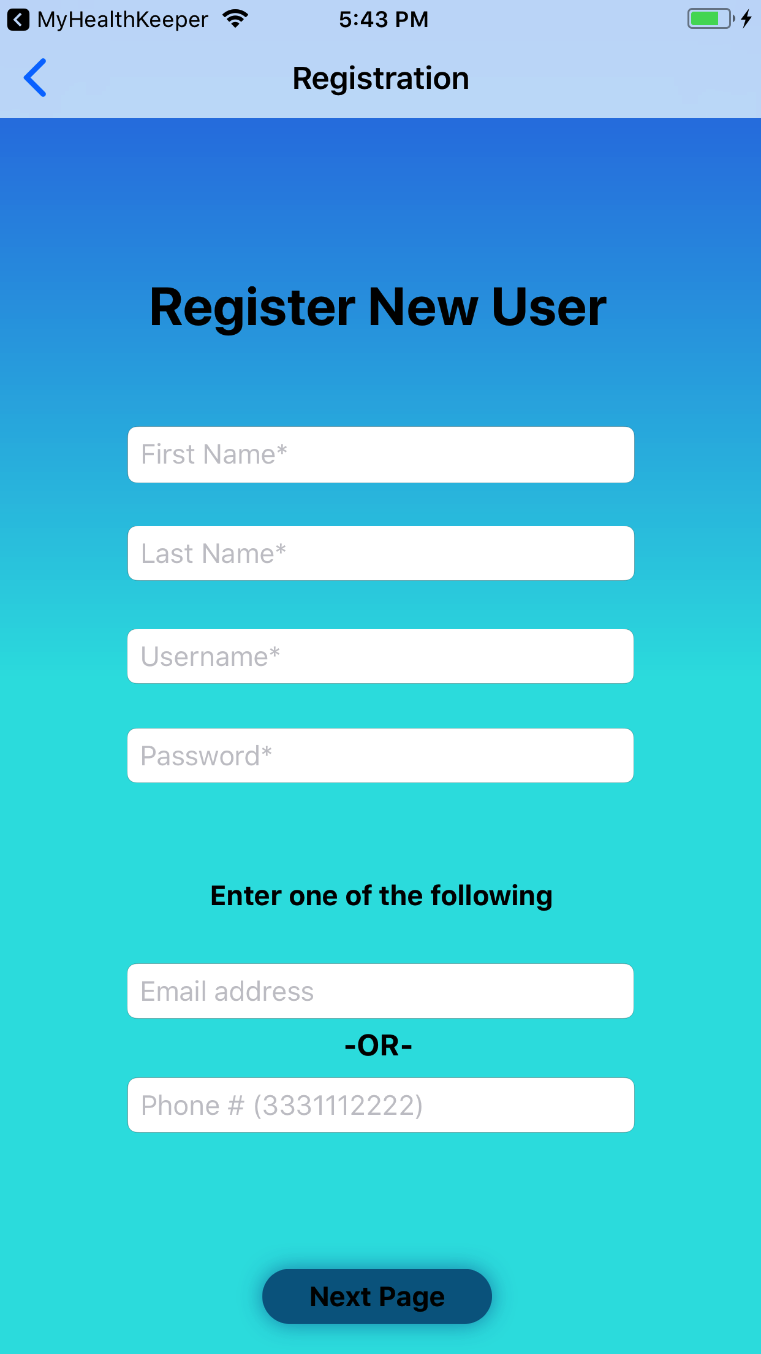
MyHealthKeeper consists of two main interfaces that patients will use: two iOS applications that will be in the form of an offline application and an online application.

When launching MyHealthKeeper, the user will see the launch screen, seen in *Figure 1*.



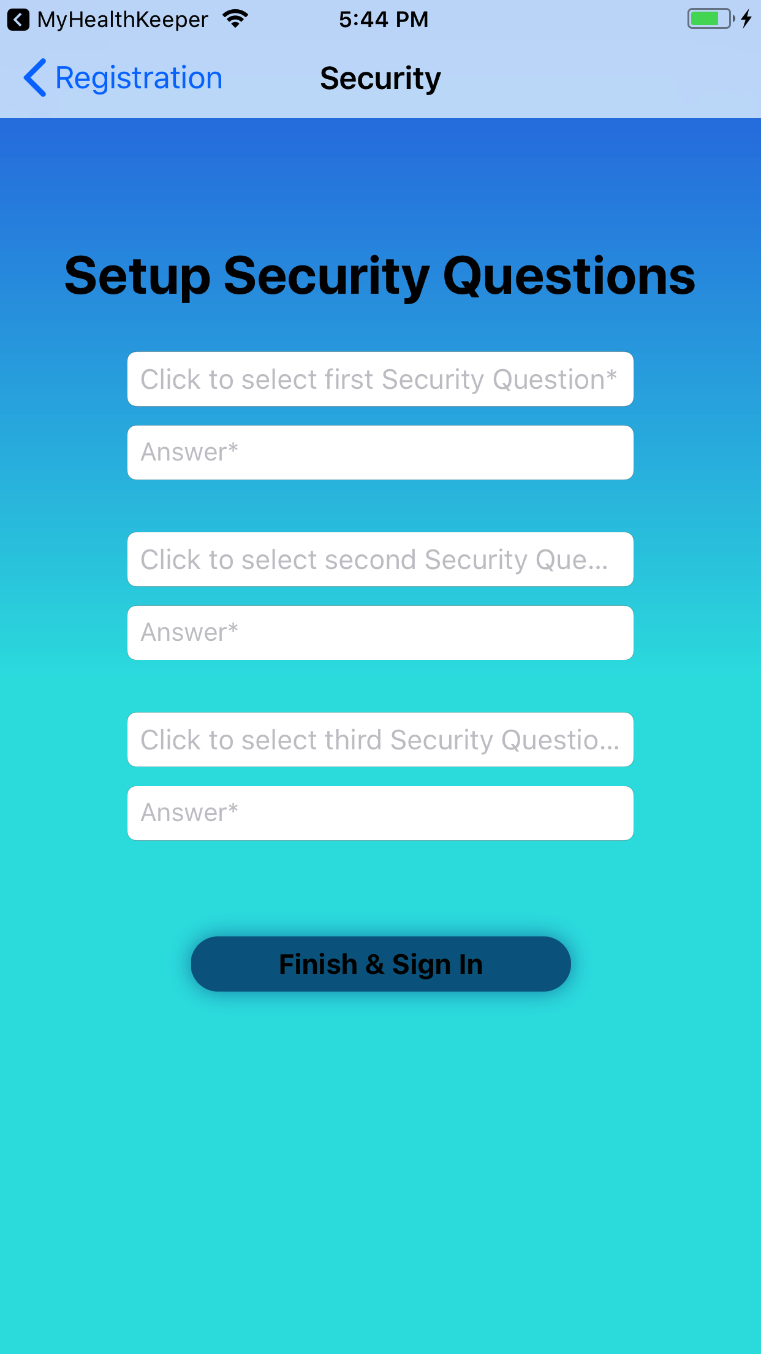
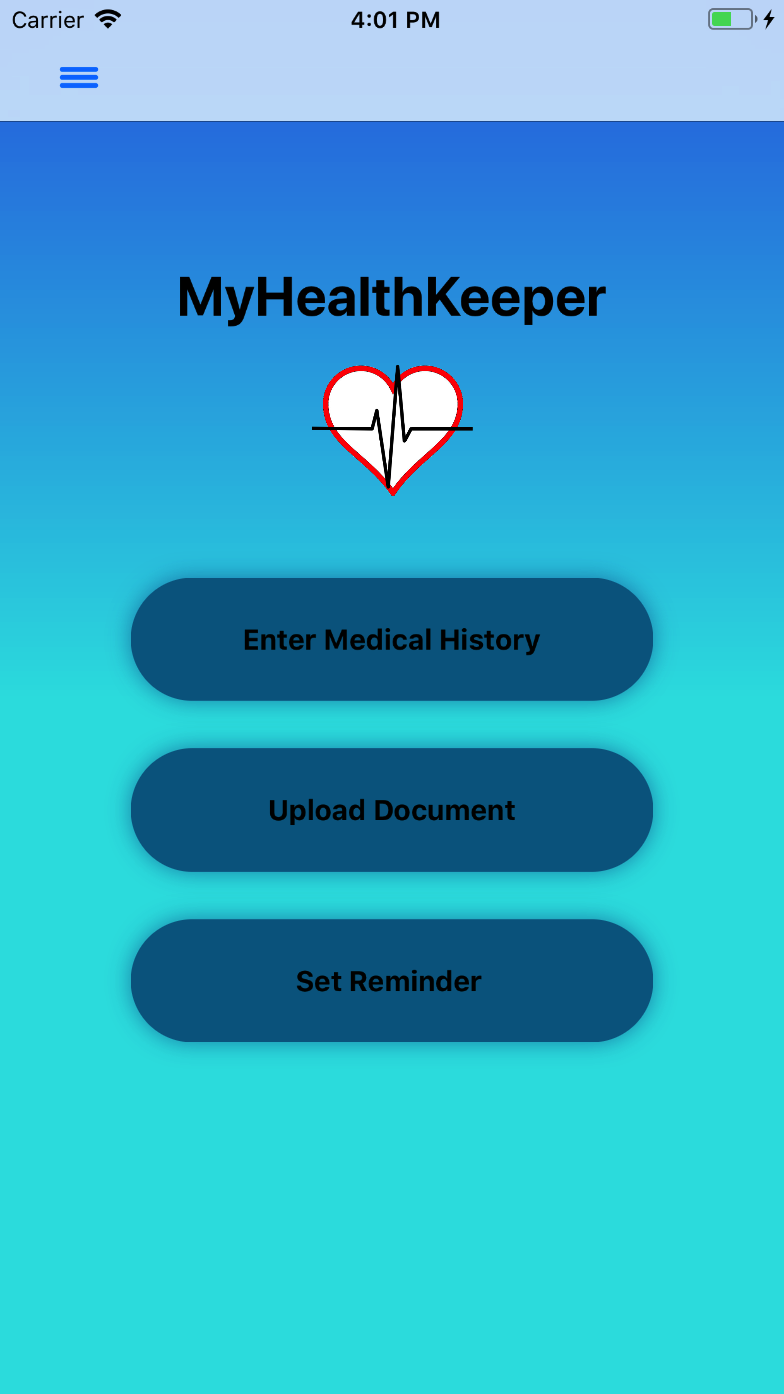
*Figure 1: Launch Screen*

The user will be directed to the Sign In page with an option to Register if they have not created an account, see *Figure 2*. To navigate to the Registration page, the user will tap on the ‘Click to Register’ button. The registration page consists of required text fields for the user to input their information with an option to set an email address or cellphone number, see *Figure 3*.

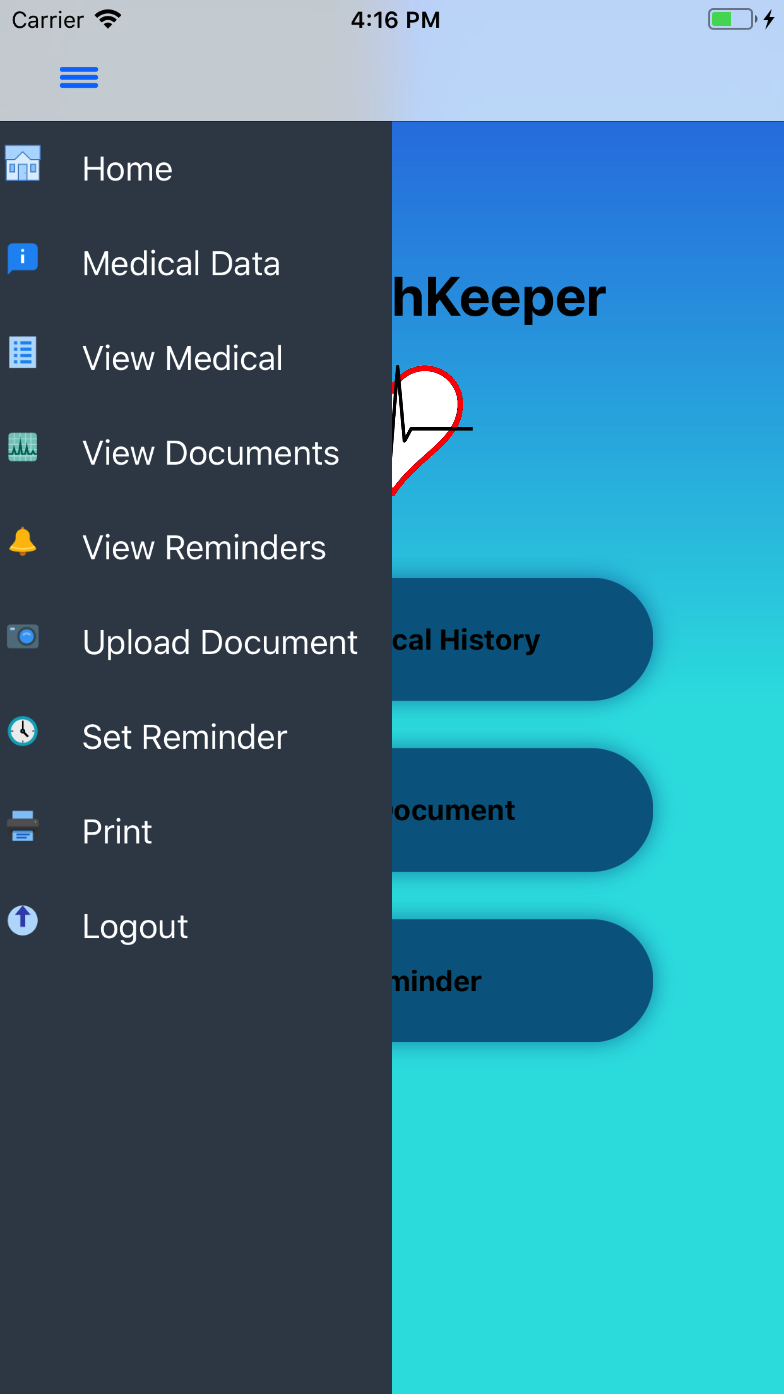
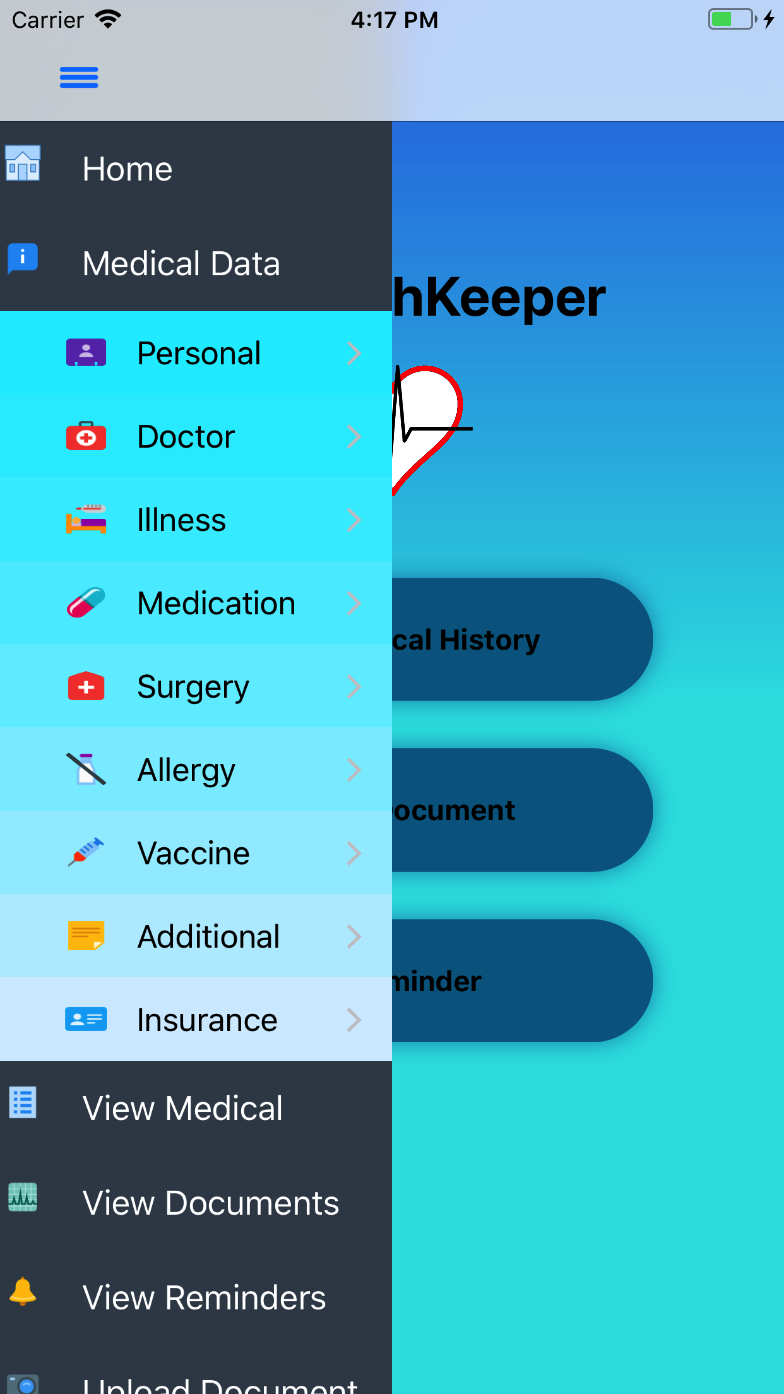
*Figure 2: Sign In* *Figure 3: Registration*

After the user enters the required information, the user taps on the ‘Next Page’ button. The user will be directed to the Setup Security Questions page. The user will be required to set answers to security questions as shown in *Figure 4*. Once the user taps on ‘Finish and Sign In’, the user will be directed to the Home page of MyHealthKeeper shown in *Figure 5*.

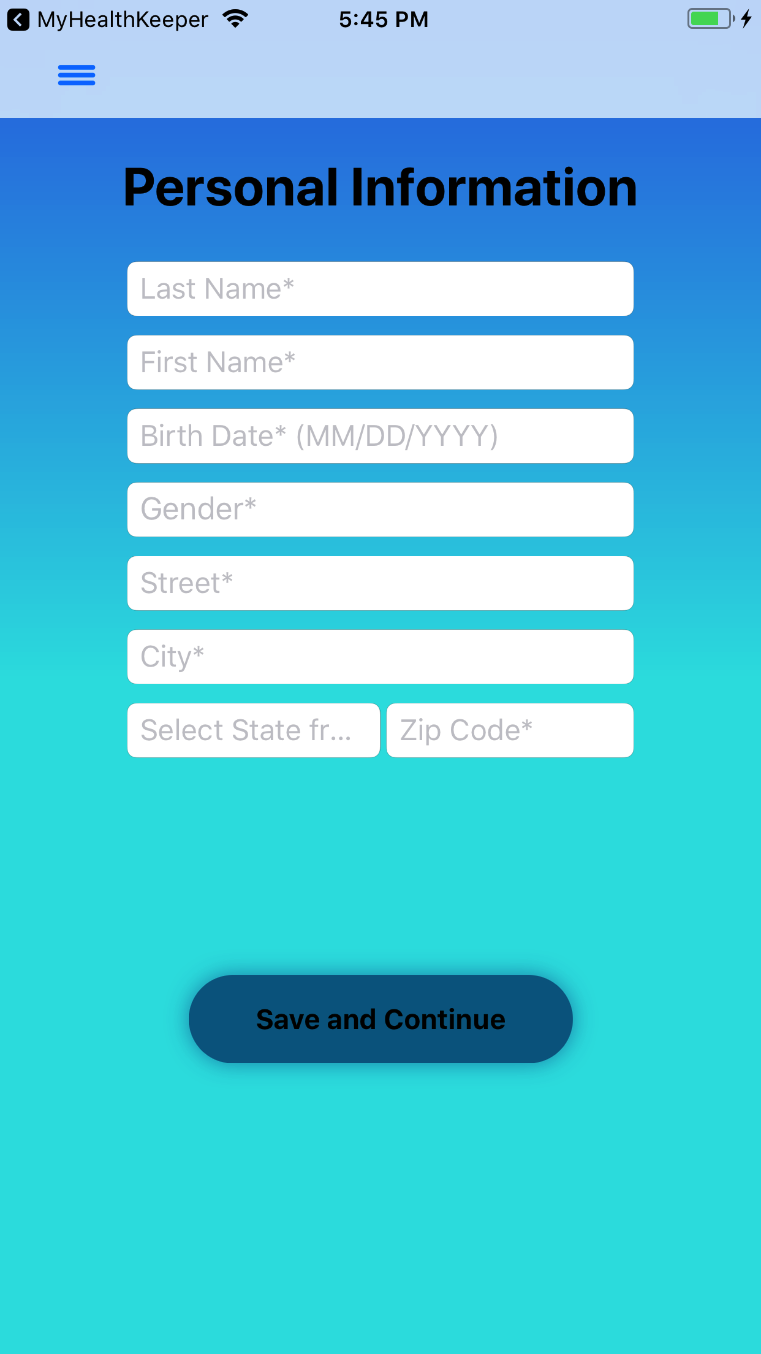
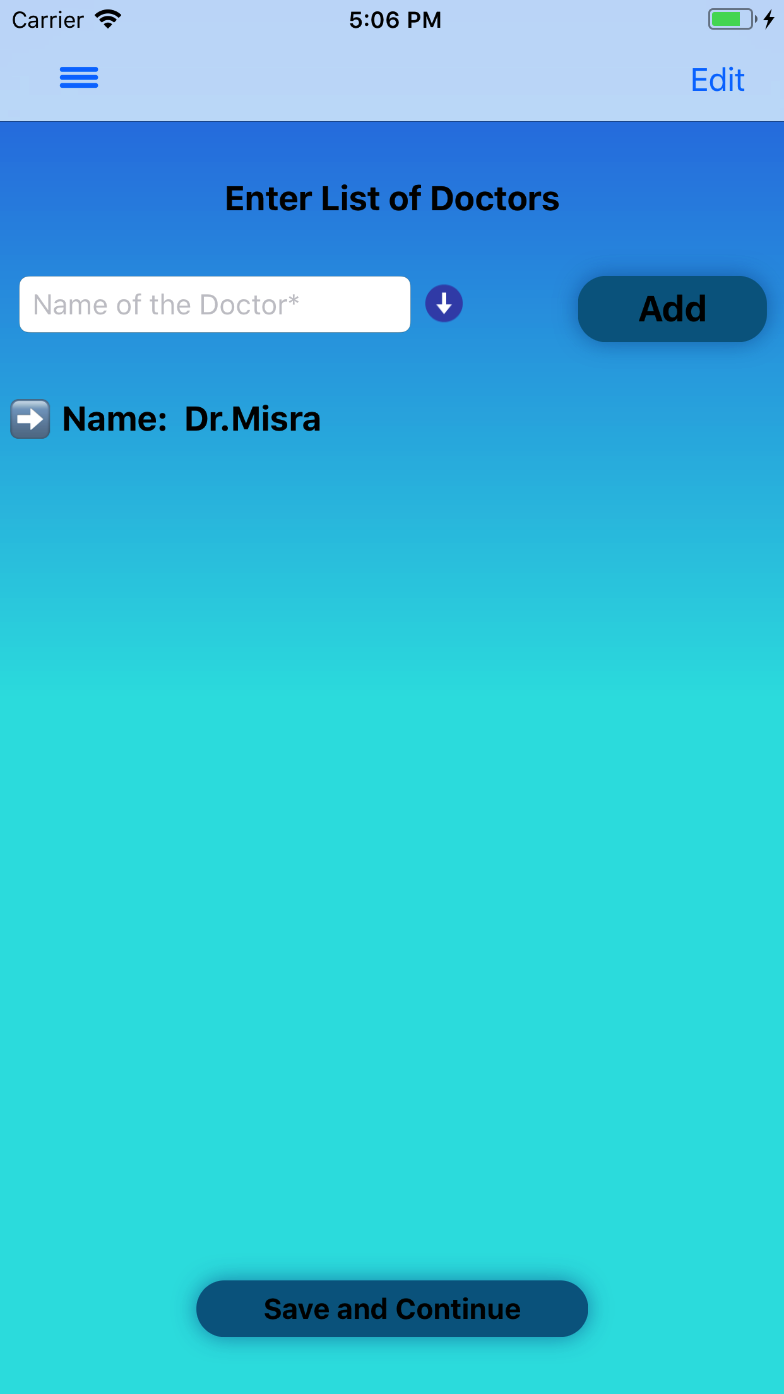
*Figure 4: Setup Security Questions* *Figure 5: Home Page*

If the user taps on the Menu that is located on the top left corner of the Home page, it will allow the user to navigate to any page in the Menu they wish to go to. The menu is shown in *Figure 6.* The user can expand all medical pages by tapping on “Medical Data” in the menu that is shown on *Figure 7.*

* *

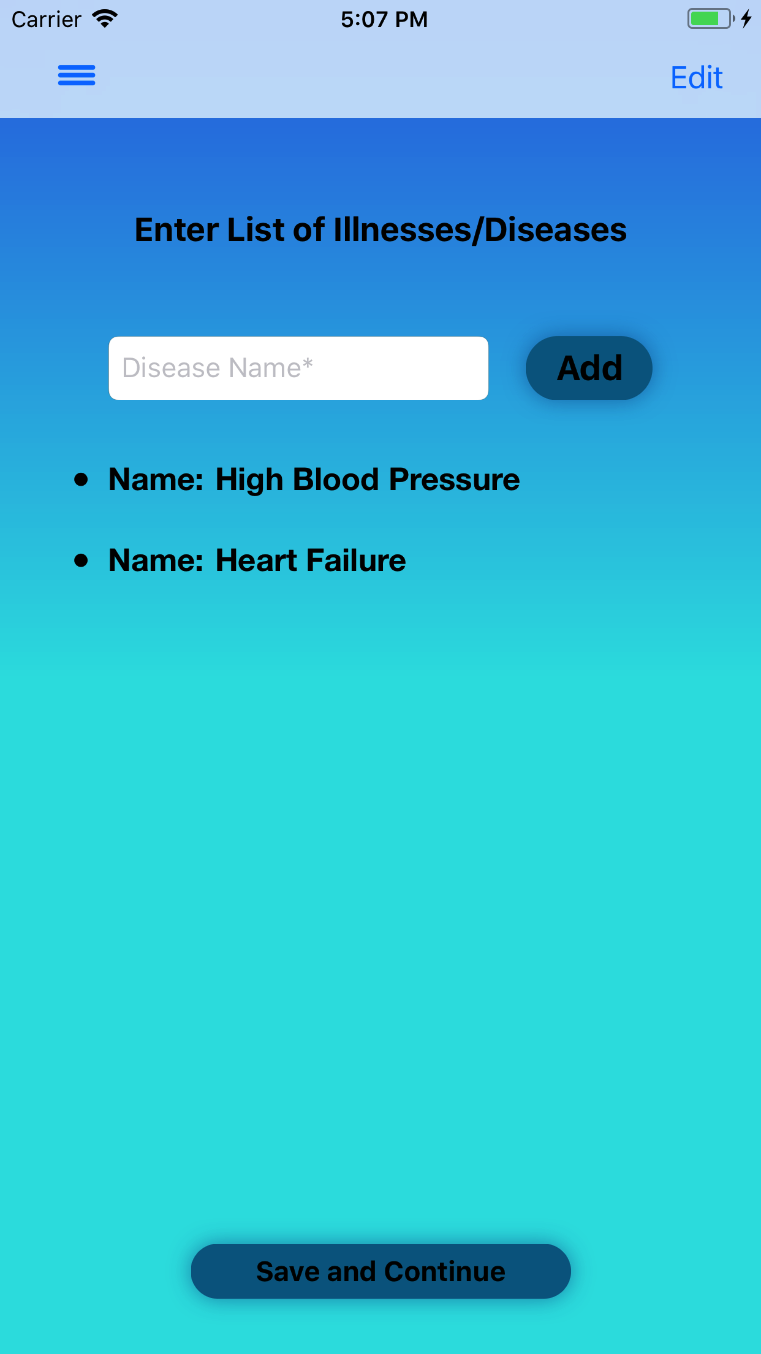
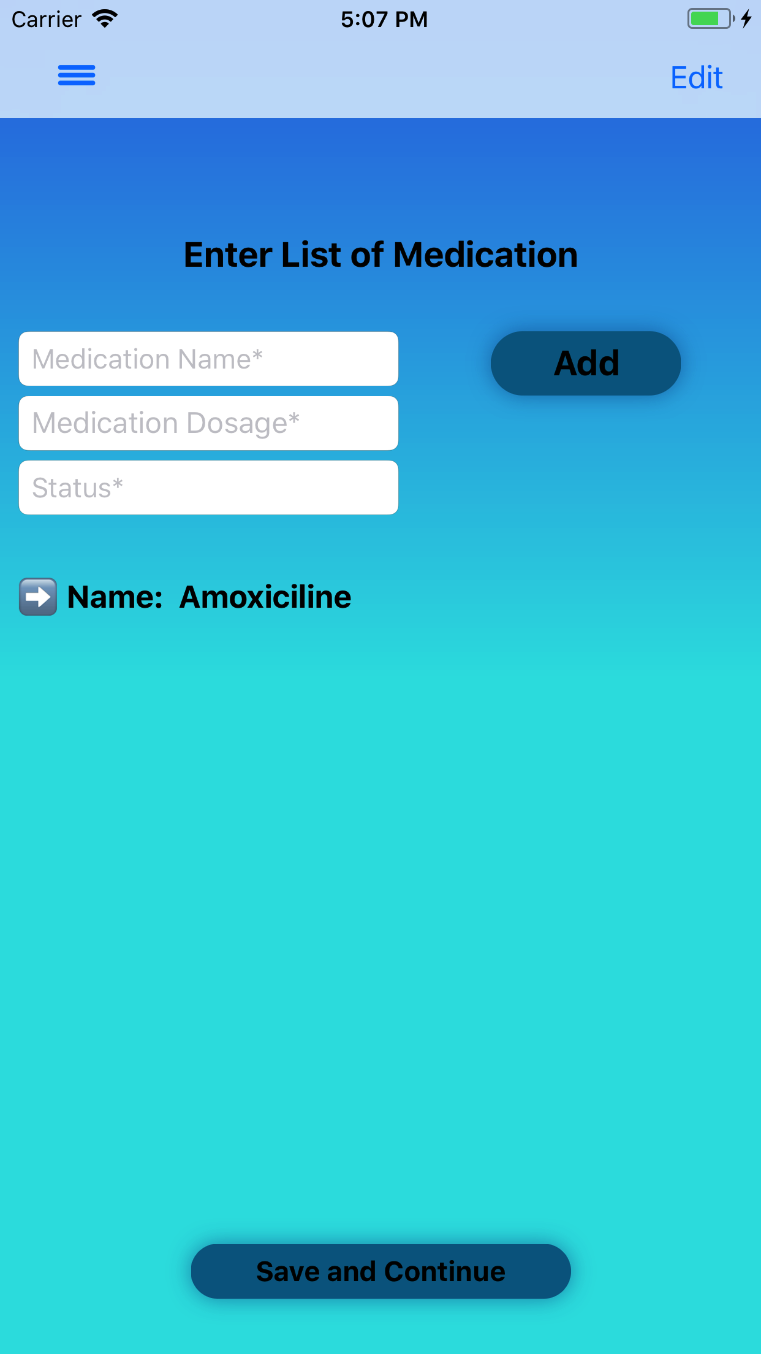
*Figure 6: Menu Figure 7: Menu with Medical pages*

If the user taps ‘Enter Medical History’ button from the Home page, they will be directed to the Personal Information page as shown in *Figure 8.* After entering the required fields, the user taps the ‘Save and Continue’ button, and the user will be directed to the Enter List of Doctor page as Shown in *Figure 9.*

* *

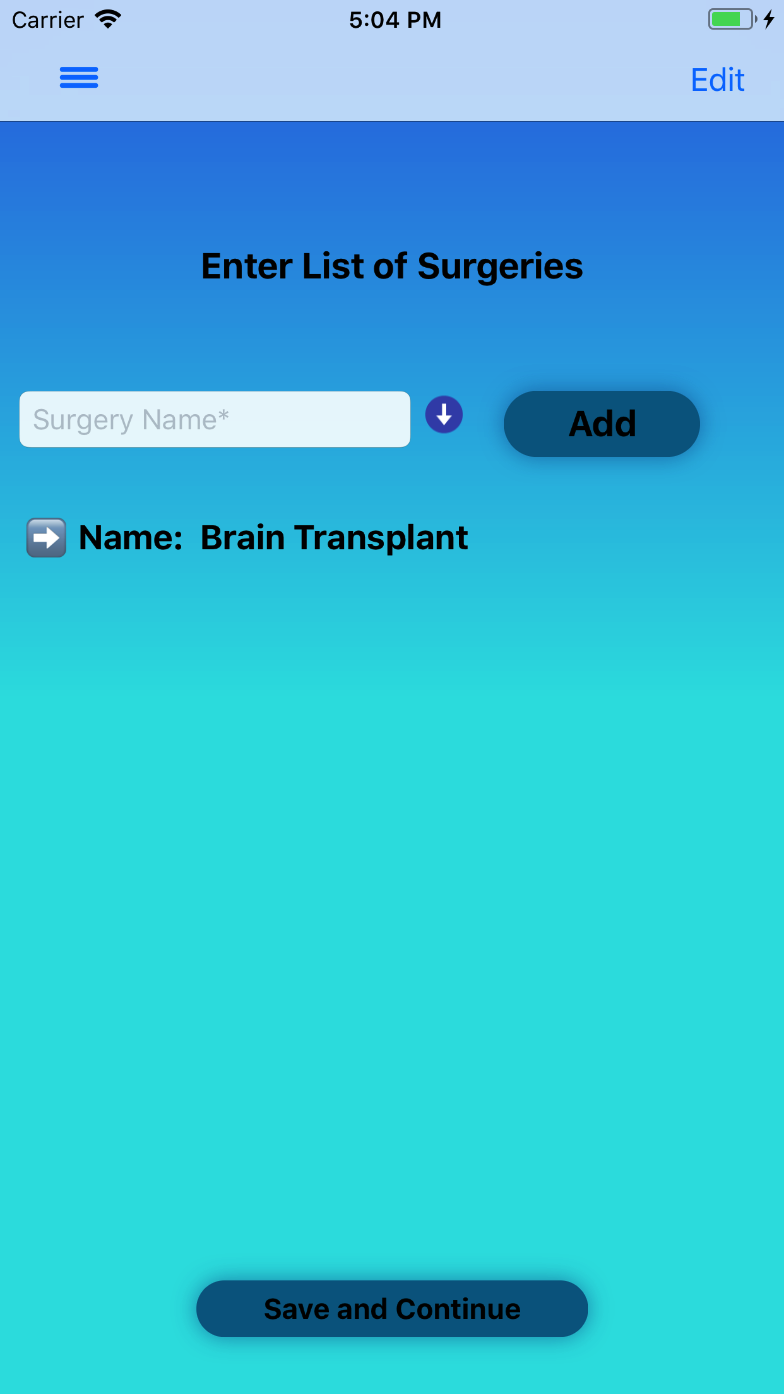
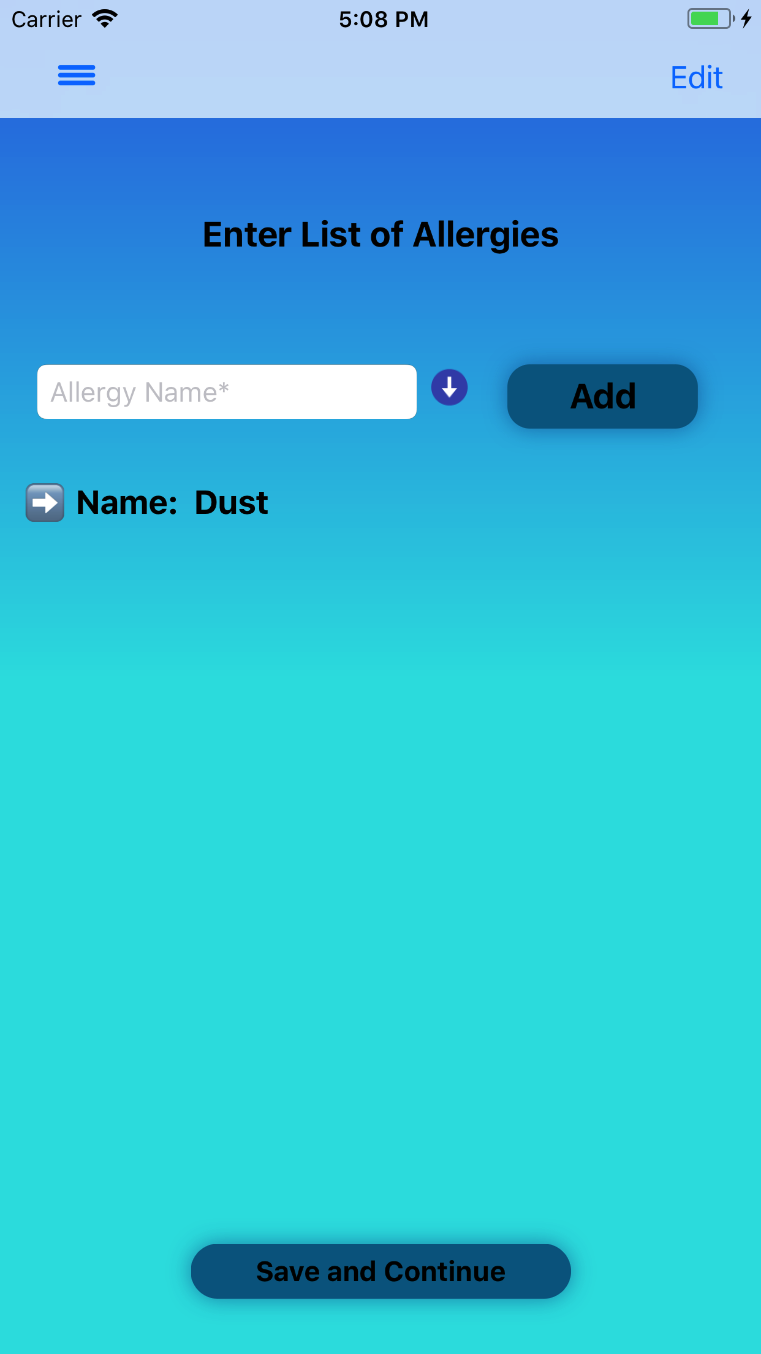
*Figure 8: Personal Data Figure 9: Enter List of Doctors*

After the user enters their Doctor Information, the user taps on the ‘Save and Continue’ button and will be directed to the Enter List of Illnesses/Diseases page as shown in *Figure 10.*  The user taps on the ‘Save and Continue’ button, and will be directed to the Enter List of Medication page as shown in in *Figure 11.*

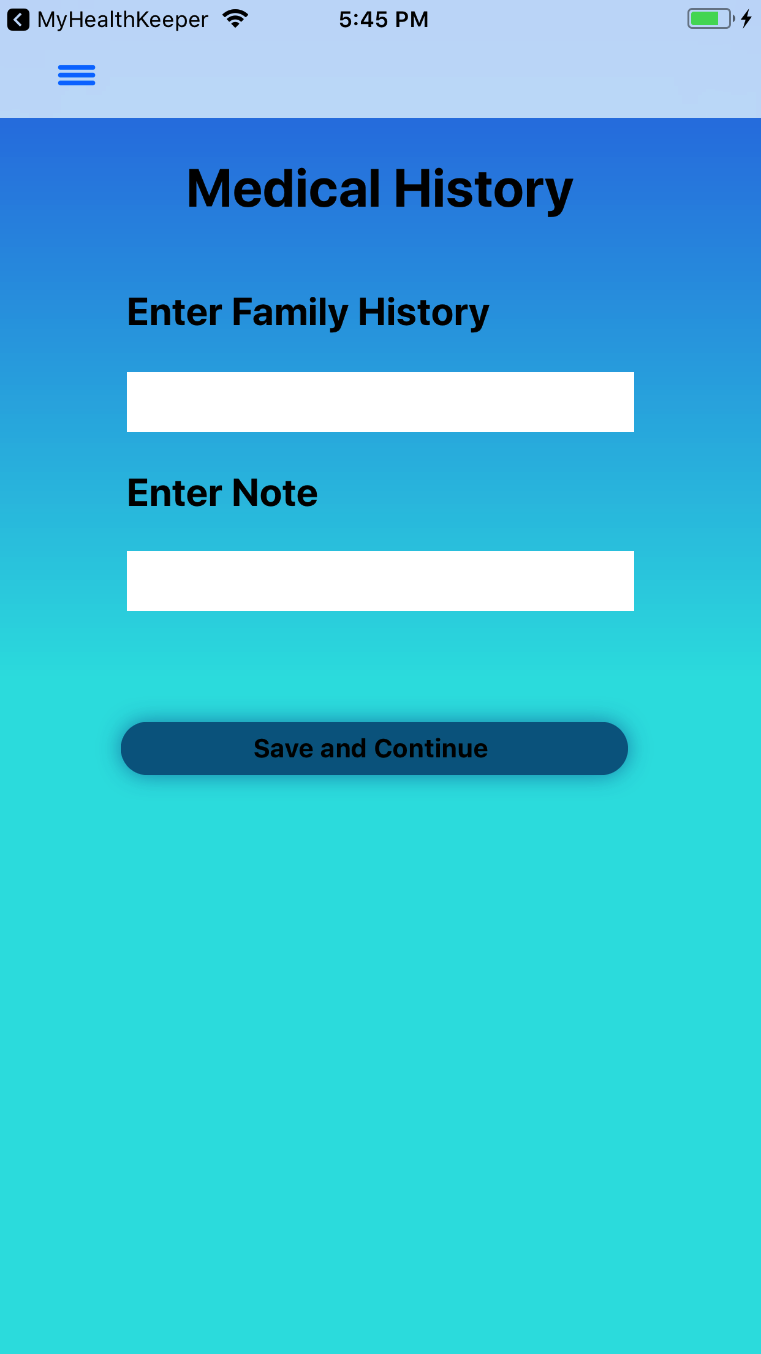
*Figure 10: Enter List of Illnesses/Diseases Figure 11: Enter List of Medication*

After the user enters their Medication Information, the user taps on the ‘Save and Continue’ button and will be directed to the Enter List of Surgeries page as shown in *Figure 12.* After the user enters their Surgery Information, the user taps on the ‘Save and Continue’ button and will be directed to the Enter List of Allergies page as shown in *Figure 13.*

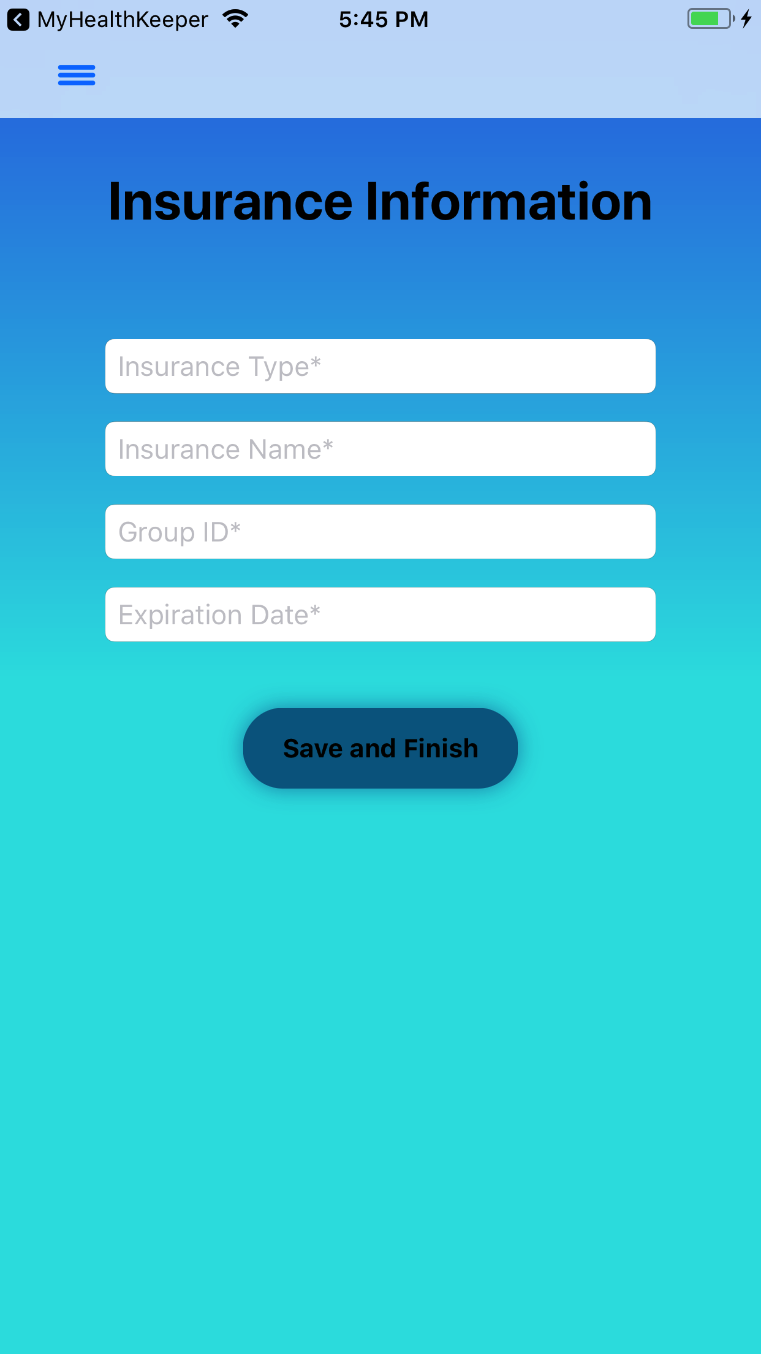
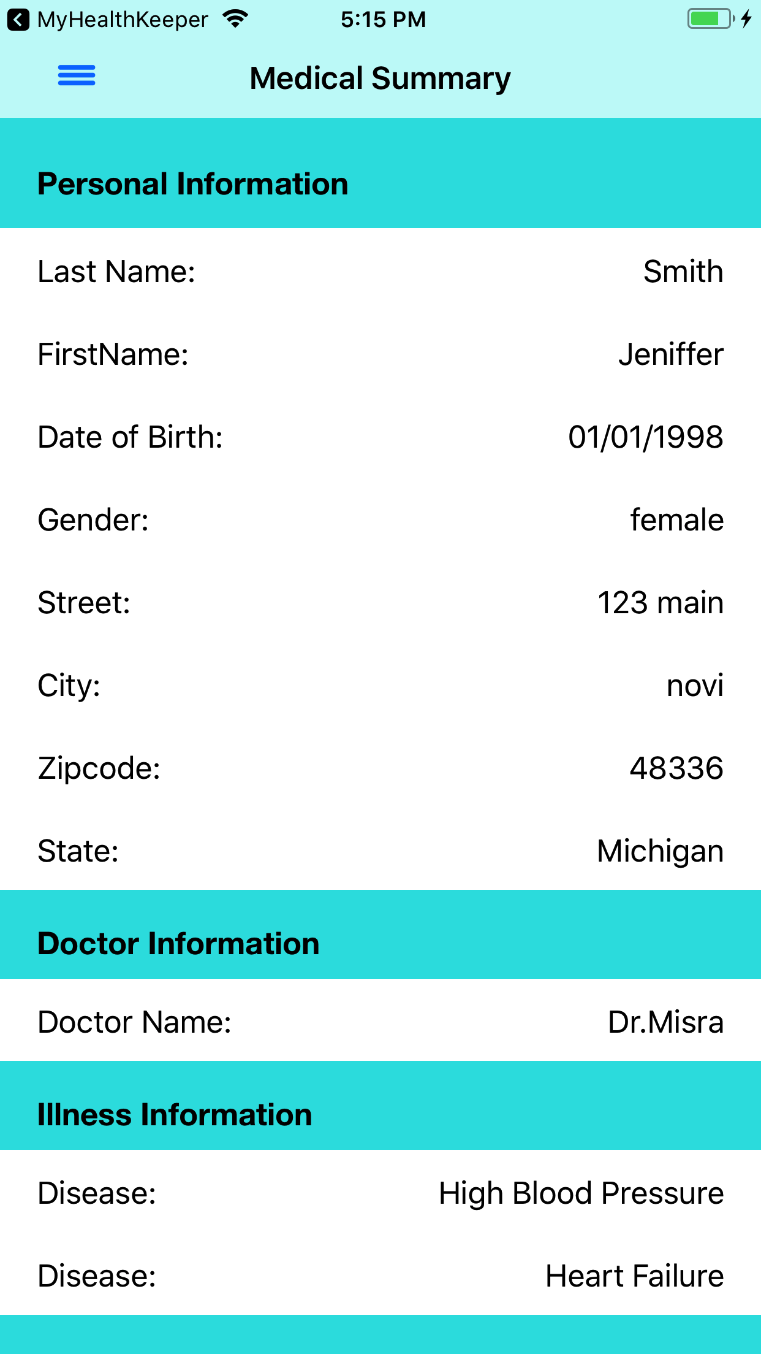
*Figure 12: Enter List of Surgeries Figure 13: Enter List of Allergies*

After the user enters their Allergy Information, they can tap on the ‘Save and Continue’ button and will be directed to the Enter List of Vaccines page as shown in *Figure 14.* After the user enters their vaccine information, the user can tap on the ‘Save and Continue’ button and will be directed to the Medical History page as shown in *Figure 15.*

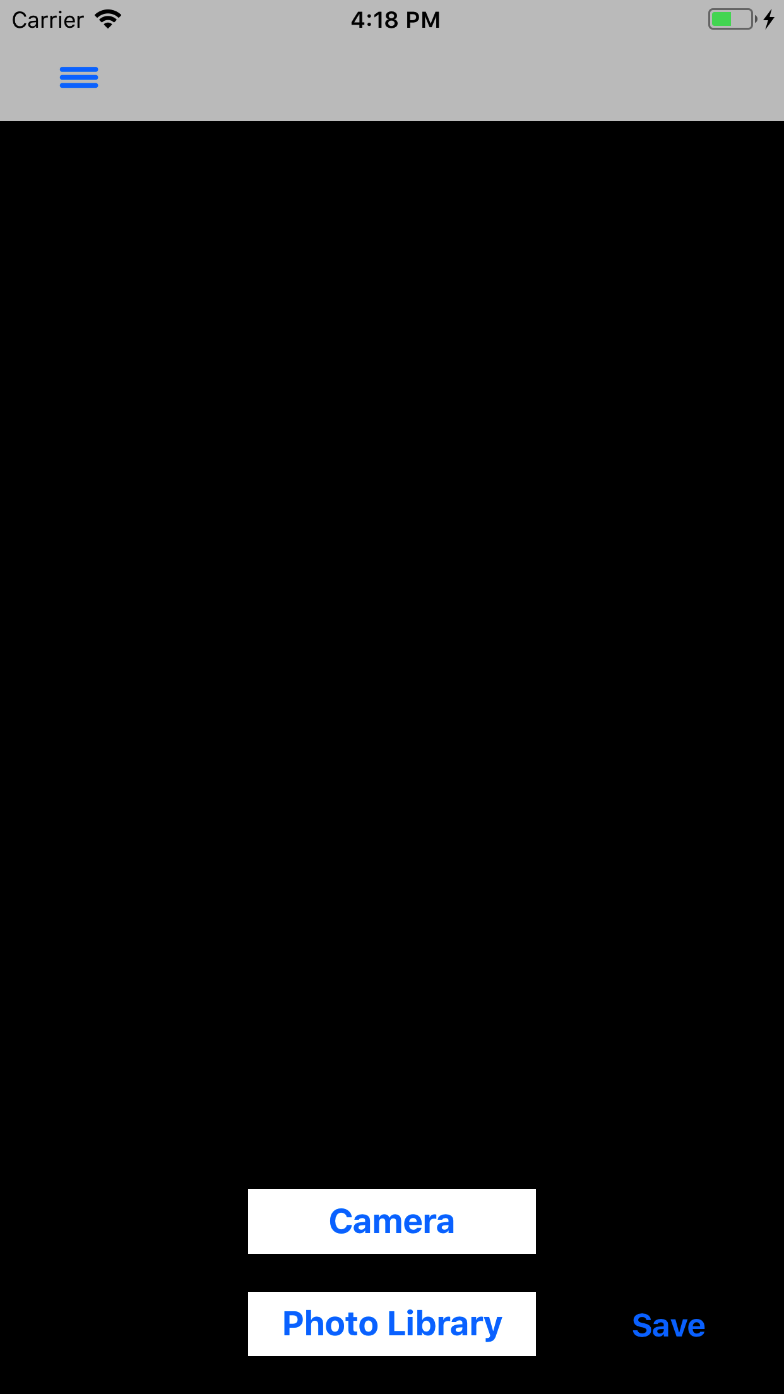
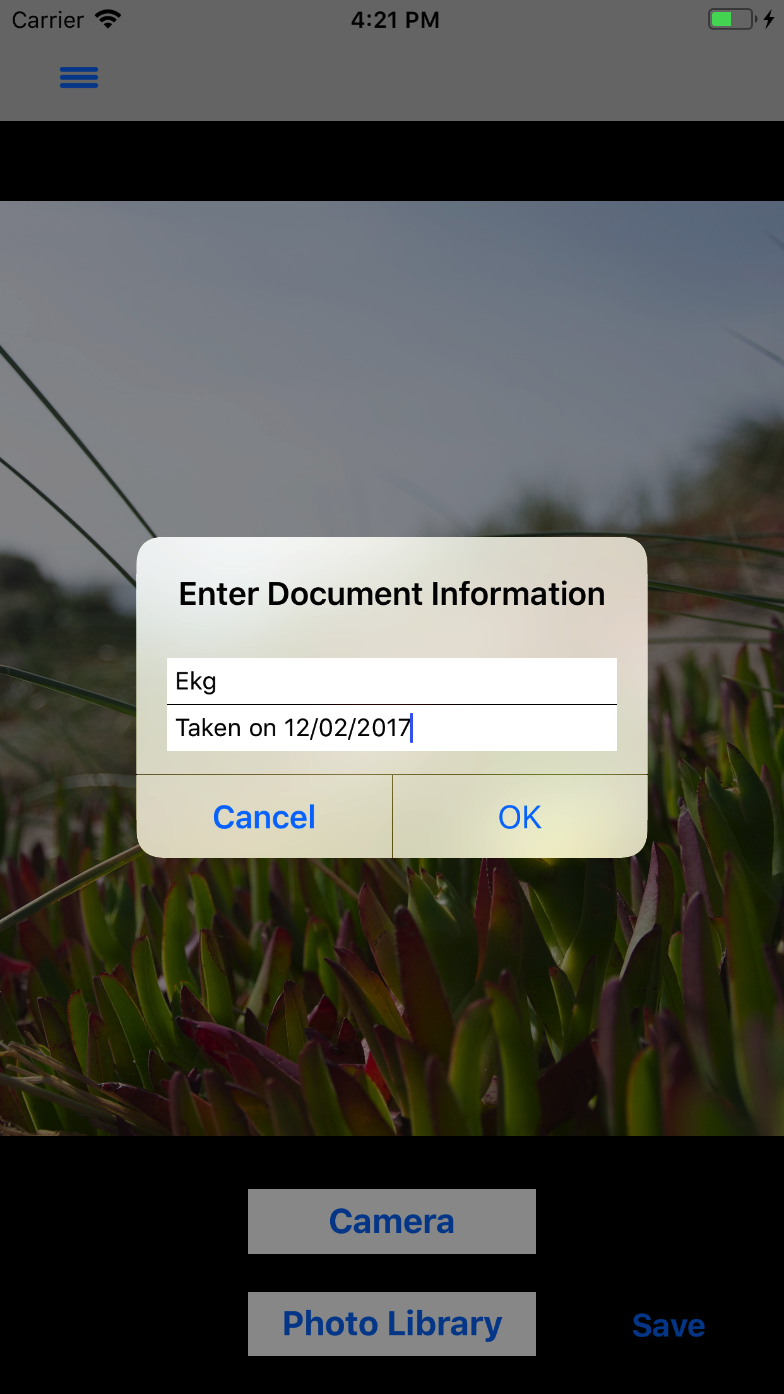
*Figure 14: Enter List of Vaccines Figure 15: Medical History*

After the user enters their Medical History, they can tap on the ‘Save and Continue’ button and will be directed to the Insurance Information page as shown in *Figure 16.*  After the user enters their Insurance information, the user can tap on the ‘Save and Finish’ button. The user can view their Medical Information on the Medical Summary page as shown in *Figure 17.*

* *

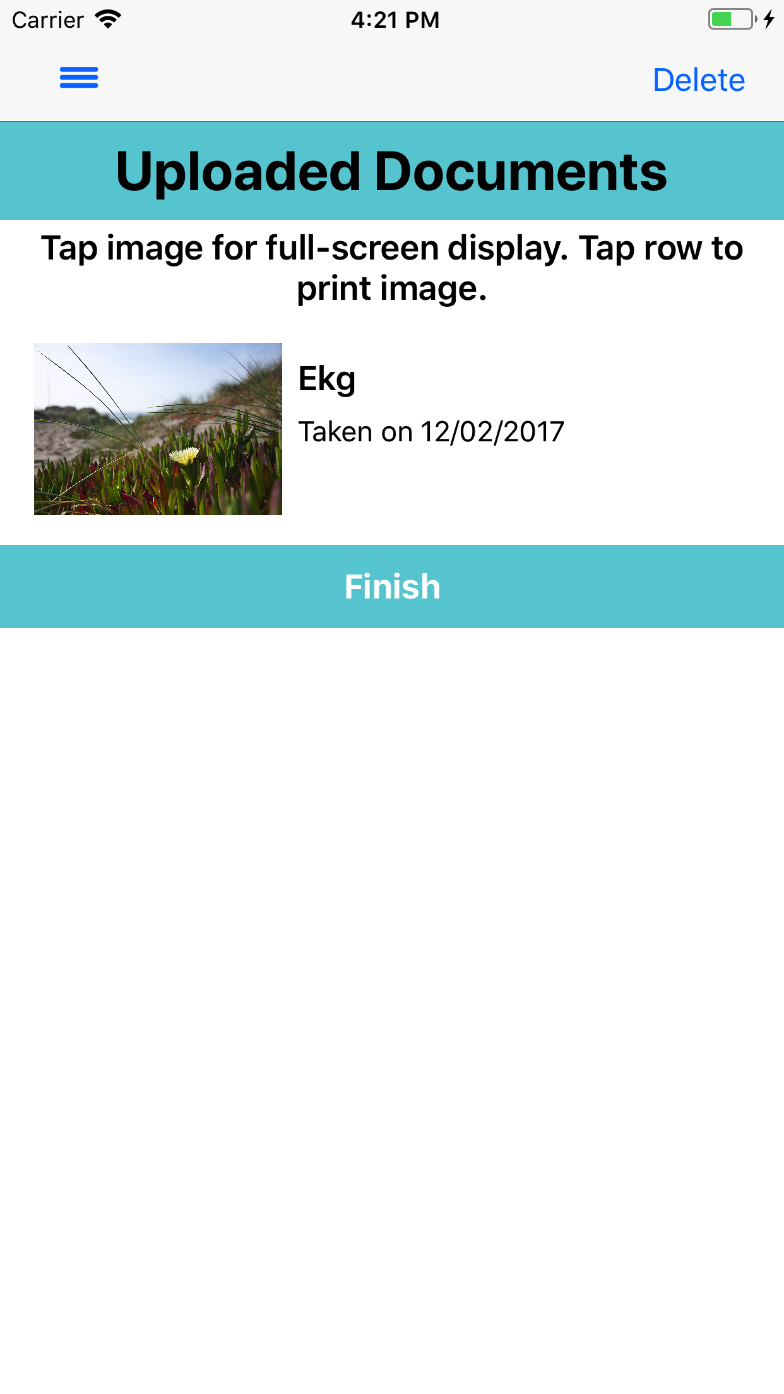
*Figure 16: Insurance Information Figure 17: Medical Summary*

If the user taps on the ‘Upload Document’ button on the Menu, or from the Main page, the user will be directed to the Upload Document page. The user can take a photo using the camera on their iOS device or choose an existing photo from the device’s photo library as shown in *Figure 18*. When the user taps on the ‘Save’ button, an alert appears where the user must enter the Document Name in the alert’s text field. The user can also enter a Document Description in the optional text field below Document Name as shown in *Figure 19.*

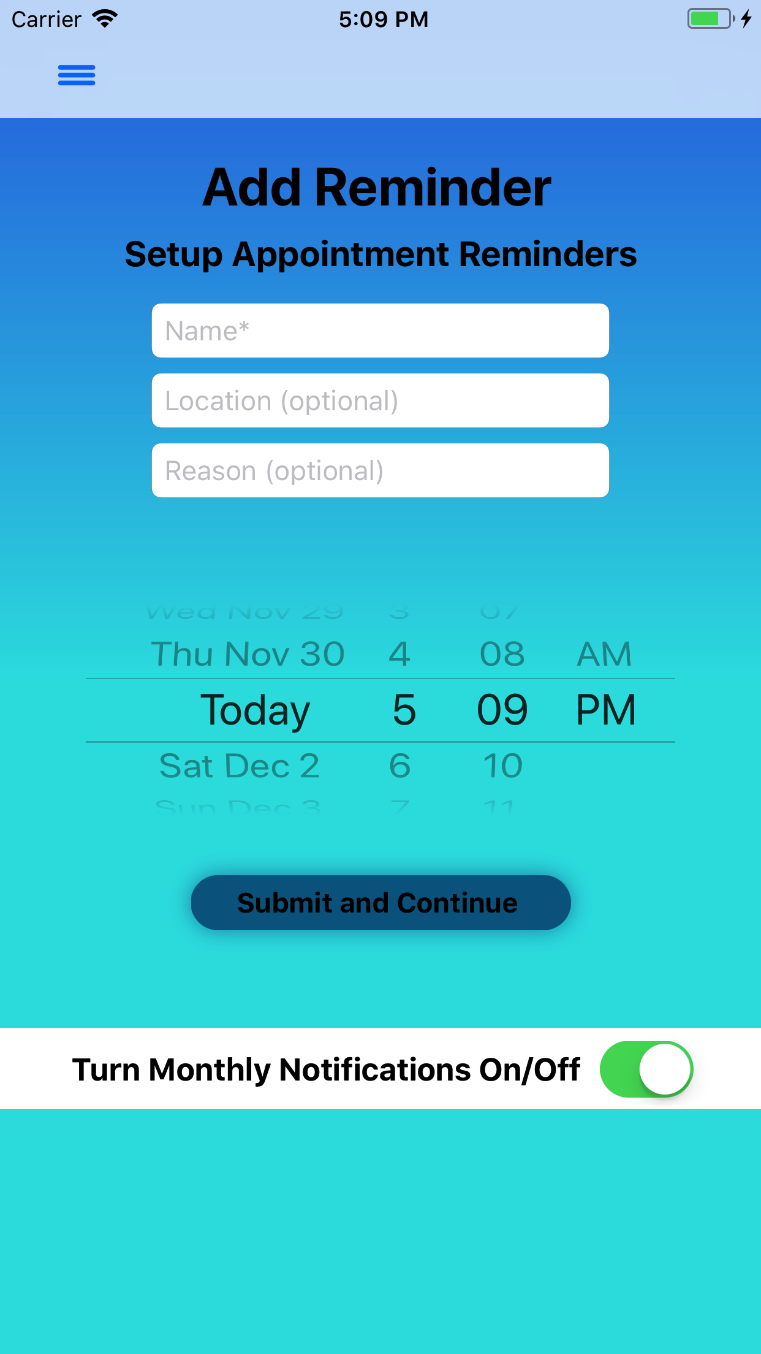
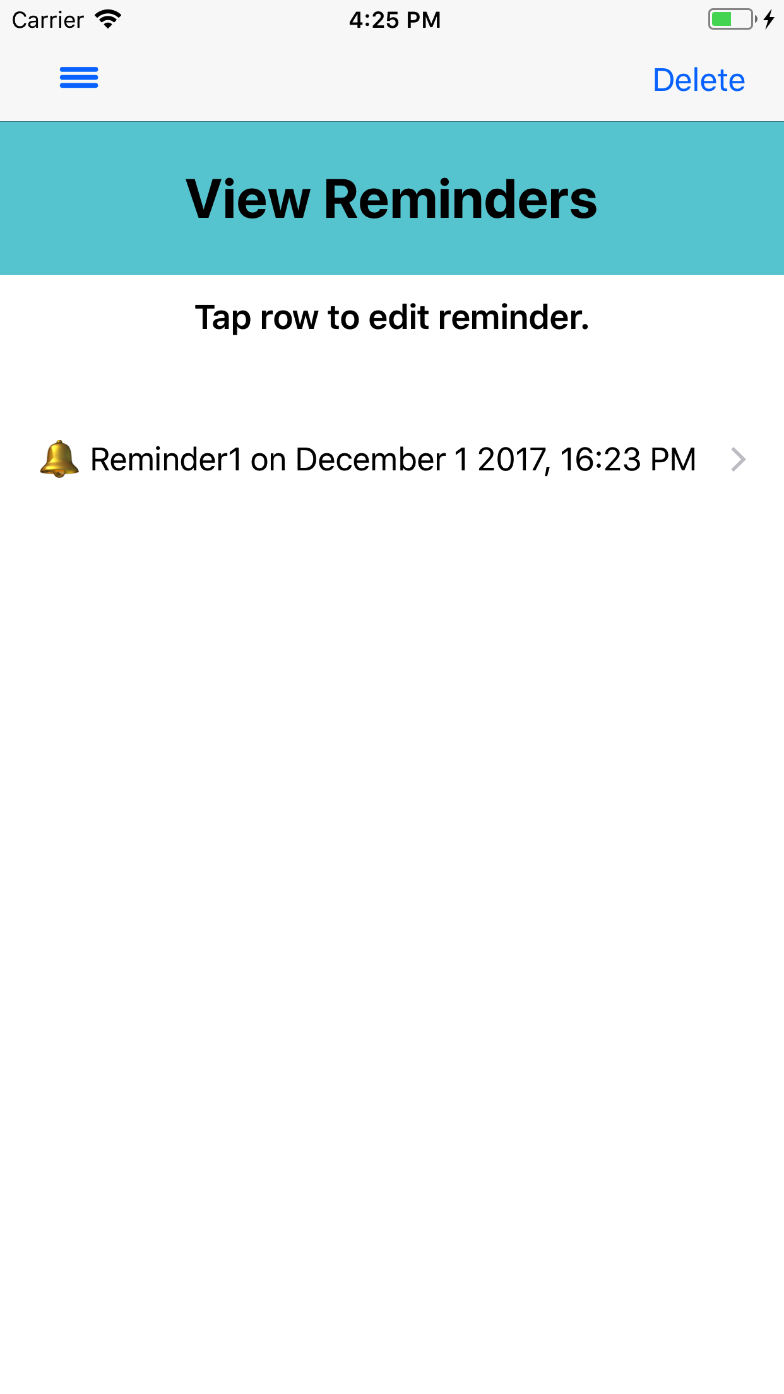
*Figure 18: Upload a Document Figure 19: Enter Document Information*

After the user enters their Document Information, they can tap the ‘OK’ button on the alert and will be directed to the Document Summary page as shown in *Figure 20*.



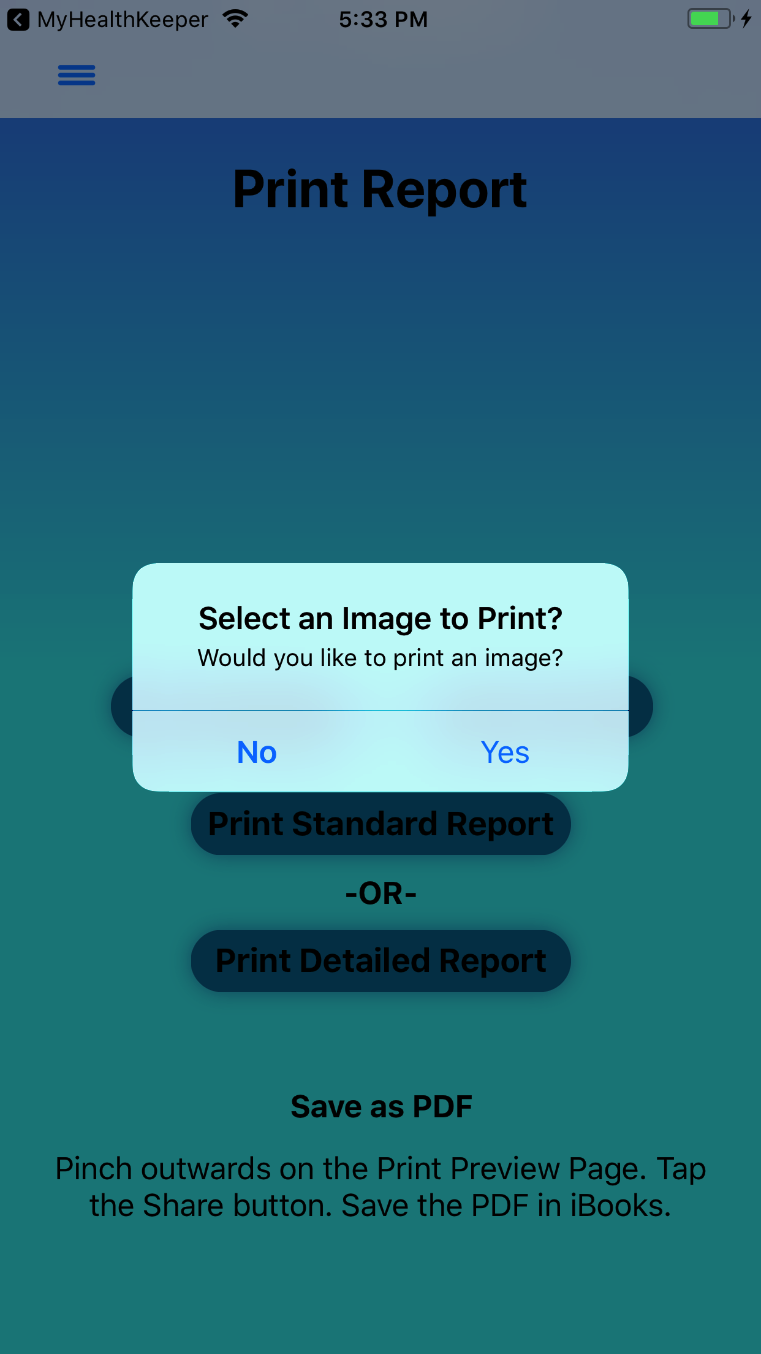
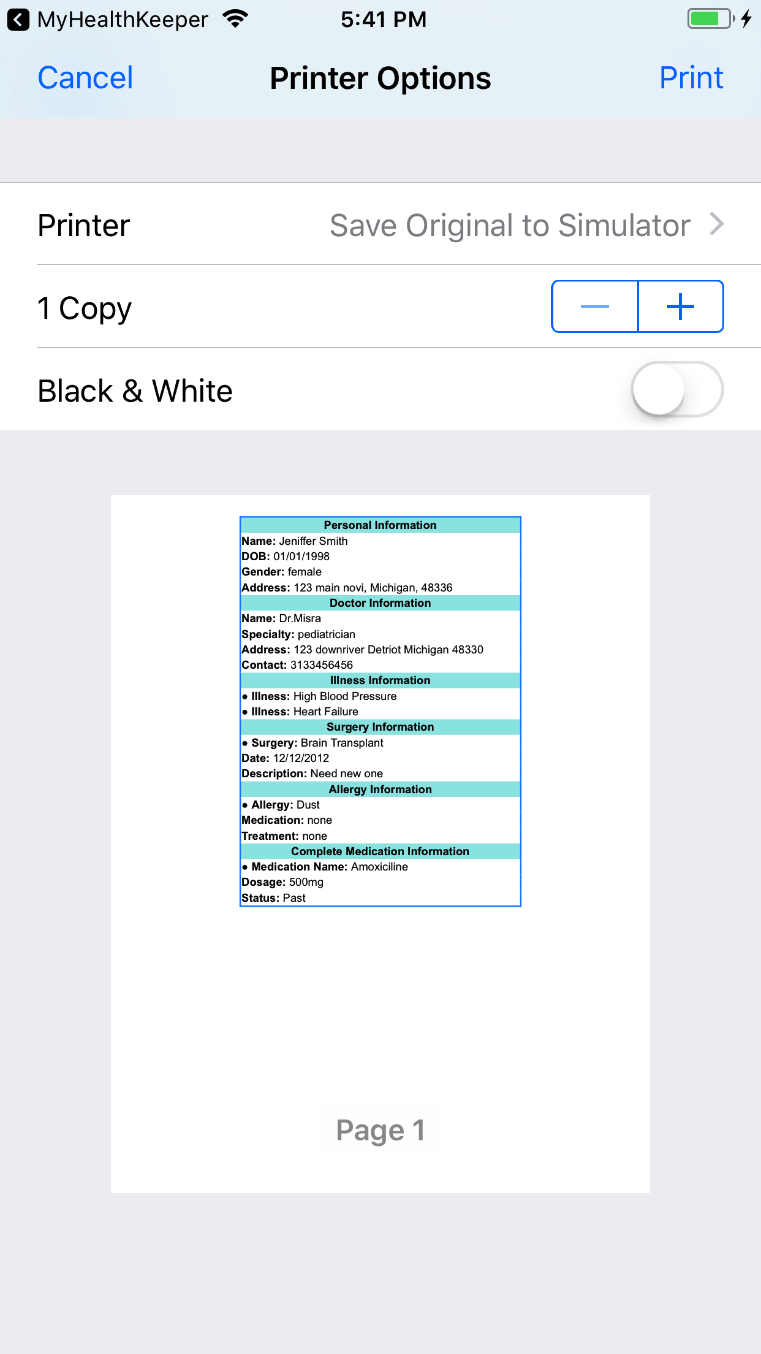
*Figure 20: Uploaded Documents*

If the user taps on the ‘Set Reminder’ button on the Menu or from the Main page, the user will be directed to the Add Reminder page as shown in *Figure 21*. The user then taps on the ‘Submit and Continue’ button and is directed to the Reminder Summary page as shown in *Figure 22.*

 ****

*Figure 21: Add Reminder*  *Figure 22: View Reminders*

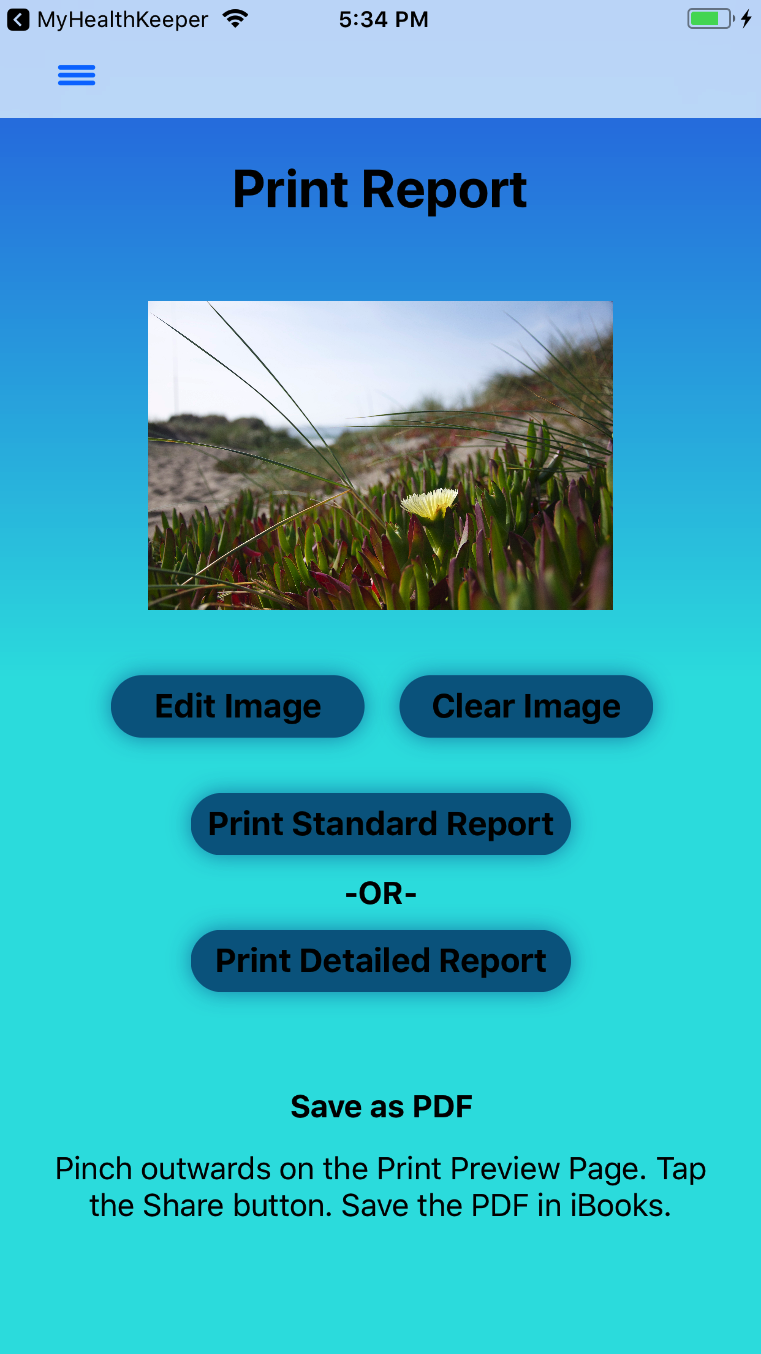
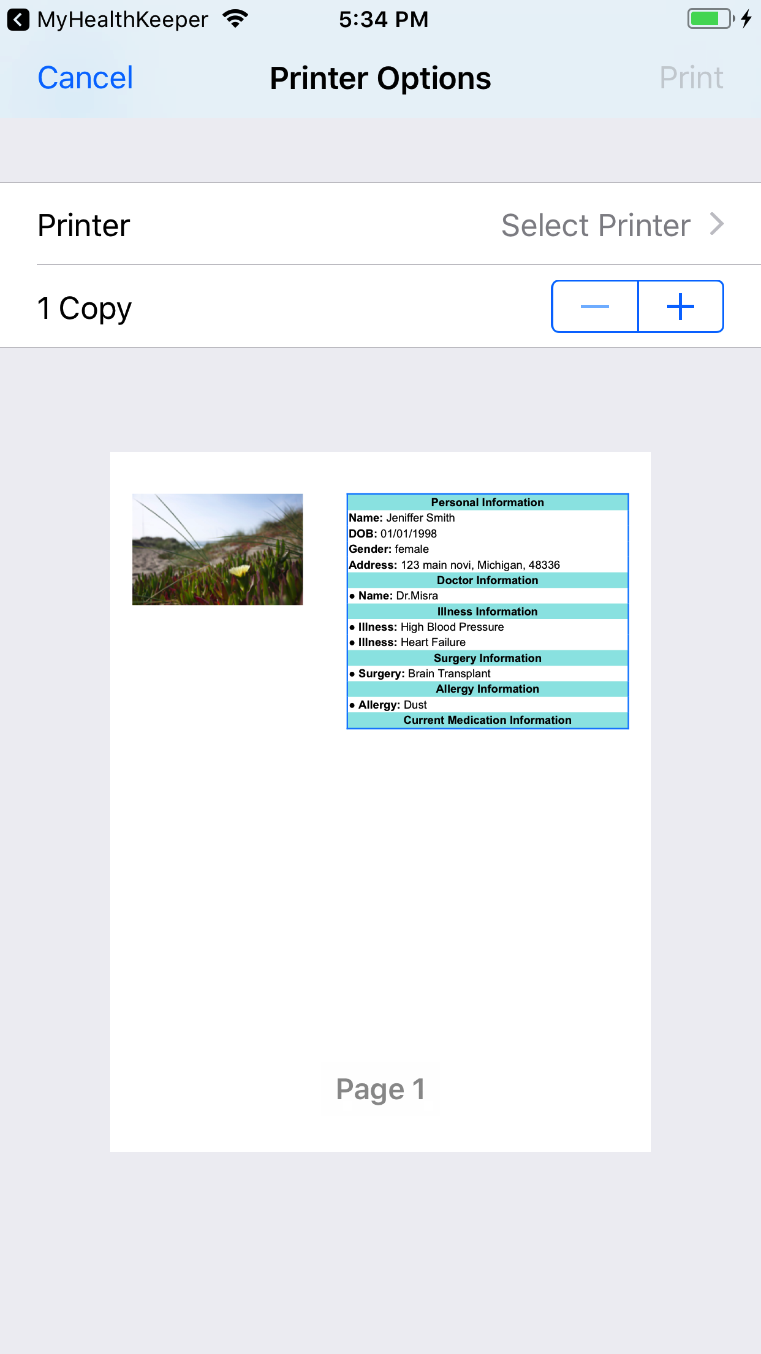
If the user taps on ‘Print’ in the Menu, they will be directed to Print Report page as shown in *Figure 23*. The user is presented with an alert to print their report with an image. If the user wishes to print the report without an image, the user taps ‘No’. If the user taps on the ‘Print Detailed Report’ button, the user will be directed to the Print page as shown in *Figure* 24.

*Figure 23: Print Report Figure 24: Printer Options*

*(Detailed Report without an Image)*

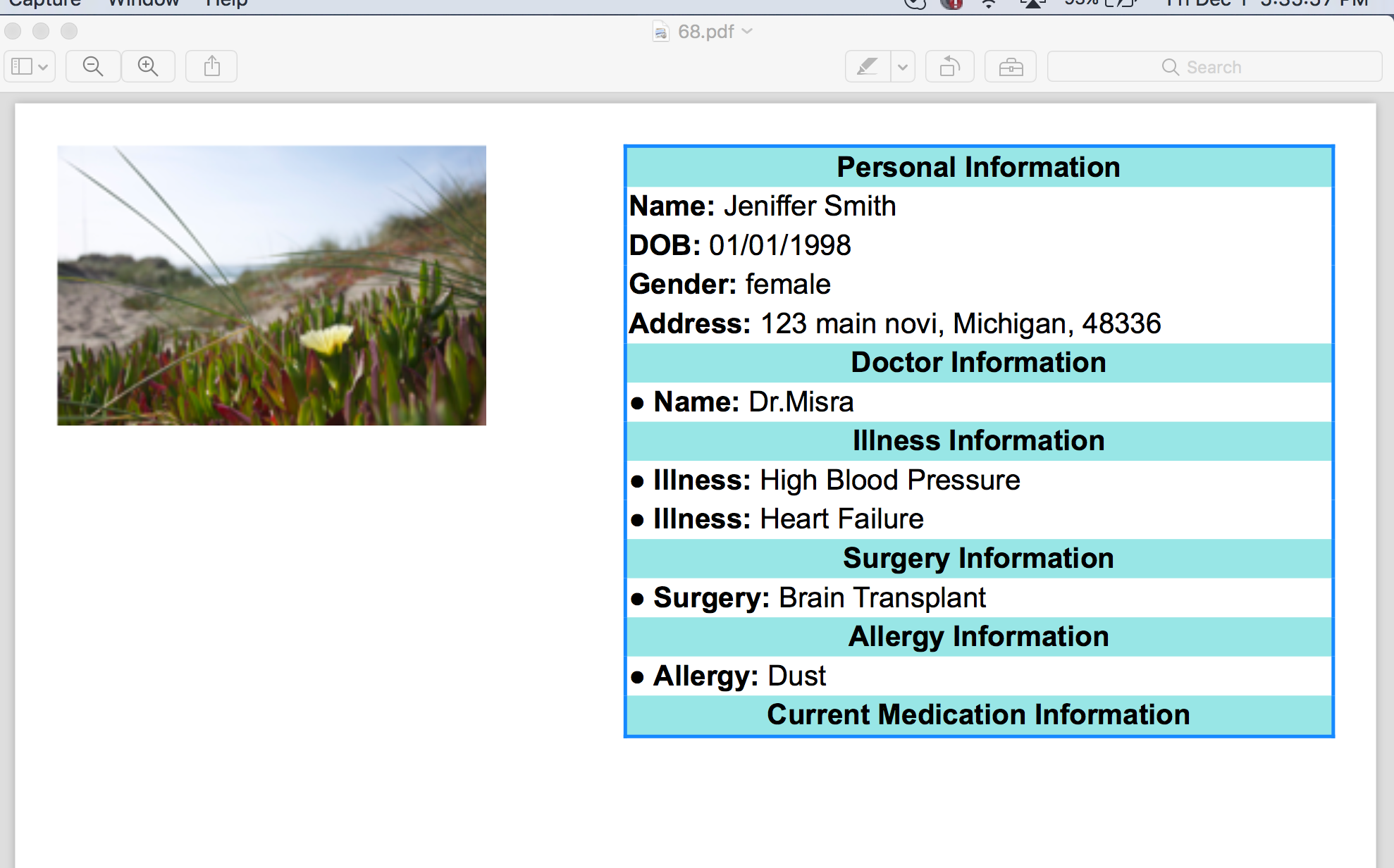
If the user decides to print the report with an image, the user will tap the ‘Yes’ button in the alert presented in *Figure 23.* The user will be directed to the Documents Summary page to select an image for their report. When selected, they are directed back to the Print Report page as shown in *Figure 25.* When the user taps on ‘Print Standard Report’. The user is directed to Printer Options page as shown in *Figure 26.*

*Figure 25: Print Report Figure 26: Printer Options*

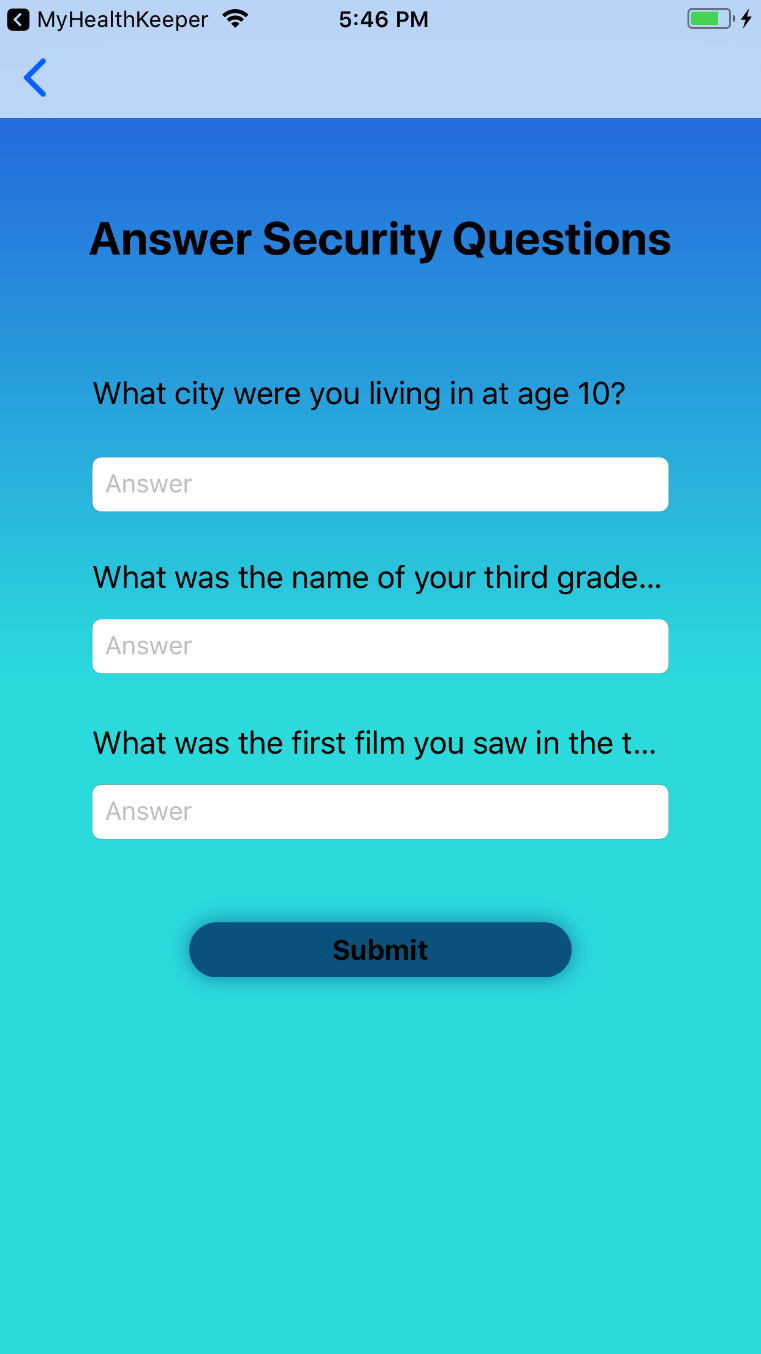
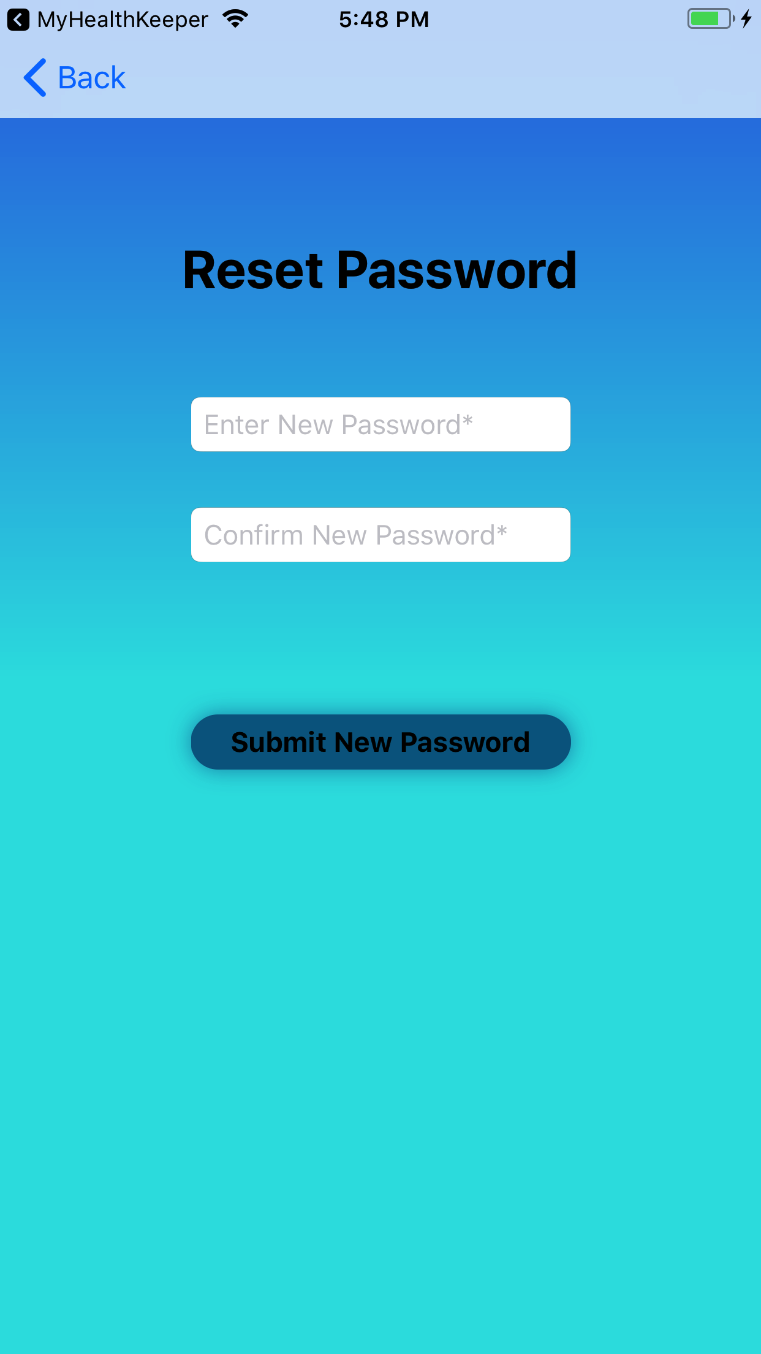
*(Standard Report without an Image)*

When the user taps ‘Print’ on the Printer Options page, they are directed to the ‘Print Preview’ page where the user can save the page as a PDF or print the page as shown in *Figure 27.*



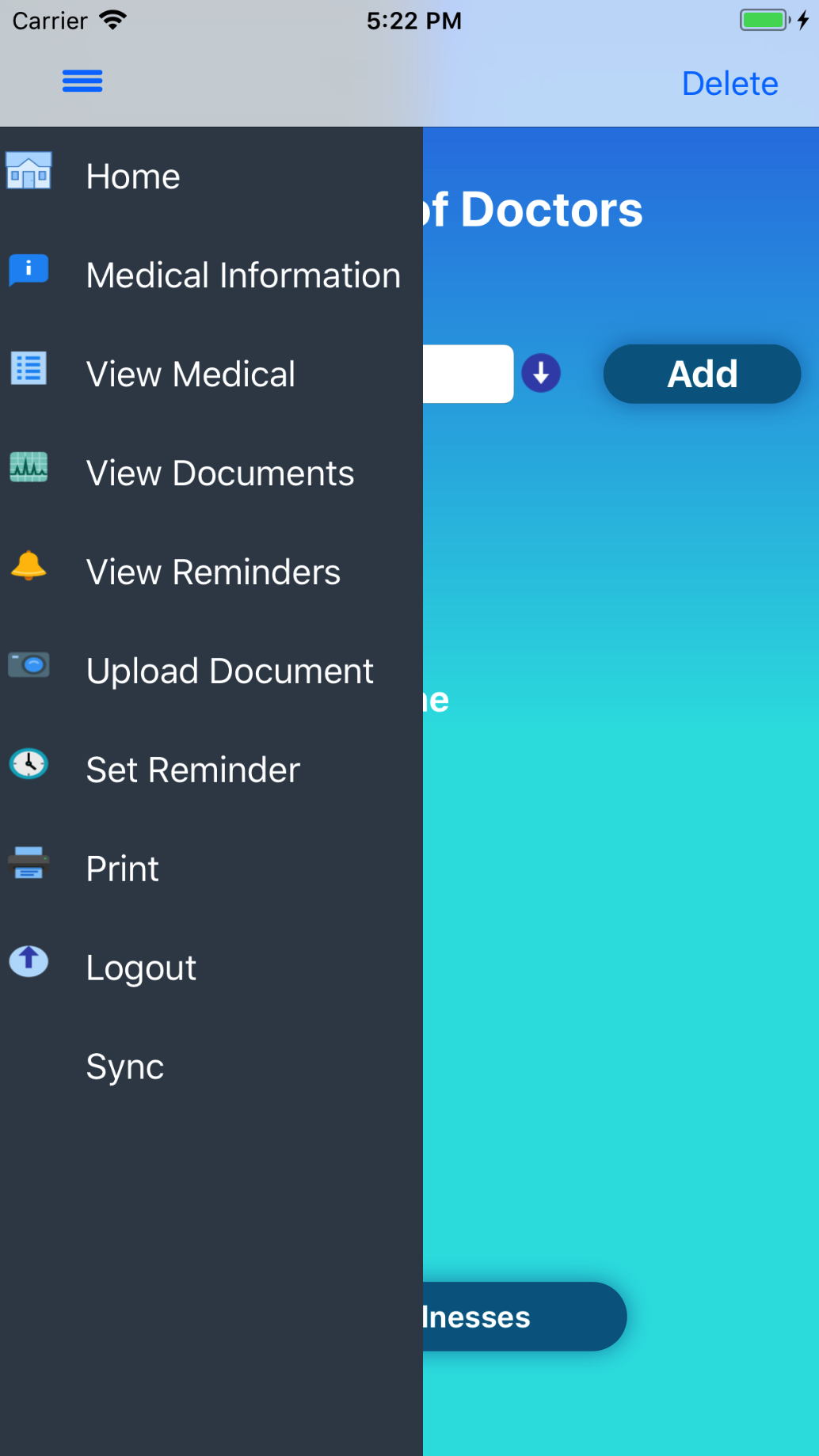
*Figure 27: PDF of Report*

If the user enters a valid username and taps on ‘Forget Password’ on the Sign In page, they are directed to the Answer Security Questions page as shown in *Figure 28*. After answering their security questions, the user taps on the ‘Submit’ button and they are directed to the Reset Password page as shown in *Figure 29*.

*Figure 28: Answer Security Questions Figure 29: Reset Password*

For the online version, if the user taps on the ‘Share’ button on the Print Preview page*,* they will be presented with nearby iOS devices that are connected to AirDrop. The user will be able to tap on a device name and send the PDF format of the medical data to the other device.

**

*Figure 30: Sync*

For the online version, once the user taps on the ‘Sync’ button in the menu as shown in *Figure 30*, they will be able to sync their data to the AWS server which will allow the user to retrieve their data from a different device.

### 3.1.2 Hardware Interfaces

MyHealthKeeper will run on an iOS device. The iOS devices suitable for this application are iPhone 5s and above with screen resolutions of 1136 x 640 pixels, and iPad 5th Generation and above with screen resolutions of 2048 x 1536 pixels [2]. MyHealthKeeper will also rely on the device’s camera to take a photo of a document that the user wishes to upload.

### 3.1.3 Software Interfaces

MyHealthKeeper will be designed to work for iOS devices running on iOS version 10.0 and above. The user must have a valid Apple ID and password to download the application from the App Store. In addition, the online application should also be able to POST and GET data to and from the AWS server.

### 3.1.4 Communications Interfaces

To ensure that the data stored on the offline application remains secure, we will be encrypting the data stored in the SQLite database. The encrypted SQLite database will be secured locally on the user’s iOS device. The database will be encrypted using a Pragma key.

The online application will send and receive data to a database running on an EC2 instance hosted by AWS. Data that is sent to the server will be protected using HTTPS. The online application will also utilize AirDrop for Share feature.

MyHealthKeeper will connect the iOS device’s camera and photo library. It will also send reminders to the user’s iOS device in the form of a notification.

## 3.2 Functional Requirements

This section will provide a detailed description of all Functional Requirements. It includes the inputs, processing, outputs, and error handling of each Functional Requirement.

Besides Functional Requirements FR24 and FR25, all other Functional Requirements are supported by both the offline and online applications. FR24 and FR25 are only supported by the online application.

**3.2.1 FR1: Registration**

**3.2.1.1 Introduction:** Allow the user to create an account for MyHealthKeeper.

**3.2.1.2 Inputs:** The new user will tap on the ‘Click to Register’ button presented on the first page of the application and will be directed to the Registration page. They will enter the following:

* First Name
* Last Name
* Username
* Password

The final field will be one of the following

* Email address
* Cellphone number

**3.2.1.3 Processing:** After each field is validated by their given constraints:

* First Name: field cannot be empty
* Last Name: field cannot be empty
* Username: 6 to 10 characters in length containing the following:
  + Letters [A-Z, a-z]
  + Integers [0-9] (optional)
* Password: field cannot be empty
* Email: field if field is not empty, email will only be accepted in the form of \_@\_.\_
* Cellphone number: if field is not empty, phone number will only be accepted in the form of ten digits

The user will tap on the ‘Next’ button, and the database will store this as the user’s account information.

**3.2.1.4 Outputs:** Once the user taps on the ‘Next Page’ button on the bottom of the Registration page, their account is created, and they are directed to the Setup Security Questions page.

**3.2.1.5 Error Handling:** If the text field constraints are violated, an error message will appear after the user taps outside the text field, or navigates to another text field, prompting the user to enter the fields according to the constraints. If the username already exists in the database, an error message will appear stating “ERROR Username already exists. Please enter a different username.”

**3.2.2 FR2: Setup Security Questions**

**3.2.2.1 Introduction:** Allow a new user to select and answer three Security questions after creating an account on the Registration page.

**3.2.2.2 Inputs:** The user will answer three Security questions displayed on the Setup Security Questions page in the designated text fields by selecting a question displayed in a picker on the bottom of the page.

**3.2.2.3 Processing:** The user will tap on a text field which displays a picker with three questions on the bottom of the page. The user scrolls the picker and taps on a question. The question will be displayed in the selected text field and the user will enter text in the ‘Answer’ text field. The Security questions and answers will be stored in the database.

**3.2.2.4 Outputs:** After the Security questions are answered in the text fields, the user will tap the ‘Finish & Sign In’ button, and they will be directed to the Home page.

**3.2.2.5 Error Handling:** If the user taps ‘Finish & Sign In’ without answering a Security question, an error message appears stating “ERROR One or more fields may be empty. Please enter a value.”

**3.2.3 FR3: Login to Application**

**3.2.3.1 Introduction:** Allow a registered user to log in to the application.

**3.2.3.2 Inputs:** The user will enter their Username and Password, and tap on the ‘Sign In’ button.

**3.2.3.3 Processing:** The user’s entered credentials are compared to the user’s account information in the database.

**3.2.3.4 Outputs:** The user’s credentials are accepted, and they are directed to the main page.

**3.2.3.5 Error Handling:** If the user inputs an incorrect username or password, an error message appears after the ‘Sign In’ button is tapped stating “ERROR Username or password is not valid. If you are a first-time user, please register.” If the username taps on ‘Sign In’ without entering a username or password, an error message appears stating “ERROR Please enter a username and password.”

**3.2.4 FR4: Forgot Password**

**3.2.4.1 Introduction:** Allow the user to reset their password by correctly answering the Security questions set up during Registration.

**3.2.4.2 Inputs:** If the user forgets their password, they will input their username in the provided text field and tap on ‘Forgot Password’. They will be directed to the Security Questions page to provide answers to the security questions they set up after creating their account.

**3.2.4.3 Processing:** The security questions associated with the username will be displayed. The user enters their answers in the provided text fields and taps ‘Submit’. The user’s entered answer is compared to the Security answers stored in the database.

**3.2.4.4 Outputs:** The answers are accepted, and the user is directed to the Reset Password page to enter text fields to reset and update their password. After entering a new password into the provided text fields, the user taps on ‘Submit New Password’ button and is directed to the Sign In page.

**3.2.4.5 Error Handling:** The user enters an incorrect answer in the Security Questions page, taps on ‘Sign In’, and an error message appears stating “ERROR Incorrect Security Answers”.

**3.2.5 FR5: Enter Medical Information**

**3.2.5.1 Introduction:** Allow the user to enter Personal, Medical, and Insurance information.

**3.2.5.2 Input:** If the user chooses to enter his or her medical history, the user will be directed to the Personal Information page to input the following:

* Last Name
* First Name
* Date of Birth
* Gender
* Street
* City
* State
* Zip code

Tapping on ‘Save and Continue’ directs the user to the Doctor Page to input the following:

* Name of the Doctor
* Specialty (Optional)
* Phone Number (Optional)
* Address (Optional)

Tapping on ‘Save and Continue’ directs the user to the Illness Page to input the following:

* Disease Name

Tapping on ‘Save and Continue’ directs the user to the Medication Page to input the following:

* Medication Name
* Medication Dosage
* Status

Tapping on ‘Save and Continue’ directs the user to the Surgery Page to input the following:

* Surgery Name
* Date (Optional)
* Description (Optional)

Tapping on ‘Save and Continue’ directs the user to the Allergy Page to input the following:

* Name
* Medication (Optional)
* Treatment (Optional)

Tapping on ‘Save and Continue’ directs the user to the Vaccine Page to input the following:

* Name
* Date (Optional)

Tapping on ‘Save and Continue’ directs the user to the Additional Page to input the following:

* Family History
* Note

Tapping on ‘Save and Continue’ directs the user to the Insurance Page to input the following:

* Type
* Name
* Group ID
* Expiration Date

For fields which require a date, the user must enter the text in the following format: mm/dd/yyyy. For fields which require a name, the user must enter only uppercase or lowercase letters up to a length of 30 characters with an exception of Illness name, Medication Name, Surgery Name, and Vaccine Name. The Surgery Description field is limited to 75 characters. For the Doctor Name field, the user can enter the special character “.” (ie Dr. Smith). For the Doctor Contact field, the user is only allowed to enter ten digits.

**3.2.5.3 Processing:** After entering information in the medical pages, the user will tap on ‘Save and Finish’, and the information will be saved in the database.

**3.2.5.4 Outputs:** Once the user taps ‘Save and Finish’, the user will be directed to the Medical Summary page.

**3.2.5.5 Error Handling:** If the user enters an invalid field, the user will get an error message which displays the valid format for that field.

**3.2.6 FR6: Update Medical Information**

**3.2.6.1 Introduction:** Allow the user to update the Medical Information they entered when tapping on the ‘Enter Medical History’ button.

**3.2.6.2 Input:** For Personal, Additional, and Insurance pages, the user deletes text displayed in the text fields by either removing it through the keyboard or by tapping the circular, Clear Field button on the right of the text field. The user then enters information into the text field and taps on the ‘Save and Continue’ button. For all other medical pages such as Doctor, Illness/Disease, Medication, Surgery, Allergy and Vaccine, the user taps ‘Edit’ on the top right corner of the page. A red, circular button is displayed to the left of the information they would like to edit. The user taps on it and taps on the red ‘Edit’ block displayed to the right of the information to be edited.

**3.2.6.3 Processing:** For Personal Information, Additional Information, and Insurance Information pages, the updated information will be stored in the database. For all other medical pages, the user will be directed to an Edit page which populates the selected information in the text fields. The user deletes the text displayed, and enters new information into the text fields. The user taps on the ‘Update’ button on the bottom of the page. The updated information will be saved in the database.

**3.2.6.4 Output:** For Personal, Additional, and Insurance pages, the user will be directed to the next page. For all other medical pages, an alert will appear stating “Edit Status Update was Successful”. After the information is updated, the data will be updated in the database and the user will be able to view their updated Medical Information.

**3.2.6.5 Error Handling:** If the user enters an invalid field, the user will get an error message which displays the valid format required for that field.

**3.2.7 FR7: Delete Medical Information**

**3.2.7.1 Introduction:** Allow the user to delete Medical Information that has been saved.

**3.2.7.2 Input:** For Personal Information, Additional Information, and Insurance Information pages, the user will tap the circular, Clear Field button on the right of each text field or erase the text displayed in the text field using the keyboard. For all other medical pages, the user will tap on the ‘Edit’ button, tap on the red, circular button displayed to the left of the information they would like to delete, then tap the gray ‘Delete’ block displayed to the right of the information to be deleted.

**3.2.7.3 Processing:** For Personal Information, Additional Information, and Insurance Information pages, medical information will be removed from the text field. For all other medical pages, the row with the selected information will be removed. Medical information will also be deleted from the database.

**3.2.7.4 Outputs:** Deleted information will be removed from the ‘Medical Summary’ page.

**3.2.7.5 Error Handling:** If the user deleted text from a required text field and taps on the ‘Save and Continue’ button, an alert will appear with the error message stating the valid format for the required field on the page.

**3.2.8 FR8: Upload Photo through Camera**

**3.2.8.1 Introduction:** Allow the user to take a picture of a medical document through the iOS device’s camera.

**3.2.8.2 Inputs:** The user taps on the ‘Camera’ button.

**3.2.8.3 Processing:** The iOS application connects to the iOS device’s camera. The user captures an image and taps ‘Use Photo’ button. The user is directed back to the Upload Document page.

**3.2.8.4 Outputs:** The image captured by the camera is displayed on the screen.

**3.2.8.5 Error Handling:** If a camera is not available on the iOS device, an alert appears stating “ERROR Camera is not available.”

**3.2.9 FR9: Upload Photo through Photo Library/Gallery**

**3.2.9.1 Introduction:** Allow the user to choose an image from the iOS device’s photo library to upload as a document.

**3.2.9.2 Inputs:** The user taps on the ‘Photo Library’ button.

**3.2.9.3 Processing:** The application connects to the iOS device’s photo library and displays the pictures stored in the library.

**3.2.9.4 Outputs:** The user taps on a picture from the photo library and the picture is displayed on the screen.

**3.2.9.5 Error Handling:** If photo library is not available, an error message appears stating “ERROR Photo Library is not available.”

**3.2.10 FR10: Save Document**

**3.2.10.1 Introduction:** Allow the user to save medical documents such as an EKG or blood work.

**3.2.10.2 Input:** The user will tap on the ‘Save’ button on the lower right corner which will present an alert and the user will enter the following fields:

* Name
* Description (Optional)

**3.2.10.3 Processing:** After the user enters the data, the user will tap on ‘OK’ which will save the image, to the application’s default directory as a JPEG file, and the file name in the database.

**3.2.10.4: Outputs:** The user will be directed to the Document Summary page.

**3.2.10.5 Error Handling:** If the user taps on ‘Save’ without an image uploaded, an alert will appear stating “ERROR Please upload an image.” If the user taps on ‘OK’ without entering a Document Name in the text field, an alert will appear stating “ERROR Document Name field cannot be empty. Please enter a value.”

**3.2.11 FR11: View Uploaded Documents**

**3.2.11.1 Introduction:** Allow the user to view the summary of all the medical documents uploaded.

**3.2.11.2 Input:** User taps on ‘OK’ in the alert displayed after uploading a picture of a document and providing a Name for the document.

**3.2.11.3 Processing:** The database will run SELECT queries which will gather all the information in the ‘Documents’ entity. The user is directed to the Document Summary page.

**3.2.11.4 Output:** The page displays information stored in the ‘Documents’ entity, such as Document Name, Description, and the uploaded image.

**3.2.11.5 Error Handling:** If there are currently zero documents stored in the database, an alert will appear stating “ERROR You have not uploaded any images.”

**3.2.12 FR12: Delete Document**

**3.2.12.1 Introduction:** Allow the user to delete a saved medical document which includes the Document Name, Description, and an uploaded image.

**3.2.12.2 Inputs:** The user will tap ‘Delete’ on the top right corner of the Document Summary page and then tap the red icon to the left of the image.

**3.2.12.3 Processing:** The specific tuple in the ‘Documents’ entity in the database will be deleted through an SQL statement.

**3.2.12.4 Outputs:** The row with the document image, name, and description will be removed from the page.

**3.2.12.5 Error Handling:** If the user did not upload any documents and the Document Summary page does not display any images, the ‘Delete’ button will not appear on the top right corner of the page.

**3.2.13 FR13: Add Reminder**

**3.2.13.1 Introduction:** Allow the user to set reminders for appointments and their iOS device will receive a reminder notification.

**3.2.13.2 Inputs:** The user will enter the following:

* Name: 1 to 30 characters in length
* Appointment location (optional)
* Appointment description (optional)
* Date
* Time

Once the reminder is set, the notification will appear on the user’s iOS device on the day and time the user specified.

**3.2.13.3 Processing:** After the user enters the required inputs, the system will verify if the information is valid (appointment name, date, and time fields are not blank). If it is valid, the system will create a new reminder tuple in the database.

**3.2.13.4 Outputs:** The reminder will be set and the system will send a notification at the date and time the user requested in the input.

**3.2.13.5 Error Handling:** If the user taps on ‘Submit and Continue’ or taps on the ‘Location’ or ‘Reason’ text fields without entering data into the ‘Name’ text field, an error message will appear stating “ERROR Reminder Name field cannot be empty. Please enter a value.”

**3.2.14 FR14: Monthly Reminder**

**3.2.14.1 Introduction:** This feature is automatically enabled to remind the user to update their medical information on the first day of every month at 12 pm. The user can disable this monthly reminder if they choose to do so.

**3.2.14.2 Inputs:** The user taps on a switch on the bottom of the Add Reminder page which will either leave the monthly reminder enabled, or disabled.

**3.2.14.3 Processing:** If the user triggers a change to disable the monthly reminders, the application will prevent the user from seeing these notifications. Otherwise, the user will continue seeing the monthly reminders on their iOS device.

**3.2.15.4 Outputs:** The user will receive notifications on their iOS device on the first day of every month at 12PM. The notification will state “Reminder to Update Medical Information Has any of your Medical Information changed in the past month? If so, please update your information on MyHealthKeeper.” If the user disables the monthly reminder feature, they will stop receiving notifications.

**3.2.14.5 Error Handling:** Unlike the appointment reminder feature, the monthly reminder feature cannot be deleted or edited. The user will only be able to enable or disable it.

**3.2.15 FR15: View Reminder(s)**

**3.2.15.1 Introduction:** Allow the user to view the reminders they set in the reminder page.

**3.2.15.2 Inputs:** The user taps on the ‘Submit and Continue’ button on the bottom of the page and then taps on the ‘View Reminders’ button on the alert presented.

**3.2.15.3 Processing:** The user will be directed to Reminder Summary page. The system will fetch data from the database and display it on the device’s UI.

**3.2.15.4 Outputs:** The reminders, with the name, date, and time, will be displayed on the ‘Reminder Summary’ page.

**3.2.15.5 Error Handling:** If the user did not set any reminders, an alert will appear stating “ERROR You have not set any Appointment Reminders.”

**3.2.16 FR16: Edit Reminder(s)**

**3.2.16.1 Introduction:** Allow the user to have the option to edit an already existing reminder.

**3.2.16.2 Inputs:** The user taps on the row which holds the Reminder Name, Date, and Time. The user taps on the ‘Yes’ button on the alert stating “Edit Reminder? Would you like to edit this reminder?” The user enters data into the provided text fields and taps on the ‘Update’ button.

**3.2.16.3 Processing:** The system will update the corresponding reminder in the database.

**3.2.16.4 Outputs:** The reminder information will be updated on the Reminder Summary page.

**3.2.16.5 Error Handling:** If the user deletes data from Name text field and taps the ‘Update’ button, an error message will appear stating “ERROR Reminder Name field cannot be empty. Please enter a value.”

**3.2.17 FR17: Delete Reminder(s)**

**3.2.17.1 Introduction:** Allow the user to have the option to delete existing reminders

**3.2.17.2 Inputs:** User taps the ‘Delete’ button on top right corner of the ‘View Reminders’ page. The user next taps on the red icon on the left of the reminder and then the ‘Delete’ button on the right of the reminder.

**3.2.17.3 Processing:** The system will remove the reminder entry from the database.

**3.2.17.4 Outputs:** The deleted reminder will no longer exist and will not appear in the ‘View Reminders’ page.

**3.2.17.5 Error Handling:** The user will receive an error message if the reminder cannot be deleted.

**3.2.18 FR18: View Full Medical History**

**3.2.18.1 Introduction:** Allow the user to view the full Medical Information Summary.

**3.2.18.2 Input:** User taps on ‘View Medical’ on the menu.

**3.2.18.3 Processing:** Tapping on ‘View Medical’ will select the data stored on the database.

**3.2.18.4 Output:** The user is able to view the Medical Summary page which consists of data the user entered in the medical pages.

**3.2.18.5 Error Handling:** If the user did not fill out any medical data and tries to view the medical summary, an error message will appear stating “ERROR You have not filled out any medical information.”

**3.2.19 FR19: Export Data to PDF Format**

**3.2.19.1 Introduction:** Allow the user to save data as a PDF document.

**3.2.19.2 Input:** User swipes out or uses 3D touch on the page displayed in the Print Preview page which will display the data in a PDF template. The user taps on the Share icon and then taps on ‘Import with iBook’.

**3.2.19.3 Processing:** The iOS device will convert the data to a PDF document.

**3.2.19.4 Outputs:** The user’s Medical Information and an image (if the user selected one) will be downloaded and saved as a PDF in iBook on the user’s iOS device.

**3.2.19.5 Error Handling:** The user cannot save their data as any other formatted document. Exporting the data as a PDF will be the only given option.

**3.2.20 FR20: Printing Report**

**3.2.20.1 Introduction:** Allow the user to print their Medical Information and an uploaded document.

**3.2.20.2 Inputs:** The user taps on ‘Print’ on the Menu.

**3.2.20.3 Processing:** After the user taps on ‘Print’, this process will retrieve selected fields from the ‘Medical Summary’ page and an image (if selected) from the database. The medical information will be displayed on the ‘Printer Options’ page. When the user taps ‘Print’ on the ‘Printer Options’ page, the iOS device will locate and connect to an available printer via AirPrint, and the data will get sent to the printer.

**3.2.20.4 Outputs:** Medical Information will be printed on an 8x11 paper.

**3.2.20.5 Error Handling:** If the iOS device cannot connect to a printer, a message will appear stating “No AirPrint Printers Found”. This error handling is provided through the AirPrint API.

**3.2.21 FR21: Connecting iOS Devices**

**3.2.21.1 Introduction:** Allow the user to share MyHealthKeeper data from one iOS device to another.

**3.2.21.2 Inputs:** Usertaps on the ‘Share’ icon on the Preview page which displays the medical data and an optional, selected image on a PDF template.

**3.2.21.3 Processing:** AirDrop will use Bluetooth to look for another nearby iOS device that it can share the PDF with. The user taps on the iOS device’s name to connect.

**3.2.21.4 Outputs:** A PDF containing the patient’s Medical Information will be sent to the receiving iOS device.

**3.2.21.5 Error Handling:** If the user taps the ‘Share’ icon and is not connected to Bluetooth or Wi-Fi, a message will appear stating “Tap to turn on Wi-Fi and Bluetooth to share with AirDrop.” This error handling is provided through the AirDrop API.

**3.2.22 FR22: Sync Data to/from Server**

**3.2.22.1 Introduction:** Allow the user to sync the data stored in the local database to a server.

**3.2.22.2 Input:** User will tap the ‘Sync’ button on the Menu

**3.2.22.3 Processing:** The application will send the user’s data to the server or receive the user’s data from the server and store it in the user’s device.

**3.2.22.4 Outputs:** The user will have synced their data to/from the server.

**3.2.22.5 Error Handling:** An error message will appear stating “Syncing data was unsuccessful”.

**3.2.23 FR23: Menu**

**3.2.23.1 Introduction:** Allow the user to navigate to different pages in the application through a slide-in menu.

**3.2.23.2 Inputs:** The user taps on the Hamburger Menu icon on the top left corner of the page, the application provides a list of pages to view, and the user taps on a page for viewing.

**3.2.23.3 Processing:** The application accepts the user’s request to tap on the menu, provides a list of pages to view, and accepts the user’s request to navigate to another page.

**3.2.23.4 Outputs:** Leaves page that the user started on, and navigates to a different page.

**3.2.23.5 Error Handling:** If the user is on a Summary page with the Menu open and starts scrolling, the menu will automatically close to allow scrolling to be enabled.

**3.2.25 FR24: Logout**

**3.2.25.1 Introduction:** Allow the user to log out of their account on MyHealthKeeper.

**3.2.25.2 Inputs:** The user will tap on the Menu, and tap ‘Logout’.

**3.2.25.3 Processing:** Tapping on ‘Logout’ will trigger the app to go to the ‘Sign In’ page

**3.2.25.4 Outputs:** The user will be logged out, and the app will display the ‘Sign In’ page

**3.2.25.5 Error Handling**: On the online application, to prevent the user from unintentionally logging out without syncing their latest changes to the server, an alert will appear stating “Would you like to save your changes before you log out?” If ‘Yes’, the user’s data will sync to the server and then the user will be logged off. If ‘No’, the user will be logged off.

## 3.3 Non-Functional Requirements

This section will provide an explanation for each non-functional requirement. The section will also include the title, requirement, and description of each non-functional requirement.

### 3.3.1 Performance

**ID: NFR 3.3.1.1**

**Title:** Database Performance

**Requirement:** The online and offline applications should retrieve data from the database within 3 seconds.

**Description:** All requests that are connected to the database are completed within 3 seconds.

**ID: NFR 3.3.1.2**

**Title:** File Upload Time

**Requirement:** All documents uploaded to MyHealthKeeper should occur within 1 minute.

**Description:** Documents, such as an image of an EKG, are uploaded to MyHealthKeeper within 1 minute.

### 3.3.2 Reliability

**ID: NFR 3.3.2.1**

**Title:** Database Error Handling

**Requirement:** Database should handle the process of saving and displaying data.

**Description:** When the user requests information that is not in the database, an error message will appear 99% of the time.

3.3.3 Availability

**ID: NFR 3.3.3.1**

**Title:** Offline Application Use Availability

**Requirement:** The offline iOS application must be downloaded in order for the user to access it at any time on an iOS device.

**Description:** The user will be able to use the iOS application any time after it has been installed.

**ID: NFR 3.3.3.2**

**Title:** Online Application Use Availability

**Requirement:** The online application must be connected to the AWS server in order for the user to access it at on an iOS device.

**Description:** The user will be able to use the online application if it is connected to the server, and the user has valid credentials.

### 3.3.4 Security

**ID: NFR 3.3.4.1**

**Title:** Local SQLite Database for Offline Application

**Requirement:** Data will be stored on an embedded database.

**Description:** The application will save data to a local SQLite database on the user’s iOS device.

**ID: NFR 3.3.4.2**

**Title:** SQLite Data Encryption

**Requirement:** Data saved in the database will be encrypted using SQLCipher.

**Description:** All data sent from MyHealthKeeper to the database will be encrypted to ensure the data is secure.

**ID: NFR 3.3.4.3**

**Title:** HTTPS for Online Application

**Requirement:** The online application will use the HTTPS transfer protocol.

**Description:** All data that MyHealthKeeper sends to the server should be sent through HTTPS.

**ID: NFR 3.3.4.4**

**Title:** User Authentication

**Requirement:** Only authenticated users can access MyHealthKeeper.

**Description:** A user should only be able to access the application if they have the correct permissions. This includes a valid username and password.

### 3.3.5 Maintainability

**ID: NFR 3.3.5.1**

**Title:** Online Application Maintenance Time

**Requirement:** Maintenance to the online application will occur during non-peak hours.

**Description:** Limited resources will be available during maintenance time.

**ID: NFR 3.3.5.2**

**Title:** Addition/Updates to Application Features

**Requirement:** MyHealthKeeper will be able to add, remove, and update features as necessary.

**Description:** Features can be added and removed as required. This may be for security purposes or improvement in quality.

### 3.3.6 Portability

**ID: NFR 3.3.6.1**

**Title:** iOS Compatibility

**Requirement:** MyHealthKeeper should be compatible on iOS devices.

**Description:** The iOS application and all of its features should be 100% visible on iOS devices such as iPhone 5s and above, iPad 5th Generation and above, and have an operating system of version 10.0 or higher.

**ID: NFR 3.3.6.2**

**Title:** Screen Resolution

**Requirement:** MyHealthKeeper should display its contents on screens for iPhone 5s and above and for iPad 5th Generation and above.

**Description:** The application and all its features should be 100% visible on iPhone 5s with screen resolutions of 136 x 640 pixels and iPad 5th Generation and above with screen resolutions of 2048 x 1536 pixels.

## 3.4 Design Constraints

The offline application should not use any server or external database. This constraint is necessary for the application to be HIPAA compliant. It should also be deployed as an iOS application in the App Store.

The online application will require the use of a server. This is being done to give the user the option to store and sync their data across any iOS device that has the application installed.

## 3.5 Logical Database Requirements

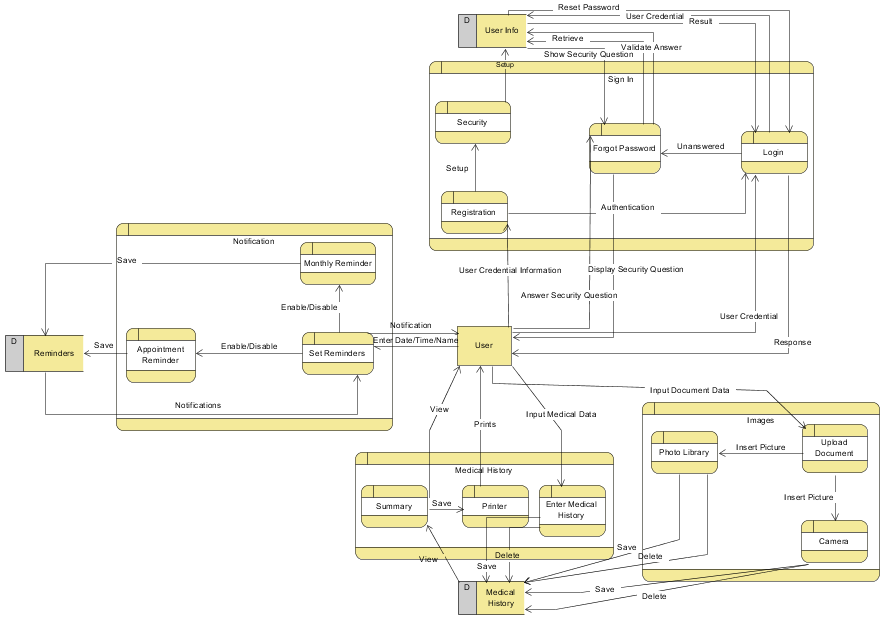
The offline application will be storing the data in an embedded SQLite database that is stored on the user’s iOS device. Since the development team is coding the iOS application in Swift and SQLite databases are not compatible with Swift code, an open source Objective-C wrapper called FMDB will be incorporated into the code. This wrapper will allow the use of Swift code to manage the SQLite database.

The online application that utilizes a server will store data to a MySQL database. The server will send the encrypted data the user enters to the MySQL database on the server.

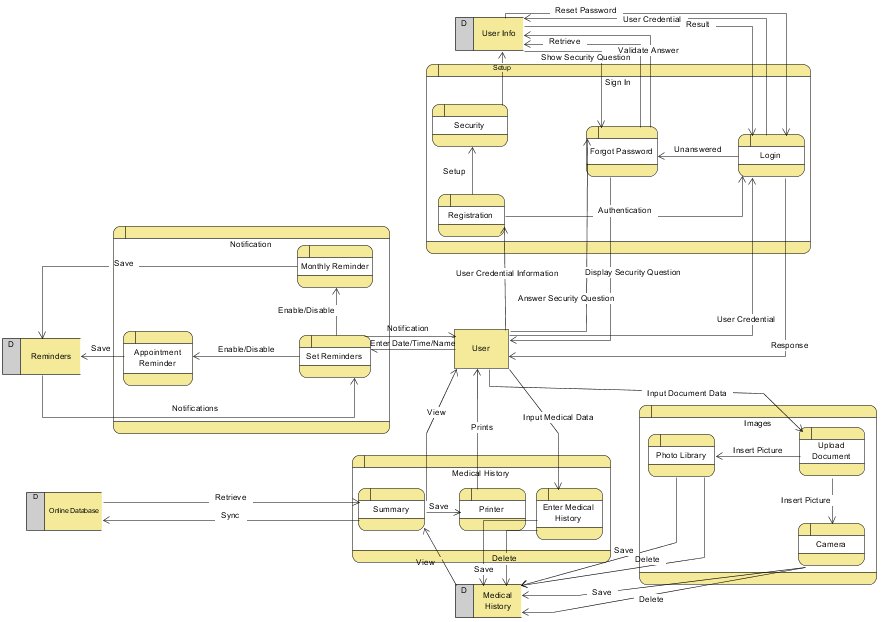
# 4. Analysis Models

## 4.1 Data Flow Diagrams

The diagrams below are level two data flow diagrams for both the offline and online applications.



*Figure 31: Data Flow Diagram Offline Version*



*Figure 32: Data Flow Diagram Offline Version*

# Appendix

## Requirements Traceability Matrix

The following traceability matrix will be used to trace each requirement throughout the application development. Each requirement will have a use case and test cases associated with it.

|  |  |  |
| --- | --- | --- |
| **Requirement ID** | **Name** | **Priority** |
| FR1 | Registration | High |
| FR2 | Setup Security Question | Medium |
| FR3 | Login to Application | High |
| FR4 | Forgot Password | Medium |
| FR5 | Enter Medical Information | High |
| FR6 | Update Medical Information | High |
| FR7 | Delete Medical Information | High |
| FR8 | Upload Photo through Camera | High |
| FR9 | Upload Photo through Photo Library/Gallery | High |
| FR10 | Save Document | High |
| FR11 | View Uploaded Documents | High |
| FR12 | Delete Documents | High |
| FR13 | Add Reminder | Medium |
| FR14 | Monthly Reminder | Medium |
| FR15 | View Reminder(s) | Medium |
| FR16 | Edit Reminder(s) | Medium |
| FR17 | Delete Reminder(s) | Medium |
| FR18 | View Full Medical History | High |
| FR19 | Export Data to PDF Format | Low |
| FR20 | Printing Report | Medium |
| FR21 | Connecting iOS Devices | Medium |
| FR22 | Sync Data to/from Server | Medium |
| FR23 | Menu | High |
| FR24 | Logout | High |

|  |  |  |
| --- | --- | --- |
| **Non-Functional Requirements** | **Name** | **Priority** |
| NFR3.3.1.1 | Database Performance | Medium |
| NFR3.3.1.2 | File Upload Time | Medium |
| NFR3.3.2.1 | Database Error Handling | High |
| NFR3.3.3.1 | Offline Application Use Availability | High |
| NFR3.3.3.2 | Online Application Use Availability | Medium |
| NFR3.3.4.1 | Local SQLite Database for Offline Application | High |
| NFR3.3.4.2 | SQLite Data Encryption | High |
| NFR3.3.4.3 | HTTPS for Online Application | Medium |
| NFR3.3.4.4 | User Authentication | High |
| NFR3.3.5.1 | Online Application Maintenance Time | Medium |
| NFR3.3.5.2 | Addition/Updates to Application Features | Medium |
| NFR3.3.6.1 | iOS Compatibility | High |
| NFR3.3.6.2 | Screen Resolution | High |