2018

Personal Health Card: Use of QR Code to Access Medical Data

Kamran Imam
University of Rhode Island, ikamran29@gmail.com

PERSONAL HEALTH CARD:

USE OF QR CODE TO ACCESS MEDICAL DATA

BY

KAMRAN IMAM

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

COMPUTER SCIENCE

UNIVERSITY OF RHODE ISLAND

2018

MASTER OF SCIENCE THESIS

OF

KAMRAN IMAM

APPROVED:

Thesis Committee:

Major Professor Noah Daniels

Victor Fay-Wolfe

Donna Gamache-Griffiths

Nasser H. Zawia
DEAN OF THE GRADUATE SCHOOL

UNIVERSITY OF RHODE ISLAND
2018

ABSTRACT

A communication and information disconnect exists between healthcare providers, care recipients and family caregivers. Poor communication synchronization of follow-up treatments, lab results, prescribed medication between providers, patients and their caregivers exists during transitions of care. For example, a patient's primary care physician and caregiver may not be notified during a hospital discharge process. As a result, poor coordination of care can lead to an increase in medical costs, negative patient care outcomes or risk of hospital readmissions. This design implements a personal health card to access general medical data with the use of QR (Quick Response) code technology. The use of QR code technology integrated with a mobile messaging application is suggested as a potential solution in improving communication and sharing of data between healthcare providers, caregivers and care recipients.

ACKNOWLEDGMENTS

I would like to thank my major professor, Dr. Noah Daniels, for his unwavering support and knowledge. I am highly appreciative of his time, guidance and expertise shared throughout this research. I would also like to thank my two additional committee members, Dr. Victor Fay-Wolfe and Dr. Donna Gamache-Griffiths, for their commitment and support provided over the course of this research. This research would not have been made possible without each committee member's participation, input and validation which, I am very thankful for. Lastly, I would like to thank the staff and faculty in the Department of Computer Science and Statistics, College of Business Administration and the Graduate School for the outstanding work they do each day.

TABLE OF CONTENTS

LIST OF TABLES

TABLE **PAGE**

LIST OF FIGURES

FIGURE **PAGE**

CHAPTER 1

INTRODUCTION

The inability to properly coordinate care of a patient has resulted in hospital readmissions, higher medical costs and lower qualities of care. The two main aspects of properly coordinating care involves the ability to communicate with various parties and the ability to share accurate data [1]. Healthcare providers today can be seen to coordinate care with the use of shared data sources, phone calls, faxes, postal mail delivery or text messaging. These existing solutions do not accurately support the proper platform needed to properly coordinate care. For example, phone calls can be challenging in order to communicate with healthcare providers due to wait times, incorrect resources being reached or miscommunication over a weak line.

In this project we seek to improve care coordination by implementing a mobile messaging application equipped with QR code technology. A QR code is a machine-readable barcode used in the application of product tracking, marketing, document management, item identification and more. The benefits include efficiency in storing data, greater data storage and quick readability. An implementation of QR code technology which can be used to store general medical data and be scanned to display the general medical data to improve the coordination of care is presented.

1.1 Goals

Due to time sensitive circumstances, the goal of this project is limited to a completed implementation and a plan for future beta testing. This project will implement a mobile application utilizing QR code technology to store and read general medical data. Along with the implementation of this project, a plan for future beta testing will be structured to verify the application's proof of concept. The implementation includes the following:

1. An alpha test plan will test all core application features with each test case and actual result where all cases pass via Apple TestFlight.
2. A beta test plan will be formed for future pilot testing purposes to verify the application's proof of concept.
3. A landing page will be formed which will include FAQs, Privacy Policy, Terms of Service and contact information.
4. A new user will have the ability to login as a new account or returning user.
5. A user will have the ability to fill in, edit and save their general medical, caregiver and primary care physician information within the app.
6. A user will have the ability to view their QR code image. By scanning the QR code image within the app, the user's general medical, caregiver and primary care physician information will be displayed.
7. When a user assigns a caregiver to their QR, the caregiver's account will show that they are listed as a caregiver of the user.

8. The caregiver will have the ability to edit the user's general medical information as well in order to reflect up-to-date information when the user's QR code is scanned.

1.2 Results

The implementation described in this project was successful in meeting the goals set above. The implementation was able to pass all alpha test cases intended for the application to be considered as a successful implementation. A beta test plan was also developed for future pilot testing to validate a proof of concept of the implementation. In the future, further development will be assessed to build upon the Android platform.

1.3 Outline

The remainder of this effort is described in the following sections. The Background section discusses the challenges in healthcare and existing technologies which attempt to improve care coordination. The Methods section provides a detailed explanation of the RESTful API, QR code technology, and alpha test plan which has been implemented. The Results section covers the successfully passed alpha test cases. Lastly, the Discussion section considers future work of this implementation which includes a beta test plan.

CHAPTER 2

BACKGROUND

The healthcare system continues to be an ever-changing model due to rising costs and inconsistent quality delivered to patients despite the skillful and countless efforts of providers. The overall goal for healthcare providers has been to maximize value for patients by achieving the best outcome at the lowest cost [1]. One aspect of the issue is seen as the difficulties in the coordination of care. An article by the American Academy of Nursing Policy indicates, "the best coordination model is one in which a patient experiences primary care as delivered by an integrated, multidisciplinary team that includes at least one care coordinator staff person [2]." When a patient, referred to as a care-recipient, receives care the care-recipient may visit multiple healthcare providers such as their PCP (Primary Care Physician), a lab testing facility, a hospital, or a specialist. Additionally, a care-recipient may also have a caregiver, known as one that provides assistance and care to others such as the elderly or a disabled family member. In this case, it is important for a care-recipient's caregiver, healthcare providers, and the care-recipient to be aligned in order to deliver the best quality of care and the lowest cost. In this section, we discuss certain challenges in the coordination of care. We then conclude this section by discussing some of the technologies which currently exist in supporting the coordination of care and how our own implementation will help.

2.1 Challenges of Coordinating Care

Traditionally, healthcare systems have always been developed as siloed architectures. Patient data in one system or one department may not exist in another system or department. An article by Harvard Business Reviews outlines six essential elements required in a value-adding IT platform [1]. First, the system needs to be centered on patients. Unfortunately, it is a challenge to understand and view a patient's entire medical history due to fragmented data across various sites and services. Data need to be aggregated around the patient rather than being aggregated around providers or departments. Next, the article supports a use of common data definitions. There may be certain specialized terms used in various departments, thus, a need for standardized measures and terminology should be established. For example, the billing department utilized a set of standard codes while a problems list utilizes a different set of standards for documentation purposes. Here, it is a challenge to ensure that terminologies are correctly utilized and analyzed when producing analytical reports. Another element of a value-based platform is the need to encompass all type of patient data such as lab tests, images or physician notes. Here, it is a challenge to gather and identify patient data as it may reside in another system outside of a provider's view. The last three elements which the Harvard Business Review discusses are the need for a platform where the medical record is accessible to all parties, a system which includes templates for each medical condition, and a system which makes it easy to extract information

[1]. These last three elements explain the need to encourage shared data and offer a standardized approach which will align providers. Additionally, by developing an architecture which allows data extraction, providers will be able to process historical data in order to make decisions based on the extracted information. Current methods of data extraction prove to be very challenging. For example, if a patient would like to request their own medical records, a written request may be required along with an additional fee in some practices. This process then takes additional time for the healthcare provider to process the request in order to release the medical records to the care-recipient. Thus, our implementation is motivated by each of these challenging elements to improve the coordination of care.

Another challenging aspect of coordinating care is the rising population of caregivers and care-recipients. A fact sheet provided by the Family Caregiver Alliance states that a population in the United States of those in the age of 65 and older is expected to double to 70 million by 2030, thus, family caregivers will also increasingly provide care for parents [3]. Additionally, most caregivers are employed with an estimated 60% working full or part-time. While most caregivers live near the people they care for, 61% live up to a one hour drive away while 15% live a 1-2 hour drive or more away [3]. Based on these statistics, caregiving will continue to be a challenging duty as the population of the elderly increases. Thus, it will be important to provide caregivers with the necessary tools in order to effectively and efficiently manage the care of the elderly.

Another important aspect of improving the coordination of care involves cost savings. A study by the American Academy of Nursing on Policy describes a previous study in which the analysis of over nine million Medicaid and dual Medicare/Medicaid patient claims records are studied to determine patters and costs associated with uncoordinated care [2]. The analysis found that uncoordinated care patients accounted for 46% of drug costs, 32% of medical costs, and 36% of total costs. Additionally, on average, uncoordinated care patient costs were 75% higher than those compared with patients whose care was coordinated. As a result, the report concluded to explain that care coordination could result in an annual savings of $240.1 billion per year if care coordination were implemented in public and private plans nationally as described in the study [2]. In general, the motivation behind this implementation has the ability to provide a large scale impact in various aspects such as improved patient satisfaction, improved health outcomes, and reduced costs. We hope that this implementation decreases unnecessary or multiple lab tests due to miscommunication. Additionally, we hope that this implementation will allow providers to follow-up with patient's on the the outcome of their health. Lastly, we hope that this implementation will provide patients and their caregivers with an efficient process of communicating with health providers.

2.2 Related Works

Based on market research, the landscape of healthcare continues to evolve where an increasing number of start-up companies are devoted to improving patient

care through digital health products such as wearable devices. Additionally, larger corporations are being consolidated in order to increase costs savings and streamline processes. In regards to improving the coordination of care, various start-ups are focused on aggregating data sources or developing a social platform for patients and their caregivers. Unfortunately, the solution of aggregating data sources is extremely complex and infeasible given that thousands of data sources could exist and would require the company to reach out to every new provider in order to integrate a new data source into the application. An example of one start-up company aggregating data sources into a single interface is known as, "Prime," which is currently no longer available [4]. Another application known as, "ICE Medical Standard," claims to have over 100 million users in the United States. ICE Medical Standard is a mobile application which enables users to input their health information into their app and set the information provided as part of the user's smartphone lock screen. Thus, in case of an emergency, others will then be able to view a user's health information by viewing the user's locked phone. The company also provides physical printed Medical ID cards as well [5]. Although, this can be helpful in an emergency situation, this solution does not integrate a messaging functionality. Additionally, this solution requires that the information be shown on a phone's lock screen which may be not be desired by some who would not like to share their medical information to be seen by others accessing the phone's lock screen. While most start-ups continue to innovate in healthcare with the use of artificial intelligence and wearable devices, data will continue to be fragmented

between systems as more data sources are created. We have mentioned the motivation behind this implementation and how other companies are improving patient care outcomes. In the next chapter, we discuss the implementation and describe various aspects of the implementation in detail.

CHAPTER 3

METHODS

In this chapter, we discuss the methods used to satisfy our goals of the implementation. We first begin by discussing the overall architecture of the implementation. Throughout this section, we provide figures of the application's user interface taken as a screenshot from the iPhone via FlightTest. We then focus in on the core functionality of the implementation which is, the use of RESTful APIs, QR code technology, and security. Lastly, we introduce a set of core questions developed for alpha testing in order to ensure that our implementation satisfies each of our goals.

3.1 Architecture

The project is implemented as an iOS application written in Apple's Swift 3 programming language. Therefore, this current application is implemented for use on Apple mobile devices only. The implementation was designed based on requirements which would be necessary in order to improve the coordination of care. A previous messaging application was recycled for this implementation in order to be efficient with our time and costs. In Chapter 5, we discuss future work and take on an agile approach to continue reiterating our implementation based on results from beta testing. We begin to explain the architecture by discussing the application's stack.

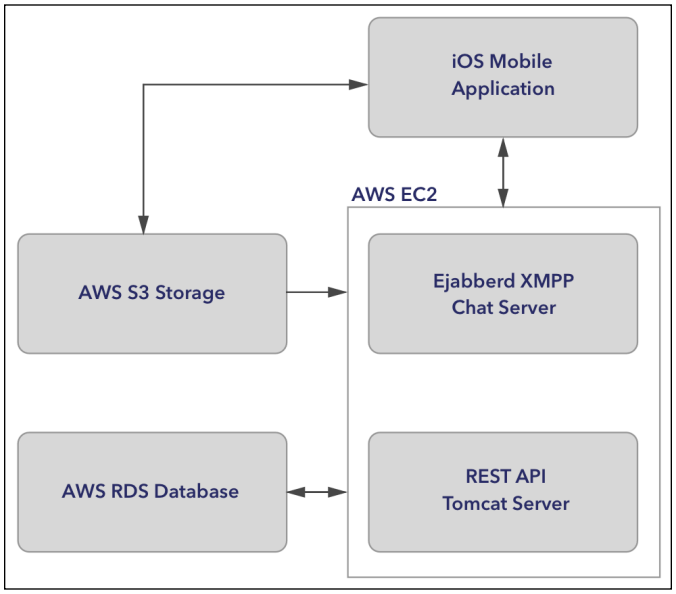

Figure 1. Architecture

As shown in Figure 1, the client side of the implementation communicates with the Application Programming Interface (API) Server and Chat Server being run on an instance of the Amazon Web Services EC2 platform. User authentication occurs each time the client is restarted by verifying with the chat server if the user is logged in. Amazon Web Services S3 is being utilized to store user profile photos which is directly uploaded by the client with the use of a secure access token processed from the API server to AWS S3. As mentioned earlier, the chat server and user login existed from a previously recycled project and is handling the messaging capabilities between users with the use of an XMPP framework communicating via XML. Here, the chat server maintains data by communicating with the Amazon Web Services Relational Database Service. The API server communicates with the use of Representational State Transfer (REST) API which sends responses in JavaScript Object Notation (JSON) format. The API server involves three overall

tasks. As we mentioned, the API server communicates with Amazon Web Services S3 to generate a secure upload token for the client in order to store the user's profile photo in the S3 bucket. Next, the API server handles user management and QR code logic by communicating with the Amazon Web Services RDS database. Lastly, the API server utilizes Google's ZXing library to generate QR code images from a string of binary data. A MySQL database instance exists on Amazon Web Services RDS platform. Here, the database contains tables for the storage of user account and QR code data being accessed by the servers. Overall, we utilize Amazon Web Services and continue to pay a monthly invoice based on the usage with rates set by Amazon. Now that we have described the architecture of our implementation, in the next section of this chapter, we discuss the implementation of the core functionalities.

3.2 Implementation

3.2.1 RESTful API

We utilize an open source framework known as Jersey RESTful Web services in Java for implementation. Our RESTful API resource is the health data being updated and viewed by users. The REST API was developed for ease of use and in order to enable capabilities of automating and connecting with other Electronic Health Records systems in the future. The API provides access to an individual's health data in addition to making updates by the user. The uniform resource identifier (URI) being utilized in order to access and update a QR code

resource is *http://server/qrCode/{id}*. In section A of the Appendix, we include a snippet of the Java model being used as the representation of our properties for our medical profile resource. The actions available for use are POST, GET, PUT, and DELETE. In order to access the medical information of a specific resource, we utilize the GET action and specific the resource's ID within the endpoint. In section B of the Appendix, we include an example of the POST method implemented.

Type	URI	HTTP Method	Response
Creates a new QR code ID	/qrCodes/	POST	201 (Success)
Retrieves Medical Data of an ID	/qrCodes/{id}	GET	200 (OK)
Updates Medical Data of an ID	/qrCodes/{id}	POST	201 (Success)

Table 1. RESTful API Methods

As shown in Table 1, we utilize the POST and GET methods. The POST method is used in order to create a new QR code ID with initially null medical data fields which is added as a resource to the collection when a user creates an account. This done so that every user account is associated with a medical information profile and QR code.

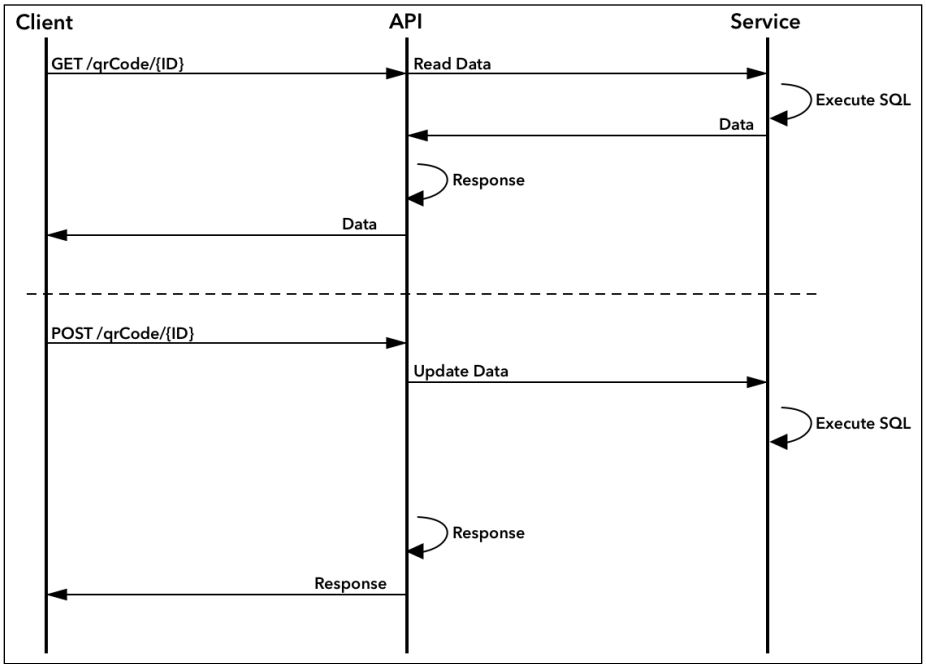

Figure 2. API Interactions

Both, GET and POST operations can be seen in Figure 2. When the user updates their medical data, we have the ability utilize the POST request in order to make updates to the user's medical information. As shown in section C of the Appendix, on line 28, we have a string query which is executed in order to make the proper updates in the database as requested by the client. string query will be updated with improved security in the future to avoid SQL injection. Additionally, we also have the ability to read data for a specific user with the use of the GET request. The API can simply be extended for further development without disrupting the entire application. The API will prove to be valuable when partnering with healthcare providers for system integration and synchronization purposes among other processes.

3.2.2 QR Code

In this section, we discuss the implementation of the QR code which includes two parts. In the first part, we review the method used for generating the QR code image with the use of Google's ZXing library. Second, we review the method used for scanning and interpreting the QR code. The QR code was first designed in Japan for the automative industry as a two-dimensional barcode and quickly spread across various industries due to its quick readability and storage capacity [6]. The amount of storage available in a QR code depends on the datatype, overall dimensions, and error correction level. The datatypes available are numeric only, alphanumeric, binary, and kanji. The overall dimensions are referred to as versions where each version includes a different type of module configuration. A minimum Version 1 configuration includes 21 x 21 modules while, the maximum is a Version 40 configuration which includes 177 x 177 modules [6]. Lastly, there are four error correction levels which are low, medium, quartile, and high which utilize the Reed-Solomon error correction algorithm [6]. In general, the error correction level and storage capacity are inversely proportionate. The error correction algorithm allows the QR code to still become readable even if the QR code is partially damaged or designed to include a marketing aspect such as a logo. In order to read a QR code, a 2-dimensional digital image sensor must detect the code and then process the data. A general QR code will include three squares

located near the corners of the image along with a fourth smaller square to focus the image in terms of size and orientation. Once stabilized and captured, the patterns are then converted into binary digits and processed through error correction to be interpreted.

Figure 3. View QR Code Screenshot

As shown in Figure 3, a user has the ability to view their QR code associated with their medical profile within the mobile application. In order for the QR code to render as an image, we utilize the Zxing barcode library [7]. Given that we are using Java, we utilize the *BufferedImage* class to operate with the image data. As shown in section D of the Appendix, we utilize the *BitMatrix* class to represent a two-dimensional matrix of bits with x and y components. We then have an *encode* method which will render the QR code as a BitMatrix two-dimensional

array [8]. The parameters provided are the contents to be encoded, the barcode format, the width in pixels, and the height in pixels. For this implementation, we encode a unique identification key associated to a user's medical profile. Once encoded, we create an empty image and modify the colors to our preferences. We then move through each pixel and identify if the bits returned from our matrix should be filled in as a color or not based on it's value. Once we have processed through the entire matrix the QR code is ready to be display or can be saved [9]. Now that we have established how the QR code is rendered, we now discuss how the QR code is read.

Within the application, we access the iPhone's camera in order to read the QR code. Thus, it is required that the user will accept a prompted request for the application to utilize the iPhone's camera. Additionally, we will be utilizing the native built-in AVFoundation framework to read the QR code [10]. Here, we implement video capturing in order to initialize and begin a video session to detect a QR code. The line of code, *captureSession.startRunning()*, begins the video session assuming that the application has obtained the user's permission to do so. Once the video session begins, the video data output relies on the *AVCaptureMetadataOutput* object to identify a QR code to be further processed. When a QR code has been identified, the QR code is captured and decoded with the use of the *AVMetadataMachineReadableCode* object [11]. We access the decoded data with the use of the stringValue property of the *AVMetadataMachineReadableCode* object in order to retrieve the QR code ID. If

the QR code ID is found and exists in our database, the medical profile details are then queried based on the ID associated with the profile.

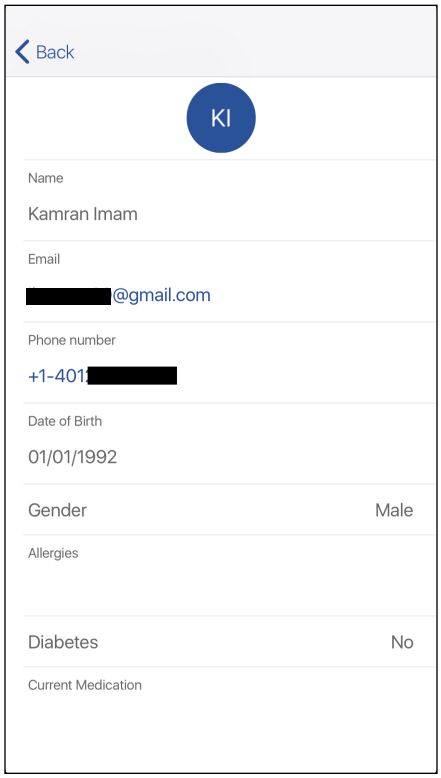

Figure 4. View Medical Information Screenshot

As shown in Figure 4, the medical information is then displayed to view. Overall, this process has become easy to implement now that Apple provides the *AVFoundation* architecture with the ability for metadata capturing. Now that we understand how our QR code is rendered and read, another aspect to briefly highlight is security.

3.2.3 Security

As with any application, security is an important topic where necessary measures should be taken to ensure that an application is secure. Briefly, we

explain how security is being implemented in this application. One aspect of security which we have implemented is the use of tokens for authentication and authorization. Users are authenticated with the use of Twilio's Authy API, a third-party API [12]. When a user logs into their account or creates an account for the first time, we ask the user for their mobile number being used in order to send the user a token via SMS (Short Message Service). This token generated by Twilio is a time-based, one-time password which must be entered within two minutes of receiving the token in order to be authenticated. Additionally, we have developed a token generated in Java as a random UUID (Universally Unique Identifier) which is a 128-bit long value guaranteeing uniqueness [13]. The UUID is generated and acts as a token for authentication and authorization purposes when a GET or POST request is made by the client. A benefit of utilizing tokens is that we can always revoke a token from a user, if needed, in order to deny future requests [14]. One area of concern to highlight is the use of SQL string queries as displayed in Appendix C which can be vulnerable to SQL injection attacks. We understand the high risks associated with passing a SQL statement as a string thus, we will revisit our implementation to utilize a more robust process of handling SQL statements. Two considerable options include the use of implementing Hibernate Object/ Relational Mapping or the implementation of prepared statements to avoid SQL injection vulnerabilities. Overall, with the use of third-party tools and universally unique identifiers, tokens can be one appropriate method to ensure that an application is secure.

3.3 Alpha Testing

This section identifies the tests which were performed in order to ensure that our implementation successfully operates as intended to do so. We begin by developing a series of questions about the implementation and discuss the actions to be performed on our program in order for us to answer these questions. Ultimately, we list a set of test cases in the format of a table, perform the test and mark if the test case passes or not.

Question 1: Does the user have the ability to create an account or login to their existing account?

Perform create account/login action

From an Apple iPhone, we will download the application via TestFlight and, as a new user, create an account. We can then check our MySQL database to view if our user table has successfully inserted a new item. We then logout of the application and try signing into the application as an existing user. We can verify this works by seeing if the application as successfully authenticated the existing user.

Question 2: Does the user have the ability to input and save their general medical information?

Input and Save information

From the application, we will navigate to the user profile and enter in dummy medical data and save the newly entered information. We will then be able

to verify if the information was correctly saved when scanning the QR code or re-editing the profile to view the information.

Question 3: Does the user have the ability to scan the QR code and view information displayed?

Scan QR code

From the application, we will navigate to the camera view and point the iPhone's camera to their QR code in order to test whether the information is accurately displayed once scanned.

Question 4: Does the user have the ability to assign a caregiver to their information?

Assign a Caregiver

From the application, we will navigate to the user profile and assign a caregiver by scanning the caregiver's QR code. We will then be able to save the information and see whether the caregiver is listed when scanning the user's QR code or by re-editing the profile information.

Question 5: Does the user's caregiver have the ability to view that they are a caregiver of the user?

Check Caregiver's View

From the Caregiver's account, we will navigate to the caregiver view page and see whether the user who is listed as a caregiver for has access to their care recipient's information.

Question 6: Does the user have the ability to message other users?

Message a User

Using two separate iPhones, we will download the application on each iPhone and create two separate accounts. We will then send messages to between the two iPhones to ensure that the messaging functionality is properly working.

The questions specified above were formed to test the core functionality of the application. In addition to these core features, detailed test cases were performed in order to cover testing of the entire application. Additionally, a beta test plan is discussed in Chapter 5 which involves a future plan to partner with a healthcare provider and pilot test the application with a group of users.

CHAPTER 4

RESULTS

In this section we begin by discussing the alpha testing results of the implementation. Additionally, we answer the questions previously presented in Chapter 3.3. The alpha testing was performed via Apple's TestFlight Application. The purpose of testing with TestFlight was to be able upload onto iTunes Connect and test the application on two Apple iPhones. TestFlight also provides us with the ability to add more testers in the future and oversee crash analytics. Overall, alpha testing is focused on finding bugs and ensuring that the implementation works as intended to do so.

4.1 Alpha Testing Results

Question 1: Does the user have the ability to create an account or login to their existing account?

Test	Expected	Actual	Status
Create Account	User initially creates an account by providing mobile number for authentication.	User is requested to enter mobile number, enters authentication passcode and account is created.	Passed
Login to Account	User logs in to existing account via passcode authentication.	User is able to login to account after successful authentication.	Passed
False Passcode	User inputs incorrect passcode and is denied logging in.	User inputs incorrect password and is denied logging in.	Passed

Table 2. User Account Test Case Results

After multiple tests, as shown in Table 2, we find that a user can successfully create an account as well as login to their existing account afterwards. We determine this test as a success based on the physical testing of application where, after creating an account or logging in, the user was then accurately navigated to the main messages screen. Additionally, we tested a scenario where the user provides a false passcode and is accurately denied login.

Question 2: Does the user have the ability to input and save their general medical information?

Test	Expected	Actual	Status
Input Medical Information	User edits their medical profile within the application.	User edits medical profile within the application and information is saved.	Passed

Table 3. Editing Medical Profile Test Case Results

A user can navigate to their profile settings and update their medical profile at anytime. To test this, we navigated to the user's medical profile and edited the information with test data. Some fields include filling out a list of medications being taken and what allergies the user has. The medical profile was then saved and we had the ability to verify that the information successfully saved by re-editing the profile, scanning the QR code and viewing the updated information within the database. As shown in Table 4, after checking via all three methods, we confirm that this test passes.

Question 3: Does the user have the ability to scan the QR code and view information displayed?

Test	Expected	Actual	Status
Scan QR code within app using camera	User navigates to camera within app and scans QR code. Medical information is displayed when scanned.	User navigates to camera within app and views medical information after pointing camera to QR and scanning.	Passed

Table 4. Scanning QR Code Test Case Results

As shown in Table 4, our next test utilized the iPhone's camera to ensure that our QR code properly works when scanned within the application. We first viewed the QR code within the app and took a screenshot to print out. Once printed out, we navigated to the application's camera icon activate the camera and scan the QR code. As soon as we pointed to the QR code, the medical information was successfully brought up for us to view.

Question 4: Does the user have the ability to assign a caregiver to their information?

Test	Expected	Actual	Status
Assign Caregiver to Medical Profile	User edits medical profile and assigns a caregiver.	User edits medical profile and assigns a caregiver.	Passed

Table 5. Assigning a Caregiver Test Case Results

One of the fields within a user's medical profile is the ability to assign a caregiver to their medical profile. In order to do so, the user must scan their caregiver's own medical QR code. Once scanned, the user will have successfully assigned the caregiver's contact information to their medical profile. In order to test

this, we utilized a second iPhone and downloaded the application. Once setup, we designated one iPhone as the care recipient and the other iPhone as the caregiver. We then went into the care recipient's medical profile and scanned the caregiver's QR code to assign them as a caregiver. As shown in Table 5, we were successfully able to do so as intended and we also verified this test by scanning the QR code and checking our database table.

Question 5: Does the user's caregiver have the ability to view that they are a caregiver of the user?

Test	Expected	Actual	Status
Caregiver View	Caregiver logs into their own account, navigates to Caregiver's View and sees the care recipient which they are a caregiver for.	Caregiver logs into their own account, navigates to Caregiver's View and sees the care recipient which they are a caregiver for.	Passed

Table 6. Caregiver View Test Case Results

In order to ensure that a caregiver understands that they are assigned as a caregiver for a user, the application includes a view which users can see who they are listed as a caregiver for. In this view, the user also has the ability to edit the care recipient's medical information given that they are designated as a caregiver. In order to verify this test, we simply viewed the Caregiver View within the caregiver's account to ensure the view displayed who they are a caregiver for. As

shown in Table 6, this test was successful and we also verified this test by viewing our database table.

Question 6: Does the user have the ability to message other users?

Test	Expected	Actual	Status
Messaging within App	User messages another user within the app and both are able to receive/ send messages to each other.	User messages another user within the app and both are able to receive/ send messages to each other.	Passed

Table 7. In-App Messaging Test Case Results

The application also provides users with the ability to message each other by adding other user's phone number registered with the account to their contacts within the application. Although the messaging portion of this application already existed, it was important to ensure that this feature was still properly working. In order to test this, we continued to utilize the two iPhones we had previously setup and messaged each other from the two iPhones. As shown in Table 7, we were able to successfully send and receive messages using both iPhones.

Overall, we were able to successfully alpha test the implementation to ensure that all aspects of the application operates properly as intended to do so. The testing mentioned above were intended to test the core functionalities of the implementation while, we also properly tested the application in its entirety as well. Given that this was an alpha test were the test was limited to ensuring the application was defect-free, a beta test is still required. In Chapter 5, we discuss the beta test plan itself along with additional future work which lies ahead.

CHAPTER 5

DISCUSSION

In Chapter 1, we had defined clear goals of the implementation. Additionally, we had mentioned the motivation behind this implementation and related works. In Chapter 4, we discussed the results of our testing by answering questions which we previously mentioned derived by our goals. Here, we will now elaborate on our results and verify that we have successfully met each of our goals.

Our first two goals were motivated by two specific reasons. One was to ensure that application works as intended to do so by performing an alpha test. Secondly, was to establish a beta test plan to ensure that the application can be validated as a proof of concept. As mentioned in its entirety, Chapter 4 validates that we have successfully performed alpha testing and passed each of the test cases to ensure that our implementation is working properly. In the next section, we continue on to discuss the beta test plan which fulfills our second goal.

Our third goal was to publish a landing page in order to have a web presence of the implementation which has been developed and to get the application properly displayed as a professional start-up company. We satisfy this goal by purchasing the domain name, atlashealth.io (subject to expire in 2018), and publish a landing page with the necessary information needed to establish a web presence as a start-up company. Our remaining goals consist of ensuring that the

implementation includes the necessary functionalities in order to improve the coordination of care.

A user can successfully create an account when initially accessing the application and also has the ability to login afterwards which fulfills our fourth goal. Goals five and six focused on the core functionality of inputting or editing medical information and utilizing the QR code to view the medical information. We have successfully verified that a user can navigate to their medical profile and edit their information as needed. A user can then share their QR code image and, when scanned, will display the user's saved medical information thereby achieving goals five and six. Lastly, we confirm that users can also assign a caregiver to their medical profile. In doing so, a caregiver can login to their account and has the ability to view who they are a caregiver for and edit the care recipient's medical information as well. Thus, we achieve our remaining goals.

Overall, from the testing conducted in Chapter 4, we successfully meet each of our goals listed in Chapter 1. The implementation provides users with the ability to save their general medical information within the application. Users then have the ability to carry and share their QR code with other healthcare providers to easily scan and view their general medical information. Additionally, the medical information includes the user's caregiver's contact information in order to effectively communicate with the caregiver as needed. Thus, the outcome of this implementation is successful. In the last chapter, we discuss the beta test plan which we have formed to verify a proof of concept by testing with a healthcare

provider. Finally, we continue on to discuss the remaining and ongoing future work of this implementation.

5.1 Beta Test Plan

This section details a plan for beta testing the application in the future. Beta testing will require partnering with an existing healthcare provider and a group of volunteer patients. Overall, beta testing is focused on evaluating the user's satisfaction and ensuring that the implementation is ready for public release. We would also like to confirm that we have a product market fit based on the application we have developed. Additionally, we need to monitor and hear of any bugs which users find in order to fix the errors while also learning more about what users like and dislike. Lastly, we would like to collect metrics in order for us to understand the impact delivered with our application. The healthcare provider which we work with may also have their own standardized test plan in place which would then be used as an alternative.

The beta test plan will be a traditional approach by having our target market test the pre-released product and allow us to gather data for future improvements. Based on our implementation, our target market for testing are patients with Alzheimer's Disease who are also receiving assistance from a caregiver. The purpose for this market is to be able to test the implementation with a caregiver and care recipient which this application may benefit most. The number of participants may vary based on the healthcare provider and their policies however, we aim our

first cycle of beta testing to be with 10-15 patients with an iOS mobile device and will be able to scale up in the next round of beta testing. In order to recruit these patients and their caregivers, we will need to extend our application to include additional features.

We will be generating a set a number of QR codes and a guide which will be printed and supplied to the participating healthcare provider. When a patient agrees to participate in testing the application, the healthcare provider will provide the patient with the printed QR code and guide while also collecting the patient's email address to be added as a tester via Apple's TestFlight App. Once the patient has downloaded the mobile application, they will then create their account and scan the QR code provided within the app. When scanned, the user will then be prompted to fill out their general medical profile and will be ready to use the application. The purpose for supplying the healthcare provider with a set of pre-generated QR codes is to have control over the number of accounts allocated for testing and for tracking purposes. The pre-generated QR codes have unique identifiers associated with them which will allow us to track and monitor the usage of each account associated with a specific QR code. Given our current implementation of a RESTful API, we can integrate this enhancement with the creation of a new endpoint which will be able to generate a set amount of QR codes. Additionally, we will need to implement functionality to handle registering the pre-generated QR codes to accounts when initially scanned. In the future, this

enhancement may also be beneficial for us in producing pre-generated QR codes printed on patches, bracelets or badges as well.

There are a series of questions we will try to answer as part of beta testing. The following questions will help us identify areas of the application which need improvement along with areas of the application's usage.

Question 1: How often are users updating their medical profile?

Question 2: What specific information is being updated most within a medical profile?

Question 3: How often are QR codes being scanned?

Question 4: What information is being utilized after scanning a QR code?

Question 5: What specific tasks are being performed where a user's medical profile is needed to be seen?

Question 6: On what medium is the QR code being displayed on when scanned (smartphone, t-shirt, card, etc.)?

Question 7: How often is a caregiver of a patient contacted from viewing the medical profile?

Question 8: What information was being communicated to the primary care physician, patient and/or caregiver from the application?

Question 9: Where any potential vulnerability or security risks found?

The questions listed above may require additional implementations to our application in order to be able to collect the type of data needed to answer the questions. Additionally, we will hold weekly meetings to discuss feedback from the

participating healthcare provider. We will also be reaching out to patients and their caregivers to learn more about their experience with the application. Our current implementation also includes a link to directly email us to report any bugs and feedback. Overall, the beta test plan will be crucial in evaluating the application's impact to improve operational efficiency, patient/caregiver satisfaction, and patient care outcomes.

5.2 Future Work

Now that we have successfully achieved the goals of this project, this application will continue to be improved and expanded based on the outcome of our beta testing. The immediate development in progress includes the necessary implementations to move forward towards beta testing. Minor changes will be included in the current development in progress such as an improved version of the user interface design work and resolution of all known security vulnerabilities which currently exist. Once completed, the application will go through an additional rigorous code review and alpha testing prior to beta testing. Currently, a pitch deck is also being developed for future participation in accelerator programs in order to quickly iterate between test versions. The start-up company's landing page will also be improved along with the Privacy Policy and Terms of Service to be regularly re-visited as needed. Given that the application is currently only available for use on Apple iPhones, the application will also need to be developed on the Android platform either prior to beta testing or public release.

I have applied to the MassChallenge Rhode Island Program which accelerates the most-promising startups across the industry at zero cost and for zero equity. Over the course of four months, this accelerator program additionally offers world-class mentorship, free office space, a community of top corporate partners and shares of up to $50,000 in cash prizes [15]. Although this program is highly competitive, I have submitted my application with the hopes to be selected and receive the support needed to continue this project as a start-up company. Additionally, I have reached out to Boston Children's Hospital Innovation and Digital Health Accelerator and have applied to their Accelerator Grant Program which accepts applications on a rolling basis. This program providers funding up to $25,000 with the potential for follow-on funding, access to expertise in developing digital health applications and tools, software development resources and a dedicated team supporting project management, strategy, pilot design, product development and more [16]. If accepted into the Accelerator Grant Program, I will then have the ability to pilot test the application and continue to develop iterative versions of the application until it is perfected to meet its purpose.

Lastly, I have gained an extensive amount of knowledge having participated in the MIT Global Entrepreneurship Bootcamp. I believe that my experience from this bootcamp will allow me to develop a formal pitch deck to raise funding with investors for additional support and enhancements. Here, it will be important to develop a viable business model which will prove to be one of the most challenging aspects for the success of this start-up company. Overall, I see a positive outlook in

receiving the support required from healthcare providers in order to continue this project in the form of a start-up company. The implementation, testing and results of this project serves as a successful base to continue on in the future.

LIST OF REFERENCES

[1] Duncan, D. S. (2014, August 25). [Online]. Driving Front Line Innovation in Health Care. Retrieved from: https://hbr.org/2013/04/driving-front-line-innovation

[2] Camicia, M., Chamberlain, B., Finnie, R. R., Nalle, M., Lindeke, L. L., Lorenz, L., . . . Mcmenamin, P. (2013). The value of nursing care coordination: A white paper of the American Nurses Association. *Nursing Outlook, 61*(6), 490-501. doi:10.1016/j.outlook.2013.10.006

[3] Caregiving. (n.d.). [Online]. Retrieved 2017, October 25, from: https://www.caregiver.org/caregiving

[4] Perez, S. (2014, June 02). [Online]. Prime Takes On The Challenging Task Of Bringing Your Health Records To Mobile. Retrieved from: https://beta.techcrunch.com/2014/06/02/prime-takes-on-the-challenging-task-of-bringing-your-health-records-to-mobile/

[5] Foundation, A. T. (2011, January 05). [Online]. ICE Medical Standard on the App Store. Retrieved March 11, 2018, from https://itunes.apple.com/us/app/ice-medical-standard/id412786820

[6] QR code. (2018, March 10). [Online]. Retrieved October 20, 2017, from: https://en.wikipedia.org/wiki/QR_code

[7] Zxing 3.3.2 API. (n.d.). [Online]. Retrieved 2017, December 20, from: https://zxing.github.io/zxing/apidocs/com/google/zxing/common/BitMatrix.html

[8] Ling, X. (2016, September 11). [Online]. How to Write and Read QR Code with ZXing in Java. Retrieved from: http://www.codepool.biz/zxing-write-read-qrcode.html

[9] Shah, A., About App Shah I'm an Engineer by profession, Says, H., & Says, A. S. (2017, July 17). [Online]. Java: Simple QR Code Generator Example - Now you Could have Narrow Border - Crunchify. Retrieved from: https://crunchify.com/java-simple-qr-code-generator-example/

[10] Cameras and Media Capture. (n.d.). [Online]. Retrieved 2017, December 20, from: https://developer.apple.com/documentation/avfoundation/cameras_and_media_capture

[11] Ng, A. (n.d.). [Online]. Building a Barcode and QR Code Reader in Swift 4 and Xcode 9. Retrieved 2017, December 20, from: https://www.appcoda.com/barcode-reader-swift/

[12] Twilio Authy. (n.d.). [Online]. Retrieved 2017, December 20, from: https://www.twilio.com/docs/api/authy

[13] Java Platform, Standard Edition 7 API Specification. (2017, December 19). [Online]. Retrieved from: https://docs.oracle.com/javase/7/docs/api/java/util/UUID.html

[14] Authentication Strategies. (n.d.). [Online]. Retrieved 2017, December 12, from: https://teamtreehouse.com/library/authentication-strategies

[15] MassChallenge Rhode Island. (2018, January 30). [Online]. Retrieved from: http://boston.masschallenge.org/rhodeisland

[16] Accelerator Grant Program. (n.d.). [Online]. Retrieved 2017, December 2, from: http://www.childrenshospital.org/accelerator/have-an-idea/accelerator

APPENDIX

A. Model of Medical Profile

```
1     public class Profile {
2
3         private String stringCaregiverName;
4             private String stringCaregiverEmail;
5             private String stringCaregiverMobileNo;
6
7             private String stringImageURL;
8             private String stringName;
9             private String stringEmail;
10             private String stringMobileNo;
11             private String stringBirthDate;
12             private Integer integerGender;
13             private Integer integerDiabetes;
14             private String stringAdditionalInfo;
15             private String stringMedication;
16             private String stringAllergy;
17
18             private String stringPhysicianName;
19             private String stringPhysicianMobileNo;
20             private String stringPhysicianEmail;
21
22             public String getStringCaregiverName() {
23                 return stringCaregiverName;
24             }
25
26             public void setStringCaregiverName(String stringCaregiverName) {
27                 this.stringCaregiverName = stringCaregiverName;
28             }
29
30             public String getStringCaregiverEmail() {
31                 return stringCaregiverEmail;
32             }
33
34             public void setStringCaregiverEmail(String stringCaregiverEmail) {
35                 this.stringCaregiverEmail = stringCaregiverEmail;
36             }
37
38             public String getStringCaregiverMobileNo() {
39                 return stringCaregiverMobileNo;
40             }
41
42             public void setStringCaregiverMobileNo(String stringCaregiverMobileNo) {
43                 this.stringCaregiverMobileNo = stringCaregiverMobileNo;
44             }
45
46             public String getStringName() {
47                 return stringName;
48             }
49
50             public void setStringName(String stringName) {
51                 this.stringName = stringName;
52             }
53
54
```

```
55              public String getStringImageURL() {
56                  return stringImageURL;
57              }
58
59              public void setStringImageURL(String stringImageURL) {
60                  this.stringImageURL = stringImageURL;
61              }
62
63              public String getStringEmail() {
64                  return stringEmail;
65              }
66
67              public void setStringEmail(String stringEmail) {
68                  this.stringEmail = stringEmail;
69              }
70
71              public String getStringMobileNo() {
72                  return stringMobileNo;
73              }
74
75              public void setStringMobileNo(String stringMobileNo) {
76                  this.stringMobileNo = stringMobileNo;
77              }
78
79              public String getStringBirthDate() {
80                  return stringBirthDate;
81              }
82
83              public void setStringBirthDate(String stringBirthDate) {
84                  this.stringBirthDate = stringBirthDate;
85              }
86
87              public Integer getIntegerGender() {
88                  return integerGender;
89              }
90
91              public void setIntegerGender(Integer integerGender) {
92                  this.integerGender = integerGender;
93              }
94
95              public Integer getIntegerDiabetes() {
96                  return integerDiabetes;
97              }
98
99              public void setIntegerDiabetes(Integer integerDiabetes) {
100                  this.integerDiabetes = integerDiabetes;
101              }
102
103              public String getStringAdditionalInfo() {
104                  return stringAdditionalInfo;
105              }
106
107              public void setStringAdditionalInfo(String stringAdditionalInfo) {
108                  this.stringAdditionalInfo = stringAdditionalInfo;
```

```
109             }
110
111             public String getStringMedication() {
112                 return stringMedication;
113             }
114
115             public void setStringMedication(String stringMedication) {
116                 this.stringMedication = stringMedication;
117             }
118
119             public String getStringAllergy() {
120                 return stringAllergy;
121             }
122
123             public void setStringAllergy(String stringAllergy) {
124                 this.stringAllergy = stringAllergy;
125             }
126
127
128             public String getStringPhysicianName() {
129                 return stringPhysicianName;
130             }
131
132             public void setStringPhysicianName(String stringPhysicianName) {
133                 this.stringPhysicianName = stringPhysicianName;
134             }
135
136             public String getStringPhysicianMobileNo() {
137                 return stringPhysicianMobileNo;
138             }
139
140             public void setStringPhysicianMobileNo(String stringPhysicianMobileNo) {
141                 this.stringPhysicianMobileNo = stringPhysicianMobileNo;
142             }
143
144             public String getStringPhysicianEmail() {
145                 return stringPhysicianEmail;
146             }
147
148             public void setStringPhysicianEmail(String stringPhysicianEmail) {
149                 this.stringPhysicianEmail = stringPhysicianEmail;
150     }
151     }
```

B. POST API Method

```
1     @POST
2         @Path("MedicalQRCodes/{qrCodeID}")
3         @Produces(MediaType.APPLICATION_JSON)
4         @Consumes(MediaType.APPLICATION_JSON)
5         public HashMap<String, Object> updateMedicalQRCode(@PathParam("qrCodeID")
6         String stringQRCodeID, HashMap<String, String> hashMapRequest) {
7             HashMap<String, Object> hashMapResponse = validateQRCode(hashMapRequest);
8             if (hashMapResponse != null) {
9                 return hashMapResponse;
10             }
11             String stringQRCodeName = hashMapRequest.get("qrCodeName");
12             String stringQRMessage = hashMapRequest.get("qrCodeMessage");
13             hashMapResponse = validateCaregiver(hashMapRequest);
14             if (hashMapResponse != null){
15                 return hashMapResponse;
16             }
17
18             QRCode qrCode = QRCodeService.getQRCodeService()
19              .updateQRCode(stringQRCodeID,
20              stringQRCodeName, stringQRMessage);
21             Profile profile =
22              QRCodeService.getQRCodeService()
23              .updateMedicalQRCode(stringQRCodeID,hashMapRequest);
24             qrCode.setMedicalDetails(profile);
25             hashMapResponse = new HashMap<>();
26             if (profile == null) {
27          return getErrorHashMapForMessage
28      ("Unable to update Medical QRCode. Please try again.");
29             }
30             hashMapResponse.put("qrCode", qrCode);
31             hashMapResponse.put(APIResponseKeys.ISERROR, false);
32             hashMapResponse.put(APIResponseKeys.MESSAGE, "Success.");
33         return hashMapResponse;
34     }
```

C. Update Service Request

```
1     public Profile updateMedicalQRCode(String stringQRCodeID,
2     HashMap<String,String>hashMapRequest){
3         DateFormat formatter = new SimpleDateFormat("yyyy-MM-dd");
4             Date myDate = null;
5                 try {
6                     myDate = formatter.parse(hashMapRequest.get("birthDate"));
7                     } catch (ParseException e) {
8                         e.printStackTrace();
9                 }
10                     java.sql.Date sqlDate = new java.sql.Date(myDate.getTime());
11
12                     String careRecipientMobileNO = "";
13                     if (!(hashMapRequest.get("countryCode").equalsIgnoreCase("") ||
14    hashMapRequest.get("mobileNo").equalsIgnoreCase(""))){
15                         careRecipientMobileNO = hashMapRequest.get("countryCode")+"-
16    "+hashMapRequest.get("mobileNo");
17                 }
18                    String caregiverMobileNO = "";
19                  if (!(hashMapRequest.get("familyCountryCode").equalsIgnoreCase("") ||
20    hashMapRequest.get("familyMobileNo").equalsIgnoreCase(""))){
21                         caregiverMobileNO = hashMapRequest.get("familyCountryCode")+"-
22    "+hashMapRequest.get("familyMobileNo");
23                 }
24
25     String physicianMobileNO = hashMapRequest.get("physicianCountryCode")+"-
26    "+hashMapRequest.get("physicianMobileNo");
27
28    String stringQuery = "UPDATE DETAILS SET care_recipient_imageurl=
29    '"+hashMapRequest.get("imageURL")+"',care_recipient_name='"+
30    hashMapRequest.get("userName")+"',care_recipient_email='"+
31    hashMapRequest.get("email")+"',care_recipient_mobileno='"+
32    careRecipientMobileNO+"',care_recipient_birth_date='"+sqlDate+"',
33    care_recipient_gender='"+hashMapRequest.get("gender")+"'," +
34    "care_recipient_additional_information='"+hashMapRequest.get("address")+"',
35    care_recipient_allergy='"+hashMapRequest.get("allergy")+"',
36    care_recipient_diabetes='"+hashMapRequest.get("diabetes")+"',
37    care_recipient_medication='"+hashMapRequest.get("medication")+"'" +
38    ",caregiver_name='"+hashMapRequest.get("familyName")+"',
39    caregiver_mobileno='"+caregiverMobileNO+"',
40    caregiver_email='"+hashMapRequest.get("familyEmail")+"',
41    physician_name='"+hashMapRequest.get("physicianName")+"',
42    physician_email='"+hashMapRequest.get("physicianEmail")+"',
43    physician_mobileno='"+physicianMobileNO+"'" +
44                                " where QRCodeID='"+stringQRCodeID+"'";
45
46                if (sharedDAO.updateData(stringQuery)){
47                    return getCaregiver(stringQRCodeID);
48                    }
49                        return null;
50                    }
```

D. Render QR Code Image

```
1     public static void writeQRCode() {
2     QRCodeWriter qrwriter = new QRCodeWriter();
3     BitMatrix matrix = null;
4     int QRCODE_IMAGE_HEIGHT_WIDTH = 512;
5
6         try {
7             matrix = qrwriter.encode(qrCode.getStringQRCodeID(), BarcodeFormat.QR_CODE,
8     QRCODE_IMAGE_HEIGHT_WIDTH, QRCODE_IMAGE_HEIGHT_WIDTH);
9             } catch (WriterException e) {
10                e.printStackTrace();
11        }
12
13     BufferedImage image = new BufferedImage(QRCODE_IMAGE_HEIGHT_WIDTH,
14     QRCODE_IMAGE_HEIGHT_WIDTH, BufferedImage.TYPE_INT_ARGB);
15         image.createGraphics();
16         Graphics2D graphics = (Graphics2D) image.getGraphics();
17         Color mainColor = new Color(14, 41, 76);
18         graphics.setColor(mainColor);
19
20         for (int k = 0; k < QRCODE_IMAGE_HEIGHT_WIDTH; k++) {
21             for (int j = 0; j < QRCODE_IMAGE_HEIGHT_WIDTH; j++) {
22                 if (matrix.get(k, j)) {
23                     graphics.fillRect(k, j, 1, 1);
24                 }
25             }
26         }
27     graphics.dispose();
28         try {
29             ImageIO.write(image, "jpg", new File("MedicalQRCode.jpg"));
30             } catch (IOException e) {
31                 e.printStackTrace();
32         }
33     }
```

BIBLIOGRAPHY

Accelerator Grant Program. (n.d.). [Online]. Retrieved 2017, December 2, from: http://www.childrenshospital.org/accelerator/have-an-idea/accelerator

Authentication Strategies. (n.d.). [Online]. Retrieved 2017, December 12, from: https://teamtreehouse.com/library/authentication-strategies

Cameras and Media Capture. (n.d.). [Online]. Retrieved 2017, December 20, from: https://developer.apple.com/documentation/avfoundation/cameras_and_media_capture

Camicia, M., Chamberlain, B., Finnie, R. R., Nalle, M., Lindeke, L. L., Lorenz, L., . . . Mcmenamin, P. (2013). The value of nursing care coordination: A white paper of the American Nurses Association. *Nursing Outlook, 61*(6), 490-501. doi:10.1016/j.outlook.2013.10.006

Caregiving. (n.d.). [Online]. Retrieved 2017, October 25, from: https://www.caregiver.org/caregiving

Duncan, D. S. (2014, August 25). [Online]. Driving Front Line Innovation in Health Care. Retrieved from: https://hbr.org/2013/04/driving-front-line-innovation

Foundation, A. T. (2011, January 05). [Online]. ICE Medical Standard on the App Store. Retrieved from: https://itunes.apple.com/us/app/ice-medical-standard/id412786820

Java Platform, Standard Edition 7 API Specification. (2017, December 19). [Online]. Retrieved from: https://docs.oracle.com/javase/7/docs/api/java/util/UUID.html

Ling, X. (2016, September 11). [Online]. How to Write and Read QR Code with ZXing in Java. Retrieved from: http://www.codepool.biz/zxing-write-read-qrcode.html

MassChallenge Rhode Island. (2018, January 30). [Online]. Retrieved from: http://boston.masschallenge.org/rhodeisland

Ng, A. (n.d.). [Online]. Building a Barcode and QR Code Reader in Swift 4 and Xcode 9. Retrieved 2017, December 20, from: https://www.appcoda.com/barcode-reader-swift/

Perez, S. (2014, June 02). [Online]. Prime Takes On The Challenging Task Of Bringing Your Health Records To Mobile. Retrieved from: https://beta.techcrunch.com/2014/06/02/prime-takes-on-the-challenging-task-of-bringing-your-health-records-to-mobile/

QR code. (2018, March 10). [Online]. Retrieved October 20, 2017, from: https://en.wikipedia.org/wiki/QR_code

Shah, A., About App ShahI'm an Engineer by profession, Says, H., & Says, A. S. (2017, July 17). [Online]. Java: Simple QR Code Generator Example - Now you Could have Narrow Border - Crunchify. Retrieved from: https://crunchify.com/java-simple-qr-code-generator-example/

Twilio Authy. (n.d.). [Online]. Retrieved 2017, December 20, from: https://www.twilio.com/docs/api/authy

Zxing 3.3.2 API. (n.d.). [Online]. Retrieved 2017, December 20, from: https://zxing.github.io/zxing/apidocs/com/google/zxing/common/BitMatrix.html