

A FLEXIBLE SOFTWARE PLATFORM FOR DISEASE MANAGEMENT

by

Mohammad Ali Ghaderi

APPROVED BY SUPERVISORY COMMITTEE:

Gopal Gupta, Chair

Lakshman S. Tamil, Co-Chair

Vincent Ng

Lawrence Chung

Copyright 2016

Mohammad Ali Ghaderi

All Rights Reserved

Dedicated to love...

A FLEXIBLE SOFTWARE PLATFORM FOR DISEASE MANAGEMENT

by

MOHAMMAD ALI GHADERI, MS

DISSERTATION

Presented to the Faculty of
The University of Texas at Dallas
in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY IN
SOFTWARE ENGINEERING

THE UNIVERSITY OF TEXAS AT DALLAS

December 2016

ACKNOWLEDGMENTS

It is my pleasure to thank those who made this dissertation possible. At first, I thank my parents for being such a great support in my entire life. I offer my regards to Professor Gopal Gupta who supported me generously from the admission process to the graduation. I am heartily thankful to Professor Lakshman Tamil for being such a compassionate support in these years. I also would like to make a special reference to Professor Mehrdad Nourani who was a great honest help. Furthermore, I am grateful to Professor Lawrence Chung who gave me a better view of my field. Lastly, I thank all of my friends who supported me in any respect during the long journey of my studies, and I wish I could write their names here.

December 2016

A FLEXIBLE SOFTWARE PLATFORM FOR DISEASE MANAGEMENT

Publication No. _____

Mohammad Ali Ghaderi, PhD
The University of Texas at Dallas, 2016

Supervising Professors: Prof. Gopal Gupta, Chair
Prof. Lakshman S. Tamil, Co-Chair

Chronic diseases afflict patients for many years. One way to decrease the burden of chronic disease care on the healthcare system is to have patients self-manage their disease. The ubiquity of smartphones presents a great opportunity for making self-management easier. This is an active research area, but the lack of appropriate software solutions compels many researchers to design and develop their systems from scratch. In this dissertation, we propose a novel and flexible software platform for the development of disease management research applications. Our platform is novel in the sense that a comprehensive platform similar to ours does not exist. We have considered a broad list of functional and non-functional features to support the most common research scenarios. To demonstrate the functionality of this platform, we have developed two representative applications—one for breast cancer risk assessment research and another one for post-surgical care research. Our initial studies show that our platform greatly facilitates the

development process of such applications. The platform helps in considerably reducing the required development time by providing many ready-to-reuse features and ensuring high quality in the architectural design. The infrastructure developed is quite comprehensive and consists of more than a hundred thousand lines of code.

Our research makes the following contributions: an advanced architectural design for developing research applications, a software platform with many built-in reusable components for medical research and consideration of non-functional requirements such as portability, reliability, maintainability to name a few, and an augmented reality module for breast tumor visualization.

TABLE OF CONTENTS

Acknowledgments.....	v
Abstract	vi
List of Figures	xii
List of Tables	xvi
 CHAPTER 1 INTRODUCTION	 1
1.1 Motivation.....	1
1.2 Problem Statement	2
1.3 Main Contributions	5
1.4 Document Structure	6
 CHAPTER 2 BACKGROUND	 7
2.1 Disease Management Apps.....	7
2.2 Software Platforms.....	8
2.3 Breast Cancer	14
2.4 Augmented Reality for Disease Management	16
2.5 Chapter Summary	17
 CHAPTER 3 ESSENTIAL FEATURES AND ARCHITECTURE.....	 18
3.1 Features List.....	18
3.1.1 Education.....	20
3.1.2 Single Vital Value Records	20
3.1.3 Time Series Data (real-time data) Recording.....	21
3.1.4 Vital Value Visualization	21
3.1.5 Questionnaire	22
3.1.6 Notification.....	22

3.1.7	Reminder	23
3.1.8	Medication records	23
3.1.9	Diet and Activity Records	23
3.1.10	Video/Voice/Text Communication	24
3.1.11	Appointment Scheduling	24
3.1.12	Medical History and Document Records	25
3.1.13	Intelligent Analyzer (Decision Support System)	25
3.1.14	Integration with other systems	26
3.1.15	Medical Devices Connectivity	26
3.1.16	Speech to Text	26
3.1.17	Online/Mobile Payment	27
3.2	Quality Features	27
3.2.1	Software Development Methodology	28
3.2.2	Security and Privacy	28
3.2.3	Extensibility	30
3.2.4	Reliability	30
3.2.5	Scalability	31
3.2.6	Maintainability	31
3.2.7	Performance	31
3.2.8	Portability	32
3.2.9	Reusability	32
3.3	Perspective	33
3.4	Architecture	34
3.4.1	Framework	36
3.4.2	Entity Layers	37
3.4.3	Server-Side API and User Interface	38
3.4.4	Client-Side Code and User Interface	39
3.4.5	Shared Library	41
3.5	Deployment Model	42
3.6	Chapter Summary	44

CHAPTER 4	THE PROPOSED PLATFORM.....	45
4.1	Base Platform.....	45
4.1.1	Authentication	48
4.1.2	Main Menu	53
4.1.3	Manage list of doctors	56
4.1.4	Define Doctor's Availability Calendar	58
4.1.5	Book an Appointment	60
4.1.6	Manage Appointments	65
4.1.7	Manage Medical History	69
4.1.8	Manage Account and Profile.....	74
4.1.9	Manage Payments	81
4.1.10	Visit (Video Communication).....	83
4.1.11	General system features	85
4.2	Augment Reality Module.....	87
4.3	Augmented Reality Process	88
4.3.1	Pre-Processing	89
4.3.2	Marker Finder.....	89
4.3.3	Position Estimator	92
4.3.4	Graphic Generation	94
4.4	Chapter Summary	94
CHAPTER 5	REPRESENTATIVE APPLICATIONS.....	95
5.1	Breast Cancer Application	95
5.1.1	Customizing the base	96
5.1.2	Education module.....	98
5.1.3	Reusing Medical History module for Risk Assessment.....	100
5.1.4	Re-using the base modules.....	101
5.2	Post-Surgical Care Application.....	102
5.2.1	Implementation.....	102
5.2.2	Pain Level Form.....	103
5.2.3	Pain Level Chart.....	106

5.2.4	Record Sleep Form.....	110
5.2.5	Physical Activity Form.....	112
5.2.6	Record Food Forms	114
5.3	Chapter Summary	117
CHAPTER 6	EXPERIMENT AND RESULTS	118
6.1	Research methodology.....	118
6.2	The proposed platform user study.....	118
6.2.1	Discussion	120
6.3	Quality Features	123
6.3.1	Security and Privacy.....	123
6.3.2	Extensibility	126
6.3.3	Reliability	126
6.3.4	Scalability.....	126
6.3.5	Maintainability	127
6.3.6	Performance	128
6.3.7	Portability	129
6.3.8	Reusability.....	130
6.4	AR Module Experiment.....	130
6.4.1	Discussion	131
6.5	Conclusion	133
6.6	Future Work	134
6.7	Chapter Summary	136
REFERENCES	137

VITA

LIST OF FIGURES

Figure 1. Apple CareKit Sample Symptom Tracker application	10
Figure 2. easyHealthApps web based code generator	13
Figure 3. The general perspective	33
Figure 4. Overall system architecture	35
Figure 5. Screenshot of the application for breast cancer risk assessment form for doctors	41
Figure 6. Deployment Model	43
Figure 7. General use case diagram for the doctor actor	46
Figure 8. General use case diagram for the patient actor	47
Figure 9. Screenshot of the login page	49
Figure 10. Screenshot of Signup page where users can register in the system	49
Figure 11. Screenshots of the "Can't Login" and "Continue Registration" pages where user will be guided to access his account when he can't login to the system	50
Figure 12. Screenshots of the platform to recover forgotten passwords	51
Figure 13. User phone verification page	53
Figure 14. Screenshot of the doctor's main menu	54
Figure 15. Screenshot of the doctor's professional info menu	54
Figure 16. Screenshot of the patients' main menu	55
Figure 17. Screenshot of My Doctors page	56
Figure 18. Screenshot of Find doctor by phone number page	57

Figure 19. Screenshot of My Calendar page (Doctor’s availability)	58
Figure 20. Screenshot of My Calendar page (Adding available time)	59
Figure 21. Screenshot of My Calendar page (Copy range form).....	59
Figure 22. Screenshot of Book an appointment page (step 1 – select a doctor)	61
Figure 23. Screenshot of Register new patient page	61
Figure 24. Screenshot of Book an appointment page (step 1 – Select a patient).....	62
Figure 25. Screenshot of Book an appointment page (step 2 – schedule)	63
Figure 26. Screenshot of Book an appointment page (step 3 – reason).....	64
Figure 27. Screenshot of Book an appointment page (step 4 – schedule)	64
Figure 28. Screenshot of Today’s appointment page.....	65
Figure 29. Screenshot of Reschedule page	66
Figure 30. Screenshot of Appointment details page (Visit Info tab)	67
Figure 31. Screenshot of Appointment details page (Doctor Report tab).....	68
Figure 32. Screenshot of Appointment details page (Doctor Review tab)	69
Figure 33. Screenshot of Patients’ Medical History page (Allergy tab).....	70
Figure 34. Screenshot of Patients’ Medical History page (Condition tab)	71
Figure 35. Screenshot of Patients’ Medical History page (Medications tab)	71
Figure 36. Screenshot of Patients’ Medical History page (Family History tab).....	72
Figure 37. Screenshot of Patients’ Medical History page (Social tab)	73
Figure 38. Screenshot of Patients’ Medical History page (Signature tab).....	73
Figure 39. Screenshot of Account Info page	74
Figure 40. Screenshot of Personal Info Edit page.....	75

Figure 41. Screenshot of Patient Insurance page	76
Figure 42. Screenshot of Profile page (Insurance tab).....	76
Figure 43. Screenshot of Profile page (Languages tab).....	77
Figure 44. Screenshot of Profile page (Documents tab).....	78
Figure 45. Screenshot of Doctor and Clinic Information page	79
Figure 46. Screenshot of Certified States page	80
Figure 47. Screenshot of Doctors' Acceptable Insurance Plans page	80
Figure 48. Screenshot of Doctors' Payment Info page	81
Figure 49. Screenshot of Visit Details page (payments tab).....	81
Figure 50. Screenshot of Doctors' Ask for Payment page.....	82
Figure 51. Screenshot of Payments list (Doctor's view)	82
Figure 52. Screenshot of Payments page (Patient's view).....	83
Figure 53. Screenshot of Calling Patient page.....	84
Figure 54. Screenshot of Received Call page	84
Figure 55. Screenshot of Video Communication page	85
Figure 56. Screenshot of Feedback page	86
Figure 57. Screenshot of Voice Recognition feature	87
Figure 58. Overall architecture of the augmented reality module	89
Figure 59. Two examples of markers used in this work. Left marker is a grid of 7x7 and the right one is 5x5. Dotted lines are the cut lines and are not parts of the markers.....	90

Figure 60. Frontal view of a marker. Left picture obtained from a camera and the right is the frontal view. Corners of a potential marker in the camera picture is mapped to the frontal view to process detection in the next step.....	92
Figure 61. The 3D model of a tumor used to be shown on the breast	94
Figure 62. Screenshot of Patient’s Menu in Breast Cancer Representative Application.....	96
Figure 63. Screenshot of BSE Training in Breast Cancer Representative Application.....	99
Figure 64. Screenshot of Risk questionnaires in Breast Cancer Representative Application	100
Figure 65. Screenshot of Patient’s Menu in Post-Surgical Care Application.....	103
Figure 66. Screenshot of Pain Level Form	105
Figure 67. Screenshot of Pain Level Chart	107
Figure 68. Screenshot of Record Sleep Form	111
Figure 69. Screenshot of Record Physical Activity Form	114
Figure 70. Screenshot of Food Group Add Form	115
Figure 71. Screenshot of Add Food Item Form	116
Figure 72. Screenshot of Calorie Chart Form.....	116
Figure 73. Experiments of the augmented reality system. (A) Screenshot (what a physician sees on the screen) tested on a bra. (B) The original plastic model without AR (C) Screenshot on a plastic model. (D) The same as B, but from a different angle.	132

LIST OF TABLES

Table 1. Essential features of a TDMS with a brief explanation	19
Table 2. Survey results for the patient users	120
Table 3. Survey results for the doctor users.....	121
Table 4. Comparison of our platform with Apple CareKit and Berkley Telemonitoring.....	122
Table 5. Basic Security checkers classes in the platform	124

CHAPTER 1

INTRODUCTION

1.1 Motivation

Chronic diseases such as cancers, diabetes, asthma, chronic obstructive pulmonary disease (COPD), chronic kidney disease (CKD) and congestive heart failure (CHF) stay with patients for years. Unfortunately, the number of patients who are diagnosed with one or more of the chronic diseases is constantly increasing in the world [1], and dealing with these diseases is one of the biggest challenges of the global public health in the 21st century [2]. As of 2010, it is estimated that 21.3% of patients in the U.S. with a chronic condition use disease management programs [3]. Yet, more than 75% of current healthcare spending in the U.S. goes to the chronic illnesses, and predictions say that the future trend is ascending [4]. Since there is no effective cure for these diseases, it needs to be managed for several years.

Chronic diseases is not the only category of the diseases without cure. Many types of the cancer are similar to them. For example, breast cancer is the most prevalent and affects more than one million individuals annually [5]. Breast cancer rate is high in both developing and developed countries; just in North America, over 1.5% of the population is affected-that is more than 3.2 million individuals [6]. Approximately more than 230,000 women are diagnosed with breast cancer annually in the United States, and more than 39,000 die every year according to the recent statistics [7].

Mobile health (mHealth) is a subcategory of Electronic health (eHealth) that is about using wireless communications, especially mobile smartphones, to practice medicine and public health. Smartphones have greatly impacted our lifestyle since their appearance. For example, self-management of chronic diseases using mobile health application (diseases such as Chronic Obstructive Pulmonary Disease COPD [8], [9], depression [10] and pain management [11]) are studied in some research works.

1.2 Problem Statement

If we think that the traditional medicine is doctor-centric, current disease management is mostly patient-centric, and we see its reflection in the mobile apps. This causes the problem of disconnection between doctors and patients, and perhaps discouragement of patients to use these apps. One of the issues is that patients don't get any valuable output from these systems.

The solution is to involve doctors in a patient-centric management. Obviously, doctors cannot be available to all their patients all the time, so they need to take actions remotely. This makes a great connection between disease management (DM) and telemedicine (TM), and traditional medicine. We call it, Tele Disease Management System (TDMS). We define TDMS as a system consisting of a software and hardware that helps patients to do self-management of their chronic diseases and keeps the doctors involved remotely in this process. Since there are many unknown areas in the disease management, the research in this area is active. A recent survey lists 133 papers related to mHealth [12].

However, many researchers start to develop their system from scratch. Thus, they need to spend weeks to months to prepare their research application. Let's explain the problem with several stories:

Story 1: Student S has recently started his Ph.D. research under the supervision of the professor X. They are working in the healthcare area. They would like to test the hypothesis that having data of the blood oxygen level along with ECG data can improve heart arrhythmia detection. Since their idea is novel, there is no publicly available dataset, and they need to create their own dataset. Therefore, they need to create their own monitoring application. This application is not trivial. It needs a lot of programming experience, and ensuring the security and privacy is not easy. Since there is no reliable platform, the student needs to spend a year without doing any research to prepare the application. He needs to prolong his Ph.D. program by one year to finish the clinical trial and analysis.

Story 2: A breast cancer research institute requires to obtain data from thousands of the women to define a new cancer risk calculator. In addition, they need to connect patients to the doctors in the institute to adjust the research questions and help high-risk patients. They are willing to do a survey to obtain patients' data and process this data to create this calculator. There is no publicly available dataset; therefore, they need to develop their own data gathering application. The development of this application is a complex process. They need to pay thousands of dollars to outsource the project. As such an application requires a lot of quality considerations, it may add a lot of time and cost to the project.

The problem is that there is no good software framework to be flexible and comprehensive enough to support the development of such applications. The researchers need to start from scratch or be satisfied with limited features available in the currently proposed frameworks. A lately published research work says: "To the best of our knowledge, there is no complete existing open-source solution for telemonitoring that provides support for client and server development in a uniform,

easy-to-use and end-to-end privacy-preserving manner [13].” The authors propose an Android software framework as a solution to this problem, but yet the server-side part of the framework still needs a lot of work. A recent survey paper also found five other software frameworks to address similar problem [12]. However, each of these solutions addresses only a part of the problem.

We believe that a complete solution should be a software framework that is flexible enough to support various research scenarios. In addition, we believe that targeting doctors or researchers who are not software developers is very limiting. Thus, the proposed framework should be designed to be utilized by software developers to support unknown scenarios that may not be available in the framework.

It is a challenging problem because the framework needs to address many similar scenarios such as recording data and storing to database while not limiting the developers’ power to define unknown scenarios. It also requires an advanced knowledge of the software architecture. However, the researchers in this area are not usually software engineers and they are not capable of defining a solid architecture or it is very time consuming to create one. Another side of the problem is that many developers or software firms may not be familiar with the challenges that come with the healthcare applications. For instance, handling the security and privacy is very hard, and accessing the data by unauthorized users can cause serious issues due to privacy laws. For example, in the United States, compatibility with Health Insurance Portability and Accountability Act (HIPAA) requires a lot of considerations in the software systems.

1.3 Main Contributions

This research has several important contributions. The main contribution of the research is a web-based software platform and a mobile application that is flexible enough to let the researchers develop their telemedicine research applications. Many features such as questionnaire, patients' vital values and visualization, video communication, appointment scheduling, medical history and notification are implemented in the base platform. The platform is customizable and it saves a lot of development time for similar applications. In addition, it is easy to understand and many libraries and base classes are provided in the framework so that a junior to mid-level developer can still develop reliable applications. The current code is more than 100,000 lines including the white lines and comments.

In this research, we also introduce a sample telemedicine application that can help patients to schedule appointments with their doctors and visit them either online or in-person. This app also keeps the medical history data of the patients and lets the doctors access their patients' medical history. This app can be used for researching patients' engagement in a telepsychiatry application. Furthermore, another representative application is also developed to be used to gather patients' data and allow researchers to analyze this data and make better breast cancer risk assessments. This app helps patients to answer questionnaires and doctors to examine that information. The doctor can also do the risk assessment analysis personally when the data is obtained from the patients. In addition, another representative application is also developed to track the food eaten or activities and record the daily calorie amount. All basic functionality of a telemedicine platform is available in these representative applications.

Moreover, as a part of our research and to show the capability of the integration with more advanced features such as image processing, we propose an Augmented Reality application that can show 3D views of the breast internal composition, especially lumps and its tumors to doctors so that they can have a better visualization. Such visualization of breast tumors is very important for doctors, oncologists, and surgeons because a better visualization can lead to a better screening, less false positive and avoiding unnecessary biopsy. It can also be helpful for planning and navigation during surgery.

1.4 Document Structure

The rest of this document is structured as follows: The second chapter covers the related research and the background knowledge. The third chapter presents the essential functionality and quality features and the platform architecture. The fourth chapter discusses the base features implemented in the base platform. Chapter five introduces two representative applications developed on the top of the platform. Chapter six provides the experiments and discussions. We conclude the dissertation in Chapter six by providing a summary and our thoughts of future works.

CHAPTER 2

BACKGROUND

2.1 Disease Management Apps

Since there is no cure for most of the chronic diseases, monitoring and managing them is the only way of dealing with them. Disease management has been around for years [14], but smartphones have created a great opportunity to make it easier and more prevalent [15]. Many mobile apps help patients with self-management of their diseases such as diabetes [16]–[18], COPD [8] and CHF [19]. Self-management of chronic diseases takes some burden off the doctors' shoulders by involving patients, and mobile application (app) stores have plenty of apps for different diseases. Smartphones greatly impacted our lifestyle since their emergence. They are changing the way people seeking for healthcare information. There is still a lack of cancer-related applications with scientifically backed data [20]. Patient-reported outcome (PRO) data can be collected more frequently by using smartphones. Some studies found reporting vital signs using SMS services for asthma patients is feasible [21]. Managing side-effects of chemotherapy and home monitoring of patients' symptoms for colon cancer using smartphones suggested similar results [22]. Asking breast cancer patients who are under chemotherapy to report their sleep disturbance-related data using their smartphones on a daily basis is also found to be feasible [23]. Explicit reminders and clinician feedback are mandatory to keep the patients engaged in voluntary self-reporting their symptoms between visits [24].

The researchers of the University of California Los Angeles introduced a framework called WANDA (Weight and Activity with Blood Pressure Monitoring System) for monitoring CHF patients [19]. WANDA has a three-tier architecture. The first tier monitors health-related measuring using sensors. The second tier consists of several web servers that maintain and store the data. The third tier is a database server. The system read the data from several devices such as weight scale, blood pressure monitor and blood glucose meter and stores it in a server-side database. The study claims that CHF patients monitored by this system are less likely to have readings fall outside a healthy range.

Rehab@Home framework is another Tele-monitoring platform that is introduced for the rehabilitation of stroke patients [25]. The platform consists of some sensors to collect the data and it is linked to a tablet. The tablet sends the data over the internet and an expert can access the processed data using a web interface. The purpose of this framework is to support the continuation of rehabilitation at home. This platform tries to address some issues related to the development of these applications such as the design of the user interfaces, data acquisition, data storage, security, and privacy. However, this effort is limited to the Android operating system.

A recent survey paper lists more than a hundred papers related to mHealth applications [12]. Since this research is a comprehensive survey until December 2015, we eliminate repeating their findings here. This research also indicates that the research in this area is active and many researchers are working hard to find better ways of using smartphones for managing diseases and telehealth.

2.2 Software Platforms

Ubiquitous and adaptive chronic disease management systems has recently gained attention. Researchers in [26] suggest that a ubiquitous healthcare system for chronic disease is needed.

There are also some efforts to develop a ubiquitous chronic disease management platform (called CHRONIOUS) for COPD and CKD [27]. However, we are still far from such systems. So, a transitional step is necessary. A software platform that helps the researchers to conduct their works can help with this step. It can also standardize the features of the developed applications.

Among the various research topics in this area, software frameworks received the least attention. In a survey of 133 papers, only five were related to the frameworks, and only two of them had presented a software platform that can be reused for the development of research applications [12]. This indicates that there is a great potential for improvement in this area as the authors say: “we expect that devising frameworks for the production and operation of medical apps within the big picture of health informatics would receive more interest as the demand for general and scalable solutions for the current challenges increase.”

Furthermore, the commercial solutions have their own limitations. Most of the applications provided in the application stores such as Google Play Store or Apple Store are designed and developed for the end users. For example, Samsung S Health or Google Fit can obtain patients’ data, but it is not possible to customize them for research purposes. In addition, the data obtained from most of these applications is not accessible. Most of the applications that provide data using their APIs don’t allow medical usages. Apple ResearchKit [28] and Apple CareKit [29] are among a few open source tools that are available for medical research.

Apple ResearchKit is designed for developing healthcare research applications. It runs only on devices such as iPhone or iPad that support iOS, and it offers a few customizable modules: surveys, consent, and active tasks. Researchers can customize the Survey module by providing their own questions and answers. The Consent module helps the researchers to inform the patients about the

information that will be collected during the study. The purpose of the Active Tasks module is to collect data from iPhone sensors while the user is performing an activity. For example, it can collect GPS data while the user is walking or running.

Apple CareKit helps researchers to develop medical care apps. It can be used to track treatment tasks on a daily basis, assess patients' progress, and provide communication with caregivers. In comparison to Apple ResearchKit, it provides more modules. Figure 1 shows a screenshot of the Symptom Tracker page of an application that is developed using Apple CareKit.

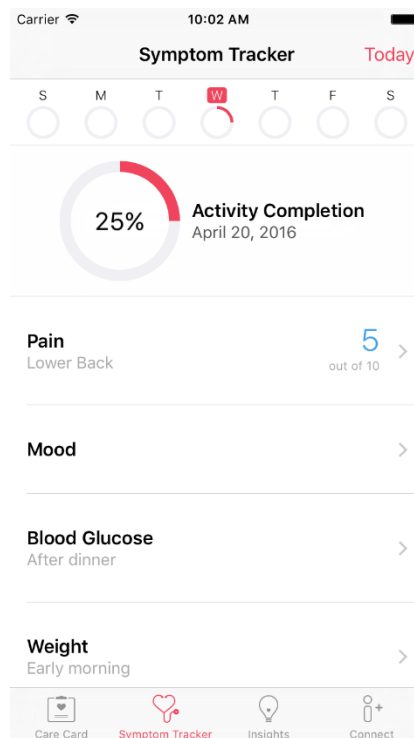


Figure 1. Apple CareKit Sample Symptom Tracker application

Care Card module can be used to manage the tasks that the patient needs to do such as taking medications or changing wound dressings. Symptom and Measurement Tracker module can be utilized for evaluating the effectiveness of the user's care plan using the subjective assessment of symptoms like pain scales and objective measurements such as blood pressure. The main purpose

of the Insight module is to display charts of the relationship between treatment and progress; it can also help the patient to be on track by providing tips or alerts. The Connect module provides a communication channel between the patient and his/her caregivers or doctors. The Care Plan Store module is a database to store the collected data. Documents module helps to create PDF or HTML reports from the obtained data.

Apple ResearchKit provides very few modules to be customized and is limited to iOS. It doesn't contain data management tools. Both ResearchKit and CareKit are limited to iOS platforms. Many potential research subjects use other operating systems such as Android, this limitation may make some research studies infeasible. As the data collected from the sensors is raw data, it won't be easy to do the analysis on the client-side code. Furthermore, these frameworks lack server-side applications and databases; it is not a minor limitation because it is necessary to collect the data on a server and run computational-heavy algorithms, and it is very time consuming to develop another platform for handling these tasks. In addition, iOS programming languages are very different from other server-side programming languages such as C# or Java; therefore, a separate skillset is necessary for someone who is going to utilize these frameworks. Moreover, preserving privacy and security is not addressed in them, and they put the burden on developers.

Berkeley Tele-monitoring is another software framework [13] . It proposed a general-purpose framework to help the healthcare professionals who are less experienced in software development to implement their telemonitoring research applications. The main target of this framework is monitoring applications that run on Android-enabled devices. Privacy, Battery consumption, ease of use, flexibility, collecting health-related data and being fault-tolerant are their main design

considerations. It seems that the main focus of the project is collecting data from medical devices using Bluetooth technology in a fault-tolerant and privacy-aware manner.

Berkeley Tele-monitoring is still a very young project. As it is mentioned in their paper, the server-side of the project needs more work to get the data and store it in a database and preserve privacy. The target of the framework is not developers but healthcare professionals. This means that the framework developers need to predict several requirements of other research works and provide a generalized solution in the framework. We argue that it can be challenging and slows down the development process as adding new features require a lot more considerations and generalization. In addition, the research projects usually come with many unknown factors that would be hard to predict beforehand. For example, if the survey module lacks a feature like an extra note for a question due to design consideration, it would be impossible for a healthcare professional to utilize it, and it would be hard for a developer to add this feature, if not impossible.

EasyHealthApps project introduces generating smartphone apps dynamically [30]. It follows a different trend in the application development. It suggests generating the application by defining the specification in a web-based user interface. A screenshot of this interface is shown in Figure 2. It defines a five-step workflow from “Concept and functionality specification” to “Initializing smartphone app.” It also focuses on “health domain users who are not developers.”

This study is limited from several points of view. First, the created apps are comparable to what already exists in the generator, and it is mostly limited to the UI. Although it reduces some UI development effort, practical usage of the concept requires a tremendous amount of extra effort. In addition, the UI usually needs to be customized to match the user experience. It also lacks data transmission to the server.

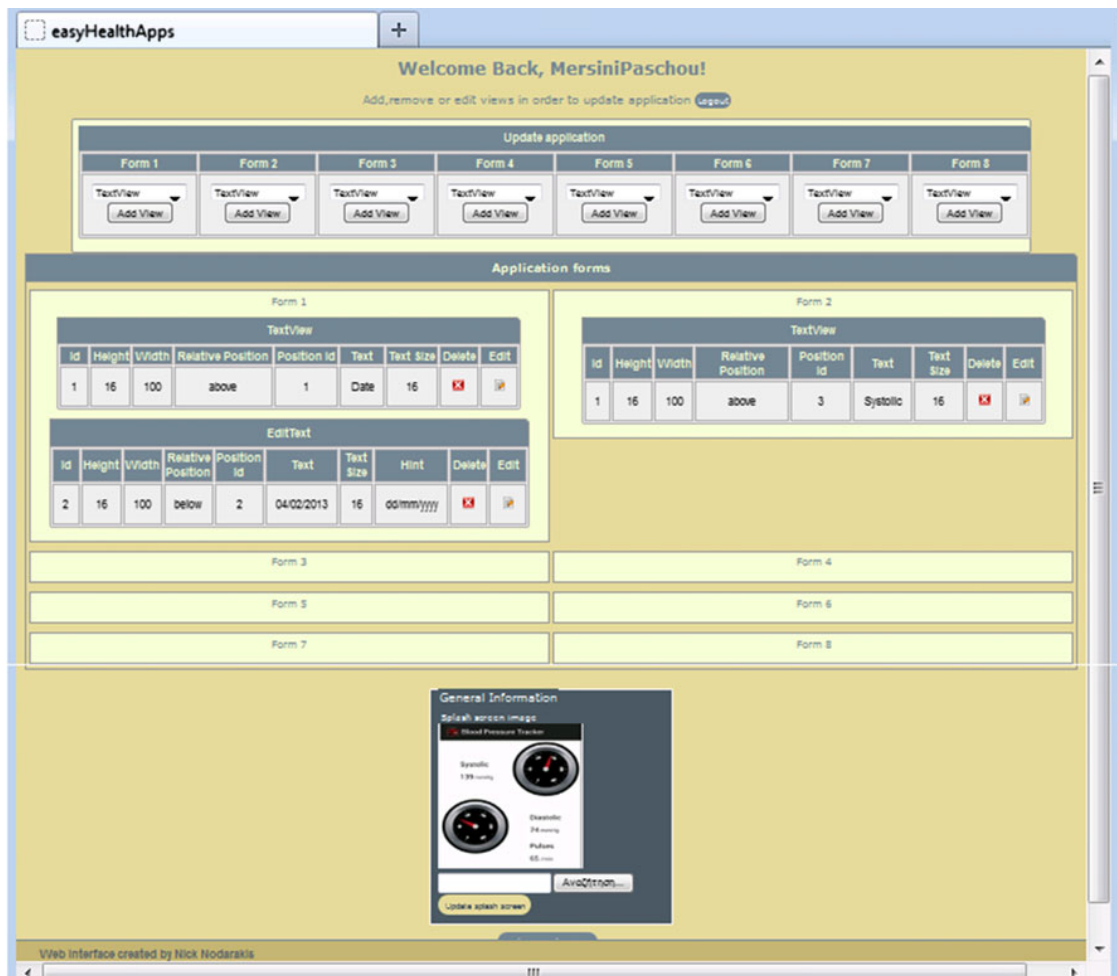


Figure 2. easyHealthApps web based code generator

Moreover, the final generated app is usually not usable without many modifications, and these modifications make it harder to maintain it especially when there is a change in the requirements that needs regeneration. This tool may be good for rapid prototyping, but it is far from being used for developing real apps.

2.3 Breast Cancer

An improvement in the detection or treatment of breast cancer will have a great impact on the life of many people. In addition, it is one of the most prevalent cancers in both developing and developed countries [6], [31], and is the commonest malignancy in women [32]. Early detection techniques have been shown to have a great impact on reducing mortality [33]–[35]. Unfortunately, the false positive rate during the routine examination is very high; as an example, the cumulative risk for false-positive mammography is 21% to 49% after ten mammography examinations [36]. This is partially due to the fact that the result of a mammography (e.g x-ray) does not significantly help doctors with tumors visualization.

For breast self-examination (BSE), it has been shown that using smartphone applications increases BSE in women younger than thirty years old [37]. It is believed that females who practice BSE are more likely to be diagnosed at earlier stages of the cancer development than who do not [38]. Short text messages (SMS) in India have been used as a trial and showed that it can increase BSE among women [39].

Knowing the risks of the breast cancer has the following benefits:

- how often the patient should be screened
- what types of screening she should undergo
- what type of risk-reduction strategies they should consider

There are currently several breast cancer risk calculators available on the Internet such as:

- 1- Gail Breast Cancer Risk Assessment Tool
- 2- Hall Detailed Breast Risk Calculator
- 3- Myriad Risk Table

4- BRCAPRO

However, none of these calculators are accurate enough to give a good risk analysis. Dr. Hall says [40]: “... the final calculated result cannot be compared to real data, which would require a 15-year study of a hundred thousand women, which is impossible. So remember, the results are estimates, and discussing your risk or breast symptoms with your own doctor, is always best.”

Therefore, it is necessary that the doctors be involved in the process of the risk calculations.

A study of smartphone application related to cancer shows that out of 295 apps, the majority of apps targeted breast cancer (46.8%); raising awareness about cancer (32.2%) or providing educational information about cancer (26.4%) was the major focus; apps to assist in early detection (11.5%) and support disease management (3.7%) and cancer prevention (2.0%) were fewer in number; in addition, the majority of the apps were not affiliated with a medical organization that obscured their effectiveness for public usage [41]. Smartphone apps are changing the way people seeking for healthcare information, but there is still a lack of cancer-related applications with scientifically backed data [20]. In addition, there is a lack of evidence-based and medical professional involvements in the development of the breast smartphone apps [42].

Telemedicine is likely to change oncologist’s daily routine because of its benefits [43]. Video conferencing for women with breast cancer living in rural areas has potentials to give them access to professionally led support groups [44]. Using telemedicine for consulting in high-risk breast cancer cases is shown to increase satisfaction among the patients when it is integrated with specialty care [45].

2.4 Augmented Reality for Disease Management

Augmented reality (AR) is a technology that enhances the user's perception from the physical real-world by adding digital information to their views. A common AR application uses a camera as an input of the real world. It detects the objects of interest using image processing techniques and renders more information about it by rendering a 3D model on the camera picture. An AR device, a simple smartphone, monitor or a head-mounted display (e.g. Google Glass [46]), can be used to view this augmented version of the world.

There are two major ways of finding camera position in an AR application: marker-based (MB-AR) and marker-less (ML-AR) [47]. MB-AR is based on finding a marker, usually a black and white picture, as a reference object and estimating the relative camera's position. On the other hand, ML-AR estimates camera's pose using existing objects in the scene without using any marker. Obviously, ML-AR is more challenging as looking at the object from different perspectives makes it harder to find the fiducial points.

New AR devices are introduced in the recent years that has a great potential of influencing the way we use AR. Google Glass is one of the earliest devices that made augmented reality possible [46]. Many research works have investigated its usefulness in healthcare such as sharing view during surgery [48], pediatric surgery [49], biomedical diagnosis tests [50], remote electrocardiogram interpretation [51] or education [52], [53].

AR is probably more widely used in laparoscopic surgeries [54] than heart [55] or breast. However, using AR for breast diseases has a long history. In 1996, a real-time stereoscopic augmented reality (AR) application used along with ultrasound-guided needle biopsy of the breast [56]. Two years later, another application of AR for ultrasound-guided breast cyst aspiration was proposed [57].

AR is also used in conservative breast cancer surgery, and it is shown to be effective [58]. Magnetic resonance (MR) images also found to be useful with breast conserving surgery [59].

Mammograms are considered as one of the best screening tools [60] for years, but the 2D view of them is not as effective as 3D views in some cases.

In 2014, the US Food and Drug Administration has approved new devices for breast cancer screening that can provide 3D mammogram™ images, i.e. tomosynthesis [61]. Research results indicate that these images can lead to a better screening [62]. In addition to 3D mammograms™, 3D CT or MRI reconstructions have been tried for accurate evaluation of breast cancer [63]. AR is becoming a new trend in medicine [64], and the research in this area is gaining more attention recently. Therefore, it is important to consider its potential usage in the future research applications.

2.5 Chapter Summary

Treating chronic diseases has always been a challenging problem. Many smartphone apps are not grounded in scientific methodology so new efforts are made to design apps that can manage chronic diseases based on medically proven protocol. In a recent survey out of 133 papers, only five were related to models frameworks for these applications. Apple ResearchKit and CareKit, easyHealthApps, Berkeley Tele-Monitoring project are among a few attempts to save the software development time for researchers. These projects have their own limitations. Thus, there is a great demand for proposing solutions in this area. In addition, new technologies like Augmented Reality are progressing very fast and open their ways in the healthcare applications. So, it is important to consider them in the future studies.

CHAPTER 3

ESSENTIAL FEATURES AND ARCHITECTURE

3.1 Features List

In this section, we provide an essential list of features that a disease management application needs to have. We use the word “feature” as a replacement for “requirements” because the definition of the requirements should be more precise in the software engineering dictionary. For instance, “system shall support questionnaires” is not a well-defined software requirement. However, we can say that “having questionnaire forms are necessary”, and it can be explained how this feature is going to be implemented.

Some researchers have identified the necessary features of chronic disease management applications as follows: (i) well-designed user interface, (ii) recording patient’s physical and/or psychological conditions, (iii) data security and patient privacy, (iv) motivating patients and psychological support, (v) recommending health improvement behaviors and learning behavior patterns which are harmful or helpful to patients’ condition, and (vi) integration with Electronic Health Records (EHR) and Electronic Medical Records (EMR) [65]. However, studies show that most of the apps neglect to follow evidence-based guidelines such as educating patients [66]. That’s why it is important to have a reference list of features.

We define the features based on a survey of more thirty chronic disease management mobile applications in Google Play Store (Android OS) and Apple Store (iOS). In addition, the users’ opinions obtained by one-on-one interviews and are considered as additional input to define new

features or refine current features. During the study, similar features are generalized as one feature whenever possible, and feasibility of the implementation using today's technology is also considered. Table 1 lists all these features with a short description. They are explained in more detail later in this section.

Table 1. Essential features of a TDMS with a brief explanation

Feature	Purpose
Education	Making users familiar with their diseases
Single Vital Value Records	Capturing values such as height, weight, and blood pressure
Vital Value Visualization	Representing data in an interpretable way
Time Series Data (real-time data) Recording	Recording time-series data like ECG in real-time
Automatic Reading Vital Values from Medical Devices	Automating reading data from devices to ensure accuracy and ease of use
Notification	Keeping doctors and patients informed about events by email, SMS
Reminder	Reminding appointments, taking medications for both doctors and patients
Medication Records	Keeping track of medicines like tablets and capsules
Diet and Activity Records	Tracking information such as calories and fat
Questionnaire	Recording any data that needs to be known about the patient in a certain time or periodically
Video/Voice/Text Communication	Remote communication between doctors and patients for answering questions or taking other actions
Appointment Scheduling	Managing the relationship between doctors and patients
Medical History and Document Records	Recording important health records such as allergies and immunization
Intelligent Analyzer (Decision Support System)	Providing information to doctors or patients by processing patients' raw data
Integration with other systems	Unifying data between different healthcare systems such as EMR/EHR, HIS or accounting

The proposed list may not be the most completed one, but it contains many essential features for a TDMS. It is clear that not all of the features listed here is necessary for a research application, but a ubiquitous TDMS need to support them. Non-functional requirements such as security, privacy, usability, adaptability, accessibility, performance and scalability are discussed later in this chapter. The support for some of these features is explained in Chapter four.

3.1.1 Education

Educating users is a critical way to help patients do self-management. Whether it is before diagnosis or after, patients who are more informed about their diseases care more about it; so, educating patients about their conditions should be considered in a TDMS. For example, breast self-examination (BSE) is important for early diagnosis of the breast cancer; women can be educated to do BSE regularly before being diagnosed with breast cancer, and they may get more information about the breast cancer after diagnosis. Education materials can consist of texts, images, videos, etc. Education apps establish a large portion of healthcare apps in mobile app stores. A TDMS should provide information about the diseases in a way that engages patients in learning it.

3.1.2 Single Vital Value Records

We called any single important information about patients' health such as height, weight, and blood pressure, Single Vital Value. Tracking of the changes is absolutely necessary. In fact, this is a generalization of all single value measurements. Recording a measurement may differ from one

chronic disease to another, but they all have similar characteristics. These values are usually (i) consist of one number and (ii) change during time.

The user may use a medical device to do the measurement and enter the value manually. In addition, the app may read the value automatically from the device and record it. The doctor should have access to this information as soon as they are recorded.

3.1.3 Time Series Data (real-time data) Recording

Some vital values are not just a single number and need to be recorded and showed in real-time. Electroencephalography (EEG) or Electrocardiography (ECG) are two examples of this kind of data. The characteristics are (i) has more than hundreds of values (depending on the device) or sample rate per second, and (ii) needs to be viewed in real-time (iii) automatic interpretation of them is complex and needs special algorithms. In addition, doctors usually need to see the data when the device is connected; it means that it should use an efficient way to transfer a large amount of data in real-time. Furthermore, showing trends and abnormalities requires some machine learning techniques to do automated interpretation. For example, showing a pulse in ECG data needs ECG interpretation algorithms to detect its features such as P, T, Q, R, and S.

3.1.4 Vital Value Visualization

It is not easy to track changes of Single Vital Values without visualization. Visualization of all sort of data is helpful for both doctors and patients. Visualization techniques such as diagrams, charts, gauges can lead to a better interpretation of data. For example, a chart of weight is useful to track weight changes during a time period. The same thing is true about time series data. Doctors would like to see the data and possibly the automated interpretation of the data in a diagram. Easy

Navigation through data is also necessary. Being able to view the data as it is recorded from the device (real-time) view is another necessity.

3.1.5 Questionnaire

One of the best ways of knowing about the patients' health condition is to ask them to answer some questions. In many diseases, there is a pre-defined questionnaire with a single purpose. Answers of the questions can be yes or no, be picked from multiple choices, typed as a single word or a number, typed as a description in a few sentences or paragraphs, marked as selected to name a few. Doctors should have access to the answers to have a more accurate understanding of their patients' situation.

3.1.6 Notification

This feature is a generalization of all messages that needs to be sent to the users. Notification system should support at least three media channels: email, SMS, and Mobile Push Notification. Email is a cheap communication way. Mobile Push Notification sends a message from a server to the users' mobile app. It is very useful because the user needs to be informed about an event when he was not online. Types of messages should be defined based on the system features such as confirmation of registration, cancellation or reschedule of an appointment, change of email addresses, password change and email verification to name a few. Integration of notifications with Analyzer is necessary because it can help to generate useful reports of the users' behavior.

3.1.7 Reminder

Sometimes patients forget to take their medication or forget that they had an appointment with their doctor. The system shall send a reminder notification to the user to inform him about an upcoming action. Medication reminder and appointment reminder are two examples of the required reminders. Reminder module can use the notification feature to keep the user informed.

3.1.8 Medication records

Patients usually have some medications to take every day. Keeping records of the medications with their name, daily dosage, quantity and usage frequency helps patients to take them on-time and helps doctors to be informed about past medications. Names of medications are not easy for patients to enter. So, the system may synchronize with an EMR/EHR system, or it should let the doctors input it. Doctors also need to look up the name from a list because the typing is not convenient. In addition, medication records should be integrated with the reminder system to remind the time of taking a medicine to the patient.

3.1.9 Diet and Activity Records

There are many situations that tracking a patient's diet can help doctors to have a better understanding of their patient's health conditions. For example, tracking diet for diabetes patients can help to determine the amount of glucose they received in a week. The doctor may prescribe new medicines or change the dosage of current medications based on that. Many patients find it hard to enter their diet manually. It is recommended to make an easy-to-use UI for entering foods.

In addition to the diet, tracking of physical activities such as running, riding a bicycle is helpful to compute the amount of calories burnt in a day. A calorie calculator can calculate the amount of calorie based on the activity type and the duration of the activity. Using this computation, a calorie chart can show the amount of calorie that is taken or burnt daily, weekly or monthly. This helps doctors to know if the patient has a balanced life or not. Patient's sleep hours are also a good indicator of the health conditions of the patient. For example, a person who sleeps only 2-3 hours a night probably has a lack of sleep and is more likely to be impatient or feel exhaustion.

3.1.10 Video/Voice/Text Communication

When patients manage their diseases, they may have some questions or doctors may need to check something with their patients. Thus, it is necessary for a TDMS to provide a communication way between doctors and patients. Ideally, a video communication is the best way. Patients may have a virtual (online) visit with their doctors by booking an appointment or just start a communication without booking when they are available. Text messages inside the system are also helpful because healthcare data is sensible, and it is not recommended to use emails or any unsecured media for the communication between doctors and patients.

3.1.11 Appointment Scheduling

Although disease management is mostly patient-centric, patients still need to consult with their doctors regularly. It is more convenient for patients to schedule their appointments with doctors within the app. Therefore, the system shall allow doctors to define their availability, and let patients schedule appointments with their doctors. Patients may choose to visit their doctors virtually using

a video communication service or may go to the physical office. Patients and doctors should be able to cancel an appointment or reschedule it to another time.

3.1.12 Medical History and Document Records

Doctors always need to be access the medical history of their patients. It includes demographic data, medications, immunization, allergies, disease history in the family and lifestyle such as smoking, exercise. In addition, many medical records are images, videos or other file types that need to be uploaded for each patient. So, the system should support uploading files. Files may also be synchronized or be accessible with the integration of an EHR/EMR system.

3.1.13 Intelligent Analyzer (Decision Support System)

Processing of the data obtained from patients leads to finding interesting patterns that can be crucial to their health. It is also a great help for doctors to interpret the data. An intelligent analyzer should work on all types of data in the system. For example, it should read ECG data (Time series data) and find arrhythmias in it, or it should be able to process uploaded Mammograms of patient's breast and detect malignant tumors in it. This part relies on machine learning and artificial intelligence techniques to find patterns.

Since processing of each type of data requires its own strategies and methods, it is not possible to define a generalized analyzer for all types of data. In addition, processing efficiency (time and processing power) can only be achieved by designing customized algorithms for each data. For instance, processing mammograms needs image processing techniques, but processing ECG data needs time-series pattern recognition methods.

3.1.14 Integration with other systems

One of the most important and difficult to implement requirements of a TDMS is integration with other health-care systems such as EMR/EHR systems. Doctors would like to see patients' self-reported data integrated with their EMR/ EHR system. This means that using standards and protocols is important. In addition, some hospital information systems (HIS) has Application Programming Interface (API); a TDMS should be able to talk to these systems, too. Furthermore, payments should be integrated with the current accounting system. Many apps have an export or communication feature that provides data, but it shouldn't be considered as integrated as this data should be imported manually.

3.1.15 Medical Devices Connectivity

It is not easy for the patients to type their vital values anytime after the measurement. They may also forget to record or may make mistakes in reporting these values. Thus, it is necessary that a TDMS obtain the values automatically from various medical devices such as weight scale, blood pressure, glucose meter, pulse oximeter, pedometer, spirometer to name a few. Required values may differ from one disease to another. For example, reading data from a glucose meter device is helpful for diabetes research, and blood pressure data is more useful for CHF. Bluetooth, Bluetooth Low Energy (BLE) and Wi-Fi are very common protocols for these medical devices, and supporting automated reading using them can be very helpful.

3.1.16 Speech to Text

Voice recognition is no longer a luxury feature for software applications. For instance the integration of the voice-function capability into all health-related apps that might be used at the

workplace is suggested in one research work [67]. In addition, typing on a small mobile screen is not easy for many users. Speech to text feature helps users to use their voice to type things on the user interface by saying words or sentences. Users speak into a microphone, and the system processes the voice and converts it to text and put it in UI.

3.1.17 Online/Mobile Payment

If the user needs to pay for services, then the payment inside the system is necessary. For example, patients may need to pay for visit costs, and doctors can charge patients, and the payment process should be done electronically. It is unlikely for a research application to need a payment system, but a TDMS need to have this feature because doctors would like to receive money in exchange for the services they offer. An online payment processing can facilitate this process.

3.2 Quality Features

The functionality is not the only important aspect of a software application. The quality of the end product is also important. Two systems may have the exact same functionality with very different qualities. For example, the operation of saving an item in the database may take ten seconds in one application, and it may take one millisecond in the other one. It is true that the quality of the software system depends on the hardware as well, but the software design plays an important role. For instance, if the performance is not considered in the design of a software application, running on a more powerful hardware won't help much. Therefore, it is very important to consider the quality in the design.

We don't use the word "non-functional requirement" to define our quality features as defining exact requirements needs a more precise definition. For example, to define performance as a non-

functional requirement, it should be testable. Therefore, “system should work fast” can’t be a non-functional requirement. However, we can say good performance is a design consideration, and then it is possible to explain how this consideration is applied to the software system.

3.2.1 Software Development Methodology

Software development methodology has a direct impact on the quality of the final product. Tailoring the methodology to the software project is a good software engineering practice because each project is unique. We have used a customized version of Scrum methodology to plan and develop our platform. The length of each sprint was one month, and it usually consisted of the research and development of one or two features. Furthermore, we have chosen Test Driven Development (TDD) as the major methodology for the development of the main parts of the system such as the framework and the services. As a result, there are more than three hundred test cases in the system. TDD not only helped us to deliver a more reliable platform, it saved us a lot of hours of development because we had to redesign the features several times during the project.

3.2.2 Security and Privacy

Security is a very broad topic in the software industry. Requirements such as authentication and authorization are discussed deeply in software security, and it is out of the scope of this study; however, there are several points that need to be considered in a TDMS.

Authentication

Authentication is to ensure that the person is who he/she declares is. A unique username and password is the most common approach to implement it. The authentication feature of the platform

is explained in more detail in the previous section. Here we discuss two other aspects of this concept here.

First, considering integration with other systems such as EMR/HER systems is critical in design and implementation of this feature. Many of current hospital systems rely on the industry best practices for security. Using authentication standards like OAuth 2.0 is recommended for authentication. In addition, it simplifies API design for the whole system.

Second, authentication should support single-sign-on feature to simplify the access of the users. This feature allows users to see all separate systems as one by accessing all systems with a single login to any of them. Doctors would like to use their current system and have access their patients' data at the same time. It is burdensome to ask them to keep login to each system separately and maintain two sets of username and passwords. In addition, patients may have separate applications to enter their data because applications focused on one area are generally easier to learn and use. For example, there can be an application for diabetes and another application for CHF. It is more convenient for the patient to login in one app and it let him access the other app without login.

Authorization

Authorization defines access policy to any resource of the system (including services and data) for each user. The importance of authorization is higher for healthcare systems. A TDMS should have a flexible, configurable and programmable authorization to simplify defining policies while allowing implementation of complex access rules.

HIPAA Compliance

Health Insurance Portability and Accountability (HIPAA) Act of 1996 sets the standard for protecting patients' sensitive health information. This information is called PHI (Protected Health Information). Billing information, MRI scans, appointment scheduling notes, phone records are

some examples of PHI data. In the United States, the system that works with the patients' data needs to comply with HIPAA rules and regulations [68]. As there is no certification entity that can issue a compliance approval for software developers, developers need to study about HIPAA-compliant apps themselves [69]. A HIPAA-complaint software platform can help developers by reducing the risk of potential privacy issues.

3.2.3 Extensibility

One of the most important quality features of a software platform is extensibility. We define extensibility as an ability to add more features without doing major changes to the architecture. We have considered several things to make this platform extensible. As the goal is to provide a framework for development of other applications, it should be easy for other developers to add modules and define user interfaces.

3.2.4 Reliability

Reliability is defined as the ability of the system to perform consistently even in the case of the failure. The reliability is not just the concern of the software but the entire system. The important thing about the reliability of the system is that the design should allow the deployment without one point of failure. It means that if any server that runs the code goes down, there should be other servers to handle the requests. In addition, the application may automatically restart to make sure that the user always has access to the system.

3.2.5 Scalability

Scalability is defined as the ability of the system to serve more users as the number of the users grows. Scalability is usually a concern of the entire system and deployment. There are two types of scalability: vertical and horizontal. Vertical scalability is related to having a more powerful hardware (i.e. having a computer with more memory or processing power). Horizontal scalability is about adding more servers with almost similar hardware configuration. For example, we can make the system scalable by adding more web servers and a load balancer server can distribute the web requests among these servers; we can add more servers (manually or automatically if it is in a cloud) when the network traffic is high. Vertical scalability is usually not a software concern, but the software architecture should consider the best practices for the horizontal scalability.

3.2.6 Maintainability

The maintainability is one of the major concerns in the software. We define maintainability as the ability to change the system's functionality without making major changes to the framework. Therefore, the system should be configurable. It means that the administrator or the supporter of the system should be able to change its behavior without making major changes. In addition, the code should be well-commented and easy-to-understand to make the changes easier. Automated test cases also play an important role in the maintainability of a software application.

3.2.7 Performance

Performance is one of the main quality features that is usually in contrast with software engineering practices. In other words, increasing performance usually requires ignorance of the software engineering practices. For example, there should be a queue system between the processing layers

to make the system scalable, and this puts processing overheads on the system that may result in the reduction in the performance. Another example is to write a compression algorithm where developing the code requires high performance. In this case, the minimum number of classes or variables needs to be defined, and it may make the code less-readable; it is clearly in contrast with the maintainability.

3.2.8 Portability

We define portability as the ability to run the user interface on different operating systems with minimum modifications. The application shall run on iOS, Android, Windows or Linux platforms as the users may have access to only one of these operating systems. In fact, this definition excludes the server code running on various operating systems. In other words, the end user shall be able to use the system on different devices, but the server-side code may not be portable.

3.2.9 Reusability

We define reusability as the ability to reuse the current platform features without making major modifications to it. It is very important for a software framework to be reusable. If the framework features or components are not reusable, then it would be hard for the developers to gain benefits by using of them, and they need to develop things from scratch. Therefore, reusability plays an important role in reducing the development time. It also impacts the reliability. The more reusable the code is, the more reliable it becomes. It is because the reusable components will be used in various components and go through a more rigorous testing, and it reduces the chances of having bugs.

3.3 Perspective

This section discusses the big picture of the system. This system has two major subsystems: A mobile application and a website. Figure 3 shows the main perspective of the product. In the patient side, patients can use their smartphones to connect to the system and interact with the system. For example, they can answer questionnaires and provide the required information for breast cancer

risk

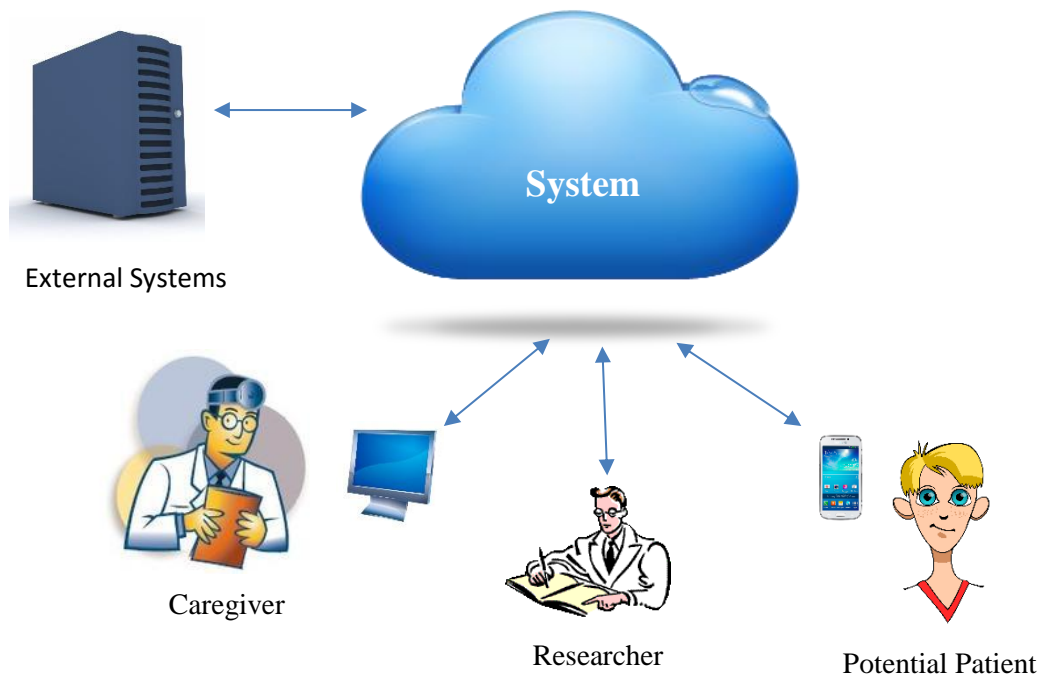


Figure 3. The general perspective

assessment to the system. The information is stored and analyzed in the System, and it is accessible for the patient and the permitted doctors. Doctors can connect to the server using their office or home computers to access their patients' health records anytime. They can also provide feedbacks to their patients or visit them online. The system can also interact with other systems (External systems) using its API. In addition, the researchers can access the data stored in the database to

analyze it and conduct their research on it. The researcher is not a system user, and he usually does not access it using the provided user interfaces, but he can have direct access to the database records and analyze the data by running his own algorithms.

3.4 Architecture

The general idea is to have a website with a responsive user interface that can be opened in web browsers of various platforms such as desktop computers, tablets, and smartphones. The idea is that the users (for instance, those who have the potential of developing breast cancer) register in the system and enter their information by answering some questionnaire forms. Doctors can also register in the system and see their patients' information. An analytics part of the system will help doctors to analyze the data obtained from the patients or help the researchers with validating their research hypotheses.

In this section, we explain the overall architecture of the proposed platform. Figure 4 depicts a conceptual view of this architecture. This picture is a logical design presented for better understanding of the architecture, and it doesn't represent all actual classes. In the following, we discuss all parts of the diagram in more details. This architecture consist of several major parts:

- 1- "Framework" is a code library that provides a library of common functions and classes that can be used for the development of any application.
- 2- "Entity Layers" is a 3-tier architecture; it consists of all classes related to data access, business logic, and service layer.
- 3- "Server-Side API and UI": this is the service-side UI layer and web service layer for all APIs.

- 4- “Client-Side App and UI” is the client-side web UI part. It is the front-end contains all views and designs.
- 5- “Entity Shared library” is a library of entity descriptions that is shared between all parts.

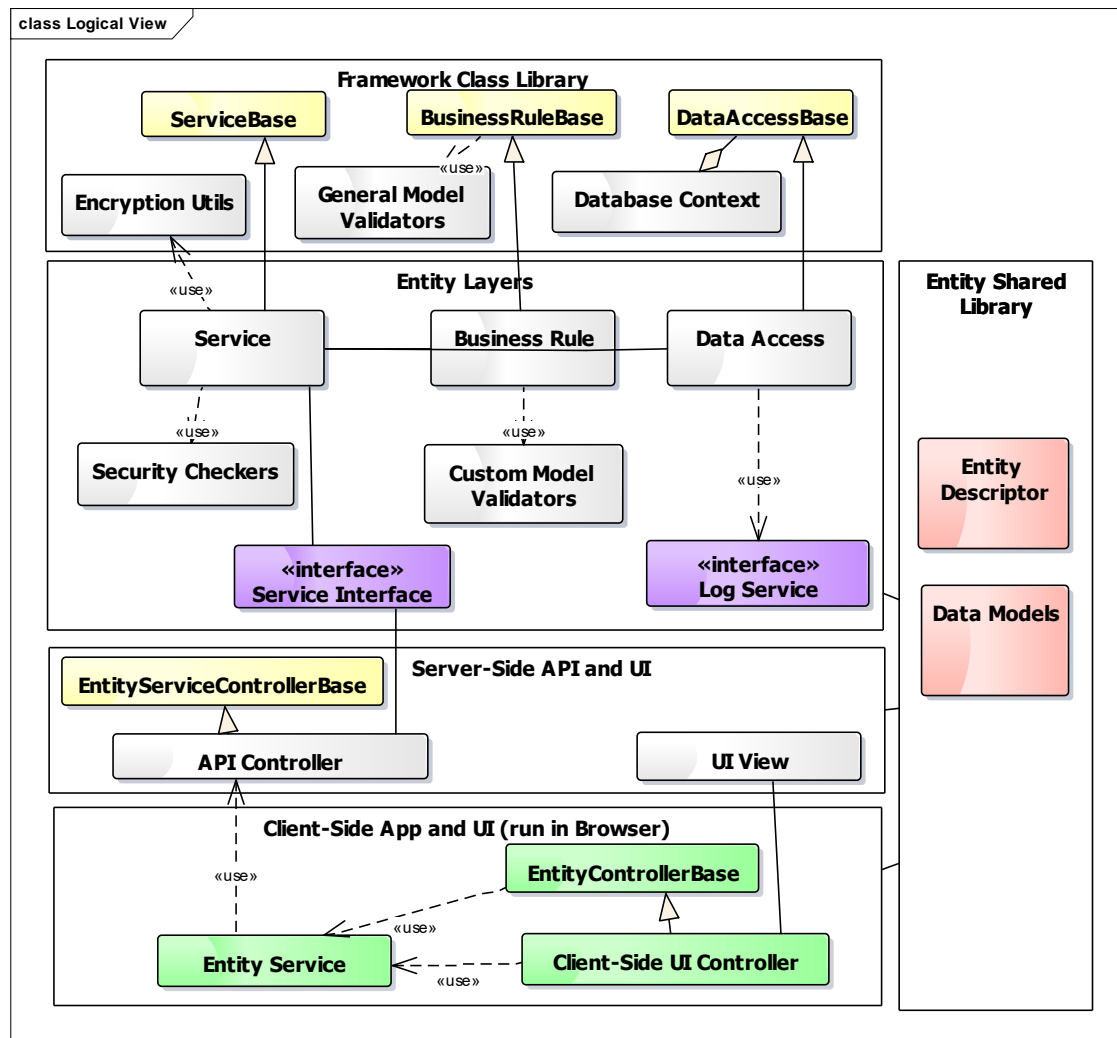


Figure 4. Overall system architecture

These parts are explained in more detail in the following sections.

3.4.1 Framework

We have used Framework design pattern to create a library of classes and methods that should be used in many parts of the program. The base classes are provided in Framework for inheritance. *ServiceBase* has CRUD (create, read, update, delete) functions and checks the authorization of the user before executing a service function. *BusinessRuleBase* contains functions that are useful for checking the validation of the data models. It automatically checks for not null values and integer minimum and maximum and string lengths. *DataAccessBase* contains CRUD functions to access the database. It has access to NHibernate Context and can create and execute queries in HQL (Hibernate Query Language). The interface defined between classes is omitted in the diagram for readability.

Since we are using NHibernate as the main component to access the database, the framework can access all major RDMS (Relational database management systems) such as SQL Server, MySQL, Oracle or any ODBC connection. However, we have used SQL Server in our main implementation due to the fact that many healthcare companies adopted it, and there are many 3rd party tools for it like RedGate Source Control that we have used to keep tracking of changes in the database and having making migration SQL files.

EncryptionUtils provides AES-256 encryption function for encrypting and decrypting sensitive information such as history records. It can encryption a string or and uses .NET AesCryptoServiceProvider that is compatible with FIPS (Federal Information Processing Standards). *General Model Validators* contains some utility functions to validate model properties. It generally checks for null fields, string lengths, number (byte, short, integer, long, float, and double) and DateTime minimum and maximum values and some advanced checks like duplication

of the record in the database. *NHibernate Context* manages to access the database using NHibernate library. It facilitates creating sessions and accessing several databases at the same time with NHibernate.

3.4.2 Entity Layers

In *Entity Layers*, we have implemented n-tier pattern to separate classes in different layers. These layers are not shown in the diagram. For each entity (for example User or Medical History), we create a separate class in each layer that inherits from its base class defined in Framework. So, each entity can do all CRUD functions without further coding. Entity layers are all C# classes and they are not bind to any specific technology like ASP.NET MVC (Model-View-Controller). So, all application services can be executed using any .NET runtime.

Service class implements an equivalent interface to provide services that are not CRUD in nature. For example, User entity is a fairly complex entity as it has to deal with the authentication so many of its services are not CRUD; we have VerifyEmail service that helps to verify emails or ResetPassword inside User Service class. It also uses various *Security Checkers* classes to check authorization of the user for executing the service before running it. For example, before executing the delete service for deleting files in the user profile, it uses OwnerDoctorSecurity checker class to make sure that the user who is going to do it is a doctor and owns that file. It also filters data before obtaining it from the database based on the roles defined the user. For instance, a patient can't access other users' medical history records (they can only view and edit their own medical history records).

Business Rule class of an entity implements extra validations specific to that entity. For example, we need to check the password entry or username with a specific regular expression in the user

registration function, and it should be implemented in its related business rule class because it cannot be generalized in Framework. Other validators that can be used across several entities are implemented in several *Custom Model Validator* classes.

Data Access class can be used to implement extra data access functions other than CRUD like calling a stored procedure. Since CRUD is the most common use case for accessing database, this class usually has an empty body for many entities as it derives its functionality from the base classes. It can also log the information and functions used in the application. Features such as Audit can be added to the system as well. Logging functionality is provided in Framework. *Log Service Interface* provides access to writing logs for all entities.

3.4.3 Server-Side API and User Interface

Since our system is largely web-based, we have used ASP.NET MVC to provide access to the service interfaces over the web. For each entity that needs to be presented on the web, we create an *MVC controller*. This controller is based on MVC Web API. It is optional to create a controller for an entity because not all entities requires to be accessed from the client-side code. For instance, services related to push notification may not be exposed through the web interface since it should only be called from other server-side services and not the user interface.

EntityServiceControllerBase generalizes access to CRUD functions. Each entity will get functionality of Restful API for the view, insert, update and delete by inheritance from this class. It facilitates converting database models (Data Model) to JSON format to be consumed by the user interface. Converting JSON objects coming from the user interface to database models used in the service layer is another functionality of this class.

MVC views use ASP.NET Razor engine to produce views. Views in this design are considered to be user-independent and database-independent; it means that no data related to one user or stored in the database will be included in any view; so, all views can be served as static views, but Razor simplifies some tedious tasks like layout design and multilingual texts inside the view. Contents of all views will be filled by another request to the web API through service layer in JSON format. MVC views also have access to entity descriptors and can generate some HTML codes using it. For example, the code inside the view to have an editor to the username field is one line, but it translates to several lines of HTML code containing an input field and its required validations.

3.4.4 Client-Side Code and User Interface

Our user interface is heavily based on AngularJS. AngularJS is a JavaScript library that simplifies development of the client-side application. Our UI (User Interface) is a combination of ASP.NET views and AngularJS single page application. Pages outside of user panels (pages before login) are like regular web pages and have to have a routing inside of the server-side part. After login, all pages are routed and handled using AngularJS routing to gives the feeling of a native application to the user.

AngularJS has its own MVC architecture. In our application, models are JSON representation of our C# models that should be retrieved from the API. Views are rendered version ASP.NET MVC views. Thus, the client-side views and models have the same representation of the server-side; however, we implement a specific AngularJS Controller for each view.

AngularJS Controller keeps the client-side logic separated from the client-side view. Functions for navigating between pages, showing messages, checking client-side validations should be implemented in this controller.

EntityControllerBase contains some functions that are common between many entity controllers. For example, many controllers want to do CRUD and show an appropriate message to the user. In addition, it has some base functions to show edit forms and lists. In fact, it reduces the required code for AngularJS controllers.

Entity Service in AngularJS App is to call all web API in a generalized format. It can do CRUD in addition to executing any other service inside the MVC Controller. Angular Controllers uses these services to access data and do changes on the server.

User Interface

We decided to do a mobile-first design to be able to run the system on various devices without re-designing or re-developing the whole UI code. Thus, Tweeter Bootstrap is adopted to create this responsive UI. JQuery is a dependency of this technology so its functions will be available to the developers. In addition, we have used several jQuery and bootstrap plugins to avoid re-inventing the wheel in UI. No commercial license component used in our code. Figure 5 shows an example of this user interface for a breast cancer risk assessment application.

The screenshot shows a web application interface for a breast cancer risk assessment. The top navigation bar includes the 'Hamad' logo, 'Home', 'Back', 'Feedback', 'Voice Command', 'Account', and 'Logout' links. A dark sidebar on the left contains a 'Dashboard' link and a 'Medical History' section with sub-links for Family History, Risk Assessment, Risk Assessment 2, Risk Allergy, Current Medications, Immunization, Past Medical History, Social, Risk Assessment 3, and Sign. The main content area is titled 'Physical Exam Initial Assessment for high risk breast clinic'. It features input fields for 'General condition' and 'Vital signs'. Below these is a 'BREAST EXAM' section with a list of clinical findings, each with a corresponding input field: 'Clinical Breast Assessment', 'Normal', 'Dimpling', 'Fixation', 'Peau D'orange', 'Erythema', and 'Retraction'. To the right of these fields is a diagram of a female torso showing the right and left breasts.

Figure 5. Screenshot of the application for breast cancer risk assessment form for doctors

3.4.5 Shared Library

The first part that is shared across almost all other component is depicted as Shared library in the diagram. Since Framework is a group of utility classes that are common in many parts of the application, it doesn't have a dependency to the Shared library.

Data Model represents the data structure of the entities the way it should be stored and retrieved from the database. They are C# classes and the data type of the properties matches data types in the database. It also comes with NHibernate HBM XML files that describe the mapping. We avoid using complex mappings (like n-to-n or 1-to-n) to obtain better performance. So, most of the joins and relations are managed in the DBMS using predefined SQL views. This design decision may

also help to adopt NoSQL databases such as Cassandra that doesn't support joins and works with denormalized tables to perform well.

Entity Descriptor contains information related to the entity and its properties such as field caption, length in the database, data type, and editor type to name a few. This information is useful to other layers. For example, MVC Views uses this information to create field editors with all required validations with one line of code.

3.5 Deployment Model

Amazon is one of the most popular cloud providers in the world, and we have adopted Amazon cloud solutions as our main deployment option. Figure 6 illustrates the deployment model of our system. The website is hosted on an Amazon EC2 (Elastic Cluster) node. It's a Windows 2013 Server machine with IIS (Internet Information Server) to execute ASP.NET applications.

Since files can be a big load on the web server, we decided to use Amazon S3 as a file server to store and serve user files. The purpose of this server is not to serve our static files, but it is a host for user files such as medical documents in a secure environment. We have used S3 AES-256 to encrypt all files on the server. Furthermore, all user files are considered as private files, and they only can be accessed with a token obtained from the server. This token is provided by an entity called AppFile in our system that checks all access permissions before issuing it. As a result, although files are completely managed by Amazon S3, accessing them is controlled by the service layer. In addition, the client-side program uploads and downloads the files from the file server directly. Thus, the load on the web servers is minimum.

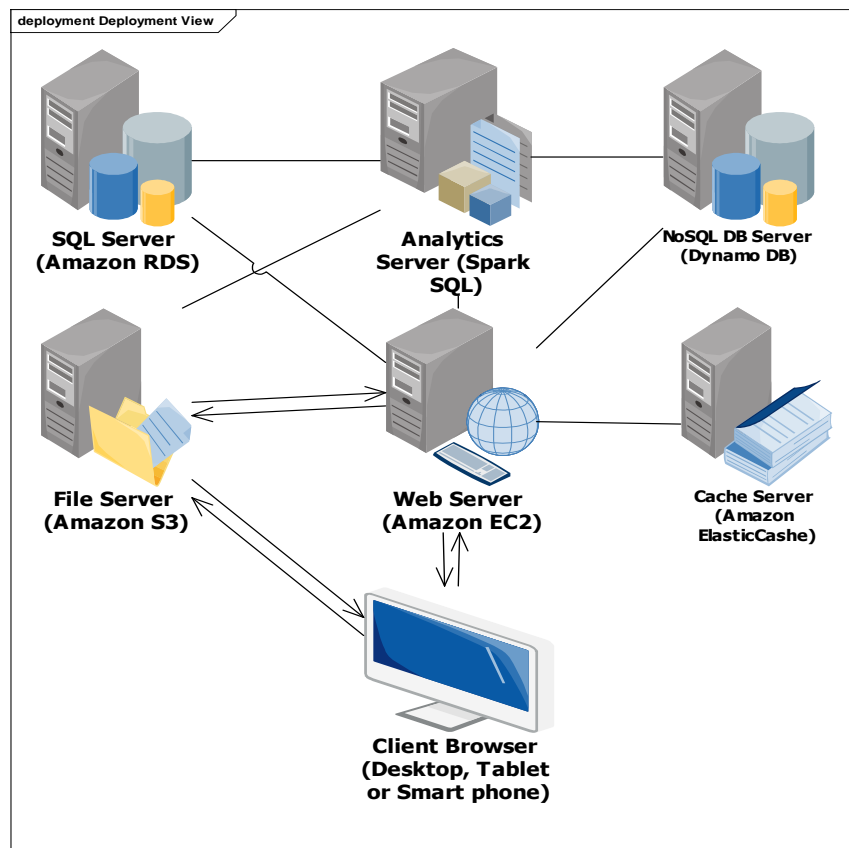


Figure 6. Deployment Model

As it was mentioned before, we have used SQL Server as our database. Therefore, Amazon RDS (Relational Database Service) seemed to be a good choice to have our database in the cloud. Amazon RDS Multi-AZ feature can provide a good availability for the database server by running a shadow instance in the case of the server failure.

Our system's Analytics Server is based on Spark SQL. The purpose of this server is to fetch the required information from the databases and do the required analysis. For example, to calculate the breast cancer risk for each patient, it can retrieve patients' data from the SQL Server database, and store the results in the same database after computing the risk. The web server can trigger starting this computation in the Analytics server.

Client browsers are web browsers and are available on many operating systems such as Windows, Mac, iOS, Android. Websites addresses are hard to memorize for many users, but the users can access the system using a native client-side application without knowing the website address. We packaged an Android application using CrossWalk technology to open the website inside the app. In addition, it can access the smartphones' native APIs like Bluetooth API to connect directly to medical devices.

3.6 Chapter Summary

A software framework for developing disease management applications should have several essential features. We have compiled a list of these features as follows: education, vital signs, time-series data, vital value visualization, questionnaire, notification, reminder, medication records, diet and activity records, video/voice/text communication, appointment scheduling, medical history and documents, intelligent analyzer, integration with other systems, medical device connectivity, speech to text and online payments. In addition to the functional features, quality features such security, extensibility, reliability, scalability, maintainability, performance, portability and reusability are important and should be considered in the architectural design of the framework. We proposed a general architecture, and we followed software development methodologies to develop a framework for these applications. A sample deployment model is also described in this chapter.

CHAPTER 4

THE PROPOSED PLATFORM

4.1 Base Platform

There are many software features that are common for different disease management applications. For example, access to the patient records should be limited to authorized doctors, or doctors should be able to visit patients online and see their medical information. We call these features as general use cases of the platform.

Figure 7 shows the use case diagram for the important general use cases of the Doctor actor in our platform. A doctor can fill his profile out so that his patients can see define a calendar of his availability and visit his patient. This is very similar to a telemedicine platform. This is also related to the features explained in the previous chapter. This diagram shows the common use cases and does not show disease-specific use cases. For example, recording the insulin dosage for diabetes patients is not represented in this diagram. The extended use cases are conceptual use cases and are shown in the diagram for more clarity.

Figure 8 depicts the use case diagram for patient users. Use cases marked with a double star are use cases that need to be defined per each disease. This diagram does not cover all the use cases such as recording activities and eating habits to be readable.



Figure 7. General use case diagram for the doctor actor

A typical scenario can be defined as follows: whenever a patient feels that she needs to contact a doctor, she can schedule an appointment with her doctors. In addition, doctors can enroll their patients into the system and schedule an appointment for them. Doctors can access medical history forms of a patient or see their risk assessment questionnaire answers.

To schedule an appointment, the doctor needs to put their license information in the system. They can enroll their patients or add them by their phone numbers or email address. Doctors can also access patients' profiles to evaluate their health conditions.

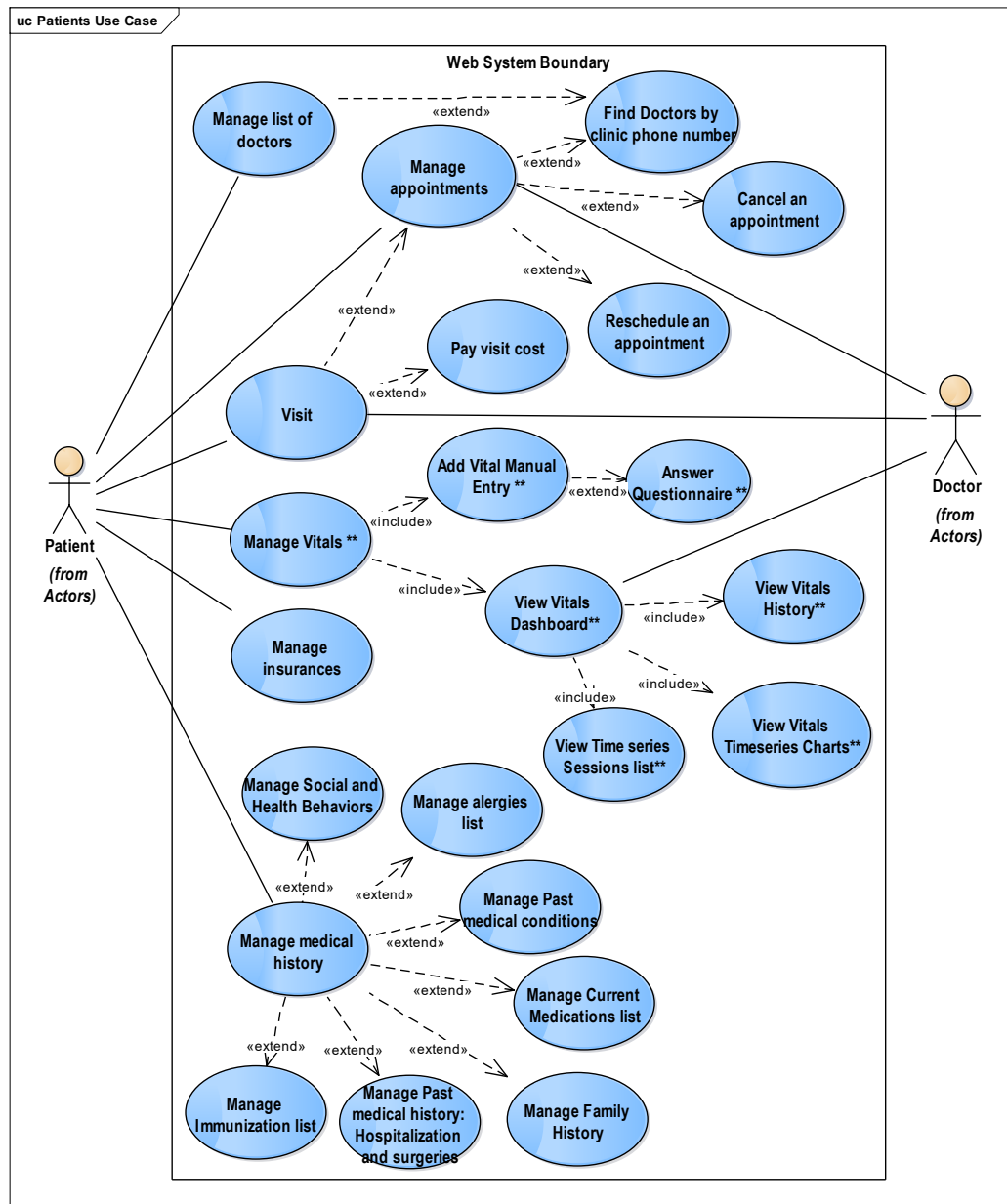


Figure 8. General use case diagram for the patient actor

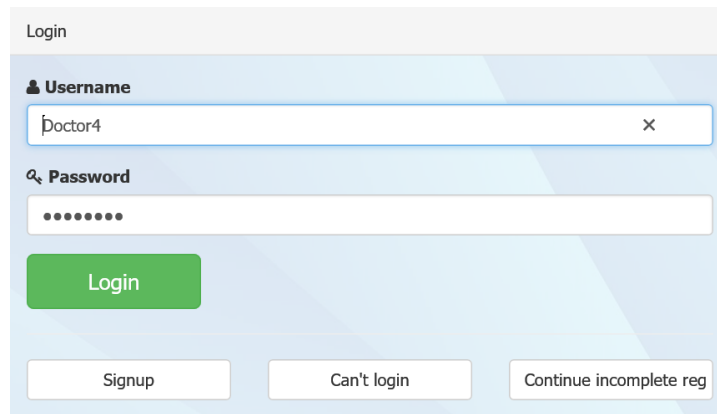
Doctors can mark their availability on their calendar so that patients can book an appointment with them, or they can book an appointment with their patients. Afterward, they can call their patients at the scheduled time. Video communication part of the system allows virtual visits over the web and mobile phones.

The features of the framework that can be used to implement an application based on the above scenario is explained in the rest of this chapter.

4.1.1 Authentication

There are four forms related to the authentication. “Login,” “Signup,” “Can’t login,” and “Continue incomplete registration.” These forms are very similar to many other software applications. Thus, we try to focus on the different functionalities here.

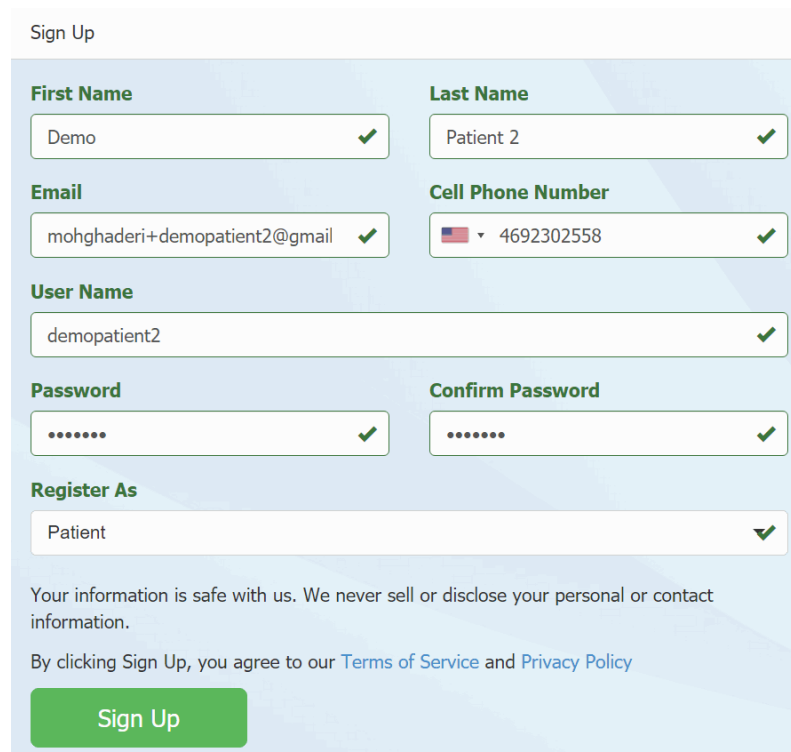
Login: To protect users and preserve privacy, all users of the system shall have an account and login to their account before using the system. The login page is the first thing will be shown to the user that is depicted in Figure 9. By entering username and password, the user, whether he is a doctor or a patient, can access the system. The system finds the role of the user and shows him the related menu. In addition, the user can navigate to other authentication related forms, too.



The screenshot shows a login form with a title bar labeled "Login". Below the title bar, there are two input fields: "Username" with the text "Doctor4" and a clear button (X), and "Password" with masked characters (dots). Below these fields is a green "Login" button. At the bottom of the form, there are three buttons: "Signup", "Can't login", and "Continue incomplete reg".

Figure 9. Screenshot of the login page

Signup: Any user should be registered in the system before using it. Signup page registers both doctors and patients. Figure 10 shows the view of this page filled with some sample data.



The screenshot shows a sign-up form with a title bar labeled "Sign Up". The form contains several input fields, each with a green checkmark indicating successful validation: "First Name" (Demo), "Last Name" (Patient 2), "Email" (mohghaderi+demopatient2@gmail), "Cell Phone Number" (USA flag icon, 4692302558), "User Name" (demopatient2), "Password" (masked), "Confirm Password" (masked), and "Register As" (Patient). Below the form, there is a green "Sign Up" button. At the bottom, there is a disclaimer: "Your information is safe with us. We never sell or disclose your personal or contact information." and a link: "By clicking Sign Up, you agree to our [Terms of Service](#) and [Privacy Policy](#)".

Figure 10. Screenshot of Signup page where users can register in the system

We have used E.164 format for phone numbers that is a standard for international numbers [70] to let the international users register in the system.

Can't Login page guides the users to be able to access their account. If they have forgotten their password, it redirects them to the "Reset Password," and if they had an incomplete registration, it redirects them to the registration form. If the user never registered himself into the website, **Continue Registration** form can guide users to provide additional required information and register. One possible scenario is when a doctor registers his patients and the patient needs to complete this registration by providing more information. Figure 11 shows a view of these forms.

The figure consists of two screenshots of a web application interface. The top screenshot is titled "Can't Login?". It contains a light blue background with a white header bar. The main content area has a light blue background with a white border. It includes a text block: "If you are registered by your doctor, but you don't have a user name and password, you need to complete your registration before login." Below this is a button labeled "Continue incomplete registration". Another text block says: "If you have forgotten your password, you can reset your password." Below this is a blue button labeled "Reset Password". At the bottom are two white buttons: "Login" and "Continue incomplete registration". The bottom screenshot is titled "Continue Registration". It also has a light blue background with a white header bar. The main content area has a light blue background with a white border. It includes a text block: "If you are already registered by another doctor or an administrator, you can easily continue your registration in this page". Below this are two input fields. The first is labeled "Cell Phone Number" and contains the text "+14692302558" with a green checkmark icon to its right. The second is labeled "Phone Verification Code" and contains the text "123456" with a green checkmark icon to its right. Below these fields is a blue button labeled "Check Phone Number". At the bottom are two white buttons: "Login" and "Can't Login?".

Figure 11. Screenshots of the "Can't Login" and "Continue Registration" pages where user will be guided to access his account when he can't login to the system

Reset Password form can help users to who are already registered in the system reset their passwords. The process is that it asks the user to provide his email or phone number at first. Afterward, the system sends a message to the user with a reset code. The user opens his email or text messages and writes the code inside “Reset Password Code” field and sets his new password. He can now log in using this new password. Figure 12 shows screenshots of this form.

Reset Password

Please enter your phone number or email to receive a reset code. Use this code to change your password

Reset password by ☒ Email ☐ Phone Number

Email

mohghaderi@gmail.com ✓

Send Reset Password Code

Reset Password

A reset password code has been sent to your email. It may take a few seconds to get there. Please check your email or phone number and use the reset code to change your password.

Reset Password Code

392049

New Password

New Password

Confirm new Password

New Password

Change Password

Figure 12. Screenshots of the platform to recover forgotten passwords

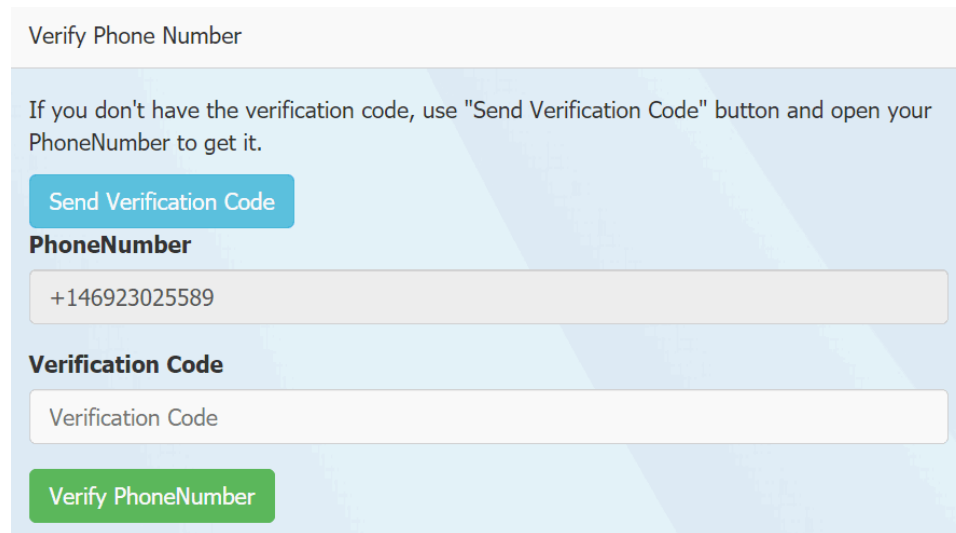
As it is shown in the pictures, the framework provides a complete authentication system for the required applications. It ensures security before accessing the accounts and information, so it reduces the programming time and risks of mistakes in copying this process in different platforms.

Email and Phone Number verification: In order to ensure the entered phone number or email belongs to the user, we send the user an auto-generated verification code and ask him/her to enter the code in the system. This code is a six digit random number that is hard enough for guessing and easy enough that a less experienced person can write down or remember. The related user interface for phone verification is shown in Figure 13. The Email verification page has a similar interface. The process has the following steps:

- 1- The user requests email or phone verification
- 2- The system generates a random number and sends it to the provided email or phone number
- 3- The user reads the number from his email or text messages and enters to the system.
- 4- If the number matches with the generated code, email or phone number would be marked as verified in the database. If not, the user can repeat the process by requesting another number.

To improve the user experience, step 3 can be automated in the mobile applications. This way, the mobile application will read the code from the message or email and automatically verifies it.

The developers can limit the certain functionality of their applications to the verified users. In addition, it can be shown on the user interface. For example, doctors can see if the registered user has provided a valid email or phone number or not and decide whether he should give him/her services.



Verify Phone Number

If you don't have the verification code, use "Send Verification Code" button and open your PhoneNumber to get it.

Send Verification Code

PhoneNumber

+146923025589

Verification Code

Verification Code

Verify PhoneNumber

Figure 13. User phone verification page

4.1.2 Main Menu

Doctors' Menu: doctors can see a menu to use various features of the application after login. This menu provides access to all other forms. There is a menu on the left side of the screen that can be used to access different parts of the system. The main screen (dashboard) also provides the same functionality. It is designed to have bigger icons and help the user visually. Figure 14 shows a view of the main menu for doctors. The description of the menu items is explained in the next sections of this document. "Professional Info" item has a sub menu that is shown in Figure 15. It simply provides access to more detailed information about the doctor.

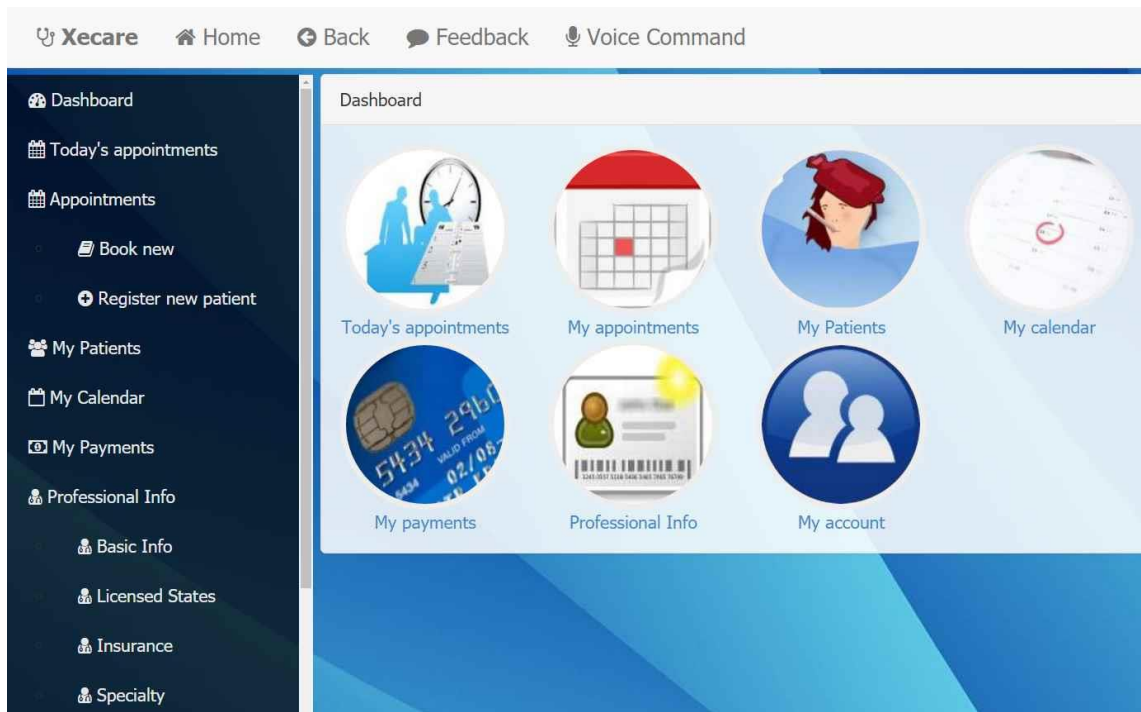


Figure 14. Screenshot of the doctor's main menu

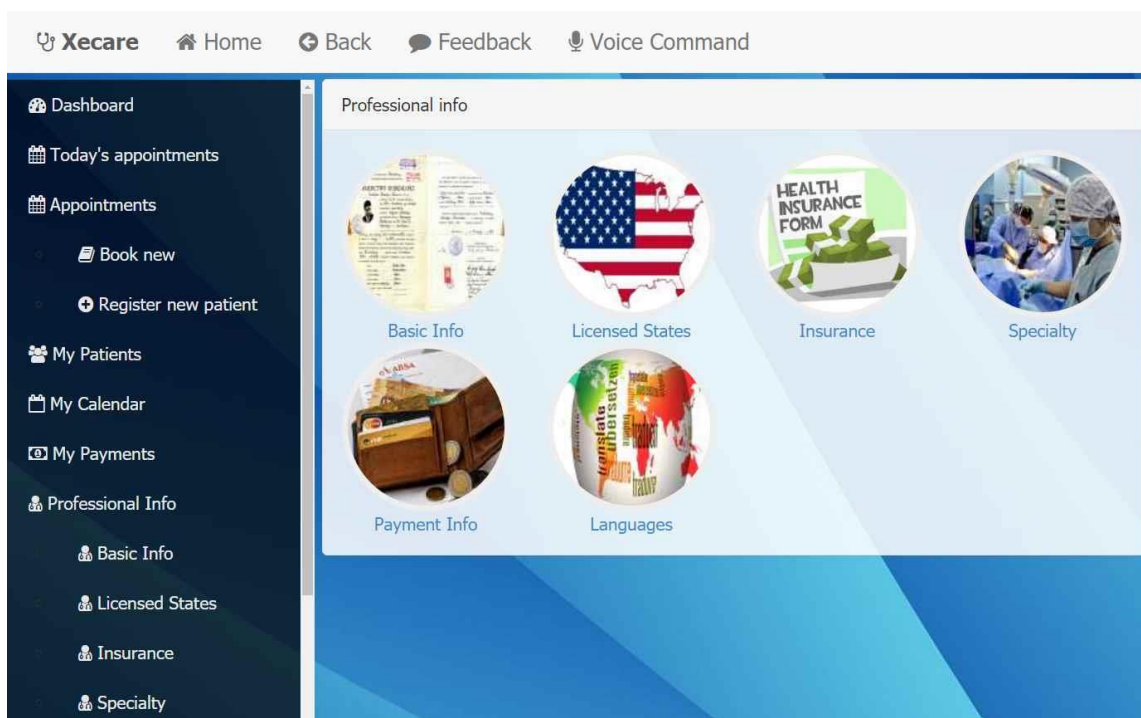


Figure 15. Screenshot of the doctor's professional info menu

Patients' Menu: Use cases of patients is different from the use cases of doctors. Therefore, patients should access a different menu. This menu helps them to navigate through the system. Figure 16 illustrates the menu that a patient sees when he successfully logs in. Patients' Menu is very similar to the Doctors' Menu from the functionality point of view.

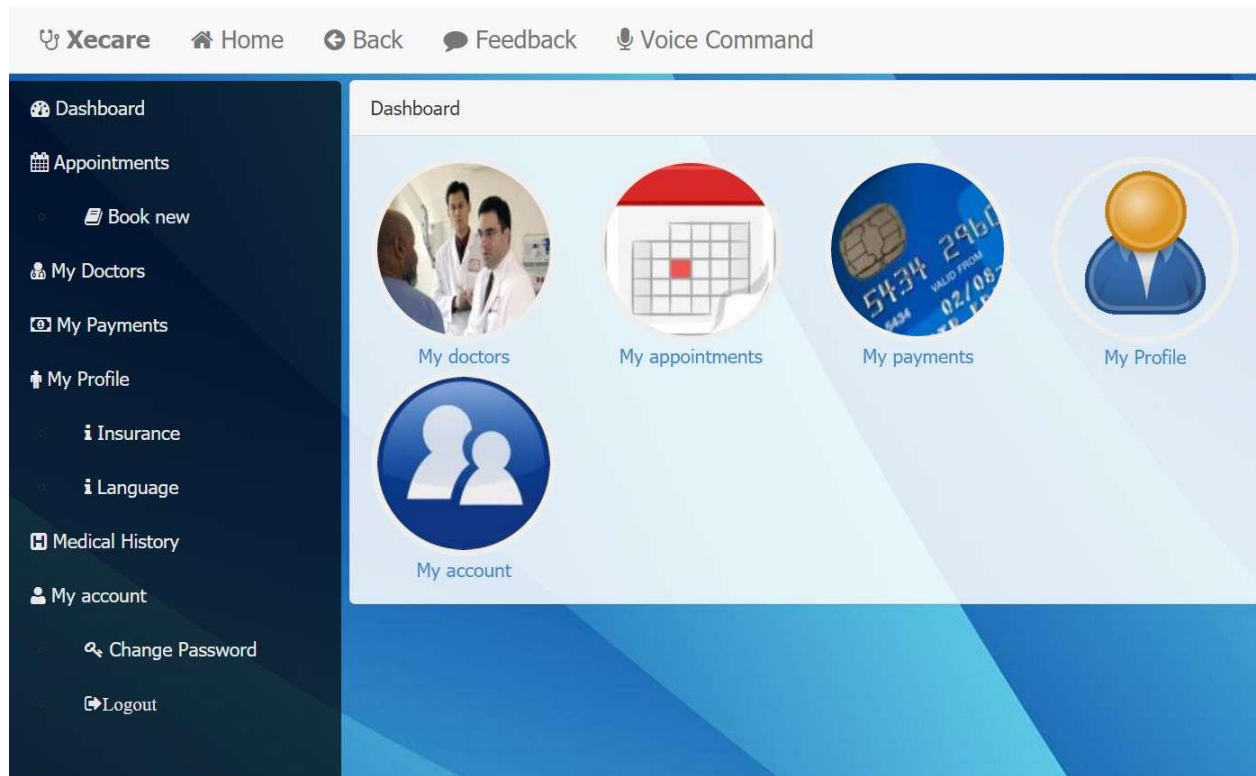


Figure 16. Screenshot of the patients' main menu

Dashboard: It provides access to the main menus in the system. Dashboard page may contain statistics or important information such as notifications, too. This is the first screen that user sees when he enters the system.

4.1.3 Manage list of doctors

Every patient can keep a list of doctors that he wants to contact more frequently. To manage the list of doctors, he should select “My Doctors” menu. The opened page has several action buttons to let the user add, find and delete doctors of his list. A patient cannot add a doctor who is not registered in the system. My Doctors page shows a list of doctors that is added to a patient’s account. Figure 17 depicts a view of this page. Each patient can add his own doctors and keep in touch with them. Patients can add a doctor by finding them by their phone number. If they mistakenly added a doctor, they can remove it using “Delete” button.

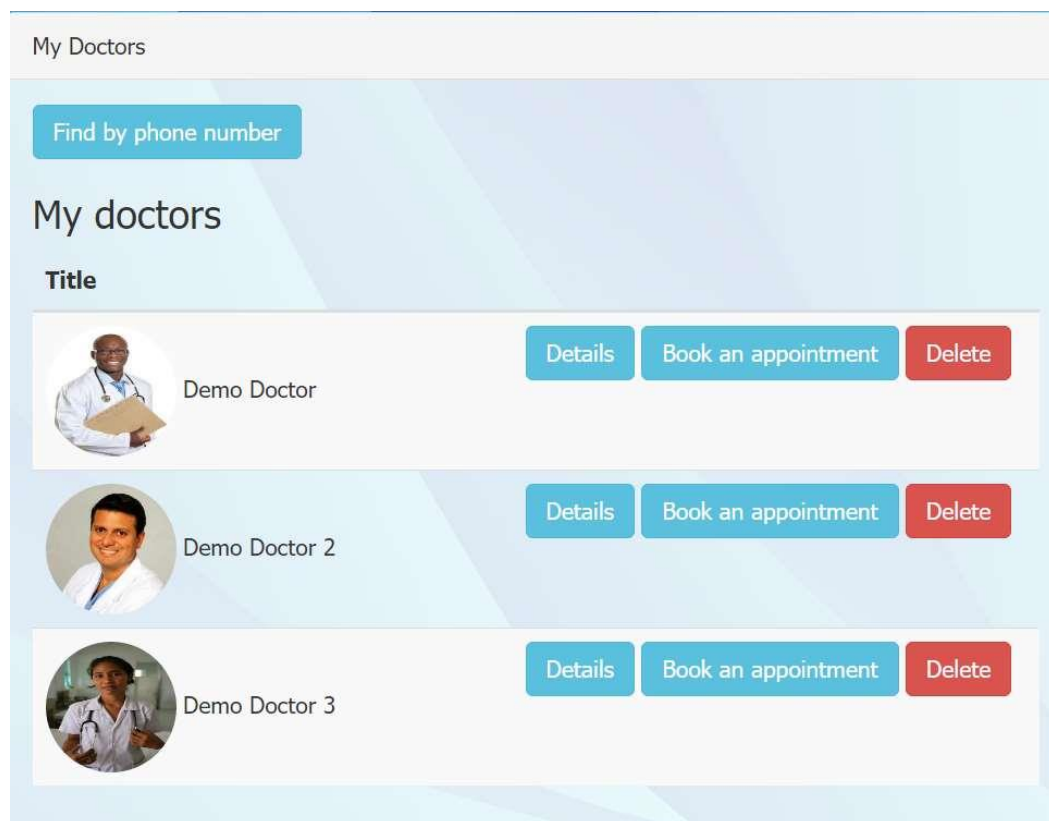
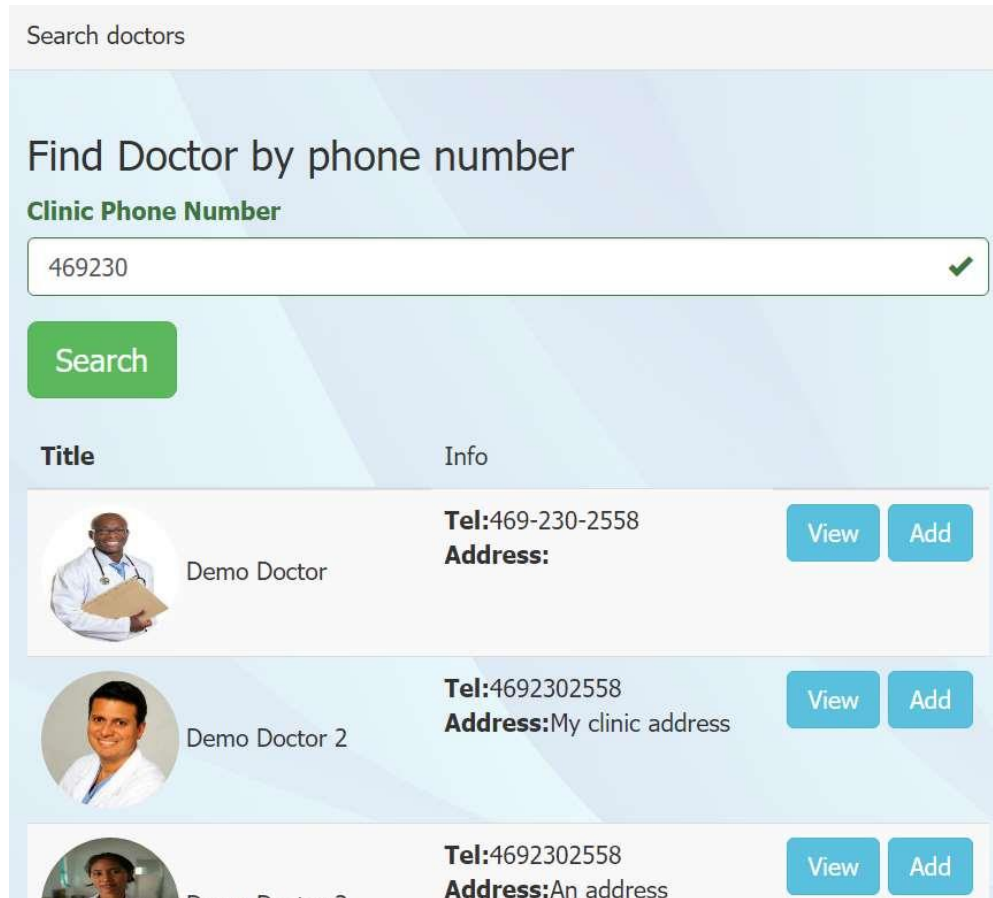


Figure 17. Screenshot of My Doctors page

The first step to manage the doctors list is to find the doctor using “Find by phone number” page.

This step is shown in Figure 18.






Title	Info
 Demo Doctor	Tel: 469-230-2558 Address: View Add
 Demo Doctor 2	Tel: 4692302558 Address: My clinic address View Add
 Demo Doctor 3	Tel: 4692302558 Address: An address View Add

Figure 18. Screenshot of Find doctor by phone number page

Then system shows a list of matched results when the user enters some parts of the clinic phone number. Using “View” button, the patient can view the doctor’s profile to ensure that the doctor is his doctor. Using “Add” button, the user can add the doctor to his list.

Doctors can also view their current patients using “My Patients” page. This page is very similar to “My Doctors” page (Figure 17). However, if the patient has not registered in the system, the doctor can still add him by providing his main contact information (Figure 23).

4.1.4 Define Doctor's Availability Calendar

Doctors need to define their availability calendar before booking an appointment. This can be done using “My Calendar” page (Figure 19). The user can select the date that she wants to define the availability, and click on the time to mark it as available. The available time will be marked as green in the calendar.

Schedule

2016/09/09

Prev Month Prev Week Prev Day Today Next Day Next Week Next Month

Delete All Copy range

12 AM to 7 AM

8 AM to noon

08 AM	08:00 AM	08:05 AM	08:10 AM	08:15 AM	08:20 AM	08:25 AM	08:30 AM	08:35 AM	08:40 AM	08:45 AM	08:50 AM	08:55 AM
09 AM	09:00 AM	09:05 AM	09:10 AM	09:15 AM	09:20 AM	09:25 AM	09:30 AM	09:35 AM	09:40 AM	09:45 AM	09:50 AM	09:55 AM
10 AM	10:00 AM	10:05 AM	10:10 AM	10:15 AM	10:20 AM	10:25 AM	10:30 AM	10:35 AM	10:40 AM	10:45 AM	10:50 AM	10:55 AM
11 AM	11:00 AM	11:05 AM	11:10 AM	11:15 AM	11:20 AM	11:25 AM	11:30 AM	11:35 AM	11:40 AM	11:45 AM	11:50 AM	11:55 AM
12 PM	12:00 PM	12:05 PM	12:10 PM	12:15 PM	12:20 PM	12:25 PM	12:30 PM	12:35 PM	12:40 PM	12:45 PM	12:50 PM	12:55 PM

Noon to 6 PM

7 PM to 23 PM

Figure 19. Screenshot of My Calendar page (Doctor's availability)

The system asks the doctor to define the type of availability and the number of the patients that she can visit in that time for the first time (Figure 20). It uses the same data to simplify the data entry process. This information can also be edited after the availability is defined. If the doctor has mistakenly marked a time as available, she can remove that time by clicking on the “Delete” button.

Schedule Slot

September 16 2016, 7:10 pm

Allowed number of patients

2 ✓

Mode of visit

Telemed ✓

Save Cancel

Figure 20. Screenshot of My Calendar page (Adding available time)

Since lots of available dates and times can be repeatable, we have implemented a copy function to copy the information. Figure 21 shows a view of “Copy Range” form. The doctor can select the date that he is going to copy and the number of days that needs to be copied. It can be one day or one week or a month.

Schedule

2016/09/09

Delete All Copy range

12 AM to 7 AM

8 AM to noon

08 AM 08:00 AM 08:05 AM

09 AM 09:00 AM 09:05 AM

10 AM 10:00 AM 10:05 AM

11 AM 11:00 AM 11:05 AM

Copy range

Copy from

2016/09/07 ✓

Range

One month

One week

Two weeks

One day

Two days

Three days

Four days

Five days

Figure 21. Screenshot of My Calendar page (Copy range form)

For example, if the doctor is only available on Monday and Wednesdays, and his available hours are similar on these days, he can follow this process: he can set the available times on one Monday

and use one day copy to copy Monday to Wednesday. Afterward, he can use the same function to copy the whole week to the next week and repeat to cover the whole month. He can continue using the same function to copy the month to the next month and build the calendar for the whole year. He can always customize his availability by selecting a particular date and modify the settings. It is worthy to mention that the framework keeps the dates in Unix Epoch milliseconds format for globalization. However, each user sees the date in his own time zone. The time zone configuration is based on the user's browser time.

4.1.5 Book an Appointment


In order to book an appointment, the doctor should be in the list of doctors. "Book an appointment" page is designed for this purpose. This page has four steps that are described the following:


The step one is to select the doctor. If the doctor is not in the list, then the patient can find her by her clinic phone number and add it to the list. Figure 22 shows the screen that a patient sees when he is going to book an appointment with his doctor.


Book new appointment

1 Select a doctor 2 Schedule 3 Reason 4 Summary

Title

 Demo Doctor

 Demo Doctor 3

 Demo Doctor 2

Not in the list,

Selected Doctor

Demo Doctor

Figure 22. Screenshot of Book an appointment page (step 1 – select a doctor)

Doctors can also register new patients if the patient is not registered in the system. Figure 23 shows this step. Registration can be done by entering the patient's name, family name and email address.

Register Patient

First Name

Last Name

Cell Phone Number


 (201) 555-0123

Figure 23. Screenshot of Register new patient page

If the person who is booking the appointment is the doctor, then this step will change to selecting the patient. Figure 24 depicts this step from doctors' panel. The system will inform the patient by

sending a notification email and asking him to complete the registration process later. Even though the registration is not complete, the doctor can book an appointment with the patient, and he will be notified.

The screenshot shows a web interface for booking a new appointment. At the top, the title "Book new appointment" is displayed. Below it, a progress bar indicates four steps: 1. Select a patient (active), 2. Schedule, 3. Reason, and 4. Summary. The main form area has a search bar and three input fields: "Phone Number", "First Name" (containing "Demo"), and "Last Name" (containing "Patient3"). Below these fields is a patient selection card featuring a circular profile picture of a woman, the text "Name: Demo Patient3" and "Tel: +146923025586", and a blue "Select" button. At the bottom left, there is a link "Can't find your patient?" and a blue "Add New Patient" button. Below that, a "Selected Patient" section shows "Demo Patient3" in a text box. A blue "Next" button is located at the bottom right.

Figure 24. Screenshot of Book an appointment page (step 1 – Select a patient)

The step two is to select the date and time of the appointment. This page shows the doctor's availability. Selecting a time will navigate the user to the next screen. Figure 25 depicts this step.

Book new appointment

1

2

3

4

Select a patient

Schedule

Reason

Summary

2016/09/07

Prev Month

Prev Week

Today

Next Week

Next Month

Date	Available times
September/07/2016, Wednesday	No time is available
September/08/2016, Thursday	8:15 am
September/09/2016, Friday	8:30 am 9:15 am 9:30 am 10:15 am 10:30 am 11:15 am 11:30 am
September/10/2016, Saturday	No time is available
September/11/2016, Sunday	No time is available
September/12/2016, Monday	No time is available
September/13/2016, Tuesday	No time is available

Selected Schedule

September 09 2016, 9:30 am

Next

Figure 25. Screenshot of Book an appointment page (step 2 – schedule)

The step three is to select the type of the appointment whether it is telemedicine or in-person visit and write the reason for the visit. The user can pick the reason from predefined reasons and can write his chief complaint and go to the next page. Figure 26 shows a screenshot of the application in this step.

Book new appointment

1 Select a patient 2 Schedule 3 Reason 4 Summary

Type

Telemed

Reason

Flu Shot

Chief Complaint

Next

Figure 26. Screenshot of Book an appointment page (step 3 – reason)

The step four is to let the user verify all the information. If the information is not correct, the user can go back to the previous steps and change it. This is the last confirmation before finalizing the appointment booking. Figure 27 depicts this step.

Book new appointment

1 Select a patient 2 Schedule 3 Reason 4 Summary

Please verify that the following information is correct, and finish the booking

Selected Patient

Demo Patient3

Selected Schedule

September 09 2016, 9:30 am

Chief Complaint:

Finish and Book!

Figure 27. Screenshot of Book an appointment page (step 4 – schedule)

When the user clicks on “Finish and Book!” button, the booking process is done. The patient will receive a notification email about the booked appointment, and the doctor will see the appointment in the list of the appointments.

4.1.6 Manage Appointments

In most of the scenarios, doctors want to check their appointments scheduled for today or one specific day and call their patients. “Today’s appointment page” that is shown in Figure 28 is designed for this purpose. In this page, the doctor can select a date (default is today’s date) and see the list of appointments of that date.

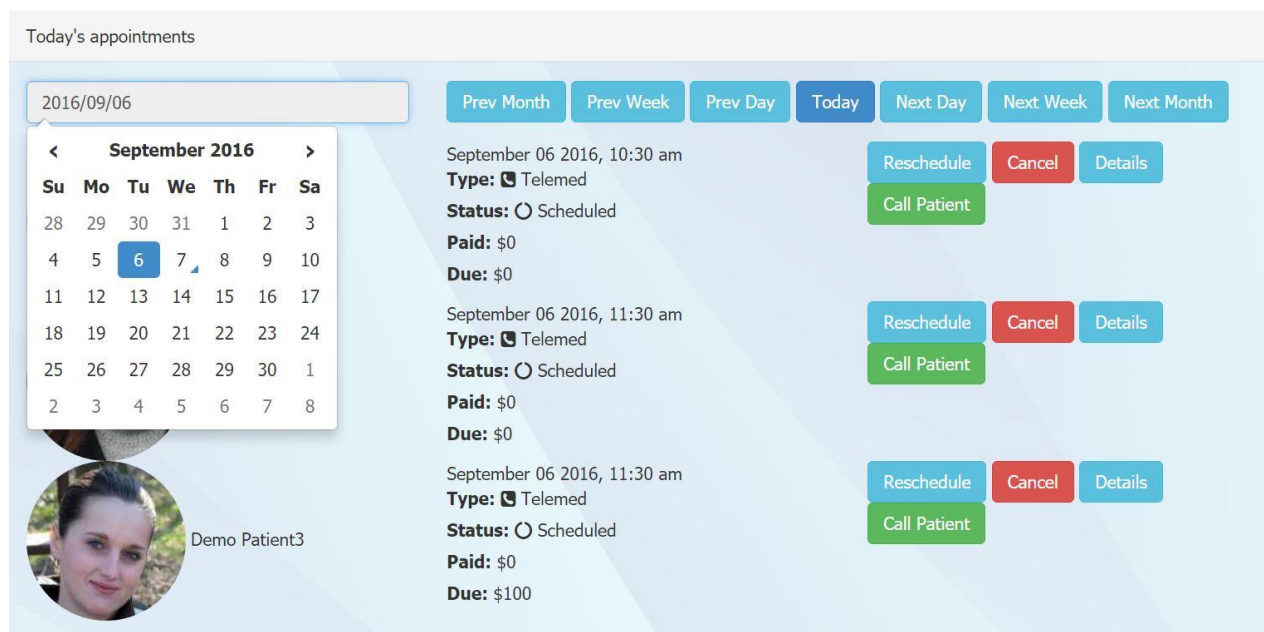


Figure 28. Screenshot of Today’s appointment page

The doctor can reschedule the appointment to another date using the “Reschedule” button. This opens “Reschedule” page that lets the user choose another date and time for the appointment from the doctor’s availability calendar. This page is shown in Figure 29.

Reschedule an appointment

Select a time for re-scheduling

2016/09/09

Prev Month Prev Week Today Next Week Next Month

Date	Available times
September/09/2016, Friday	9:20 am 9:40 am 10:20 am 10:40 am
September/10/2016, Saturday	No time is available
September/11/2016, Sunday	No time is available
September/12/2016, Monday	No time is available
September/13/2016, Tuesday	No time is available
September/14/2016, Wednesday	No time is available
September/15/2016, Thursday	No time is available

Patient

Demo Patient

Doctor

Demo Doctor

Reschedule from

September 06 2016, 10:20 am

Reschedule To

Re-Schedule

Figure 29. Screenshot of Reschedule page

The doctor can also cancel an appointment. When the appointment is canceled, it can't be rescheduled, and the patient will be informed about it by email.

The doctor can access more information and options about an appointment by going to its detail page. An appointment detail page provides several other screens that will help the doctor to manage the appointment by taking other actions or entering more information about it. Figure 30 shows a screenshot of this page. The doctor can also mark a visit as ended after the appointment is done by

clicking on “End Successfully.” The ended appointments will now be shown in Today’s appointment page so that the doctor can have a better management over visiting the patients.

The screenshot shows a web interface for appointment details. At the top, there's a header 'Visit information' and a row of tabs: 'Visit Info' (selected), 'Payments', 'Doctor Report', 'Doctor Review', and 'File Attachments'. Below the tabs, the form displays the following information:

Patient	Demo Patient 4
Doctor	Demo Doctor 3
Status	September 06 2016, 10:30 am
Status	Scheduled End Successfully
Type	Telemed
Paid	\$0
Due	\$0
Reason	
Chief Complaint	

At the bottom of the form, there are two buttons: 'Cancel Appointment' (red) and 'Reschedule' (blue).

Figure 30. Screenshot of Appointment details page (Visit Info tab)

This page also has several tabs such as Payments, Doctor Report, Doctor Review and File Attachments that are described in more detail in the following. When the doctor wants to keep a record of the visit, he can write it in the “Doctor Report” page. He can also write a prescription here. Figure 31 depicts a screenshot of this page.

The design of this page is simple because most of the tasks should be done in an EMR/EHR system. This screen can be used to show the required information synchronized from an EMR rather than to be used as a data entry form.

The screenshot shows a web interface for appointment details. At the top, there is a header bar labeled "Visit information". Below this, there is a horizontal tab bar with five tabs: "Visit Info", "Payments", "Doctor Report", "Doctor Review", and "File Attachments". The "Doctor Report" tab is currently selected and highlighted. Below the tabs, there are two main sections. The first section is titled "Doctor Report" and contains a large, empty text area for writing. The second section is titled "Prescription" and also contains a large, empty text area. At the bottom of the form, there is a green button labeled "Save".

Figure 31. Screenshot of Appointment details page (Doctor Report tab)

The doctor can also write more comments about the visit by selecting options in the Doctor Review page. The main purpose of this page is to keep a history of the finding that the doctor had after a visit. For example, she can select “Loss of feeling well-being” or “Eye pain/irritation” if she found out the patient had such issues. The options of this form are categorized based on the common body organs to simplify finding them. A view of this page is shown in Figure 32.

The screenshot shows the 'Doctor Review' tab of an appointment details page. At the top, there are five tabs: 'Visit Info', 'Payments', 'Doctor Report', 'Doctor Review' (which is active), and 'File Attachments'. Below the tabs, the 'General' section is visible. It has a header 'General' and two radio buttons: 'Applicable' (selected) and 'Not Applicable'. Below this, there are four checkboxes: 'Significant weight loss' (unchecked), 'Loss of feeling of well-being' (checked), 'Fatigue or loss of energy' (unchecked), and 'Difficulty sleeping' (unchecked). A text input field labeled 'Comment' is below the checkboxes. The 'Eyes' section follows, with a header 'Eyes' and two radio buttons: 'Applicable' (selected) and 'Not Applicable'. Below this, there are five checkboxes: 'Blurred vision' (unchecked), 'Double Vision' (unchecked), 'Spots in front of your eyes' (unchecked), 'Eye pain/irritation' (checked), and 'Need for corrective lenses' (unchecked). A green 'Save' button is positioned over the bottom of the 'Eyes' section. A partial 'Comment' input field is visible at the very bottom.

Figure 32. Screenshot of Appointment details page (Doctor Review tab)

File attachments tab helps the doctors to update a file to an appointment. Since the form is very similar to the Patient's Documents page, we didn't put its screenshot here. A view of this is shown in Figure 44.

4.1.7 Manage Medical History

Doctors usually need to see patients' chart or history of their medical records. We called this Medical History. Each patient can access the forms of medical history through his panel and fill them out once before booking an appointment with his doctors.

Note: We obtained these forms from a hard copy of actual hospital forms. As mentioned before, the design considered in these forms may lack several necessary components. These forms are customizable based on the research application and it also can be used to contain data provided in an EMR/EHR system rather than manual data entry. In addition, these forms can be filled out by both patients (if you are going to visit the doctor remotely) and doctors (if it is an in-person visit). So, the editing option is available for both of them.

Medical history page has several tabs: Allergy, Condition, Medications, Family History, Immunization, Past Medications, Social behavior and signature. Figure 33 shows these tabs and Allergy page. For each allergy item, the patient can type a title for the allergy and the type of reaction that he has to it. Patients can select their relevant conditions in the conditions tab.

The screenshot displays the 'Medical History' page with the 'Allergy' tab selected. The page features a header with tabs for Allergy, Condition, Medications, Family History, Immunization, Past Medications, Social, and Sign. Below the tabs, the 'Allergy' section contains two columns: 'Allergy' and 'Type of reaction'. The 'Allergy' column has five input fields with the following text: 'milk', 'soy', 'peanuts', 'Allergy', and 'Allergy'. The 'Type of reaction' column has five input fields with the following text: 'shortness of breath', 'itchy rash', 'itchy rash', 'Type of reaction', and 'Type of reaction'. A green 'Save' button is located at the bottom center of the form.

Figure 33. Screenshot of Patients' Medical History page (Allergy tab)

All common medical conditions are listed on this page and categorized to simplify the access. Figure 34 has a view of this tab. The patient can find the appropriate conditions and marked them and save the form.

The screenshot shows the 'Medical History' page with the 'Condition' tab selected. The page is divided into three main sections: Allergies, Heart, and Rheumatoid Arthritis. Each section contains a list of conditions with checkboxes. The 'Allergies' section includes: Anaphylaxis (Severe Allergic Reaction), Chronic Rhinitis, Cold / Flu / Allergies, Food Allergy (checked), Hives, Latex Allergy, and Sinusitis. The 'Heart' section includes: Angina, Congenital Heart Disease, Coronary Angiogram, Coronary Angioplasty, Coronary Artery Bypass, Heart Attack, Heart Murmurs, Heart Palpitations, High Cholesterol, and Stroke. The 'Rheumatoid Arthritis' section includes: Arthroscopy, Celebrex, Cortisone Injection, Remicade, Rheumatoid Arthritis, Total Hip Replacement, and Total Knee Replacement. At the bottom, there is a 'Senior Health' section with a checkbox for 'Alzheimer's Disease'. A green 'Save' button is located at the bottom center.

Figure 34. Screenshot of Patients' Medical History page (Condition tab)

Medications tab is designed to let the patients enter their current medications. It has two fields: drug name and dosage. The immunization and past medications forms are also very similar to this form so we didn't put their screenshots. Figure 35 shows a screenshot of this page.

The screenshot shows the 'Medical History' page with the 'Medications' tab selected. The page is divided into two main sections: Drug and Dose. Each section contains a list of input fields. The 'Drug' section includes: Acetaminophen, Drug Name, Drug Name, Drug Name, and Drug Name. The 'Dose' section includes: 2 tablet / day, Dose, Dose, Dose, and Dose. A green 'Save' button is located at the bottom center.

Figure 35. Screenshot of Patients' Medical History page (Medications tab)

Family History tab contains information about the medical conditions of the family of the patient. It has information about the serious illnesses of the father, mother or siblings. The user can also enter the age when the condition is detected. Figure 36 shows a view of this form.

The screenshot shows a web application interface for a patient's medical history. At the top, there's a header 'Medical History' and a navigation bar with tabs: Allergy, Condition, Medications, Family History (selected), Immunization, Past Medications, Social, and Sign. Below the tabs, the form is organized into columns: Relationship, Alive/Dead status, Age (Or age at death), and Serious illness(es). The form lists four family members: Father, Mother, Sister1, and Sister2. For each member, there are radio buttons for 'Alive' (selected for Father and Mother) and 'Dead'. The 'Age' field contains '45' for Father, '40' for Mother, and a placeholder 'Age' for Sister1. The 'Serious illness(es)' field contains 'high blood pressure' for Father, 'dibetes' for Mother, and a placeholder 'Serious illness(es)' for Sister1. A green 'Save' button is at the bottom.

Relationship	Alive/Dead	Age (Or age at death)	Serious illness(es)
Father	<input checked="" type="radio"/> Alive <input type="radio"/> Dead	45	high blood pressure
Mother	<input checked="" type="radio"/> Alive <input type="radio"/> Dead	40	dibetes
Sister1	<input type="radio"/> Alive <input type="radio"/> Dead	Age	Serious illness(es)
Sister2	<input type="radio"/> Alive <input type="radio"/> Dead		Serious illness(es)

Save

Figure 36. Screenshot of Patients' Medical History page (Family History tab)

Patients' social behaviors are captured in a form of a questionnaire from the patients. It asks some questions about their smoking, drinking, exercising habits and the doctor can interpret his patients' conditions based on that. Figure 37 shows a view of this form.

This form is interactive and its components can change based on the answers received from the user. For example, when the user says "Yes" to "Do you drink caffeine?" question, the system asks "how many cups per day?" question, and asks the user enter more information. This simplifies the answering process by asking the patients to answer the relevant questions only.

Medical History

Allergy Condition Medications Family History Immunization Past Medications Social Sign

1. Do you currently smoke or chew tobacco? ☐ Yes ☒ No

have you in the past? ☐ Yes ☒ No

2. Do you sometimes drink alcohol, beer or wine? ☐ Yes ☒ No

3. Do you drink caffeine? ☒ Yes ☐ No

how many cups per day?

2 cups

4. Do you wear a helmet when riding a bike? ☐ Yes ☐ No

Save

Figure 37. Screenshot of Patients' Medical History page (Social tab)

Patients need to sign the medical history form when they completed all sections. This signature lets the doctor know that the patient has provided all necessary information. It can also be used to save the whole data in the version history. Figure 38 is a view of this form.

Medical History

Allergy Condition Medications Family History Immunization Past Medications Social Sign

By signing below, I hereby certify that to the best of my knowledge all the information I have furnished on this form is complete, true, and accurate. Patient initials:

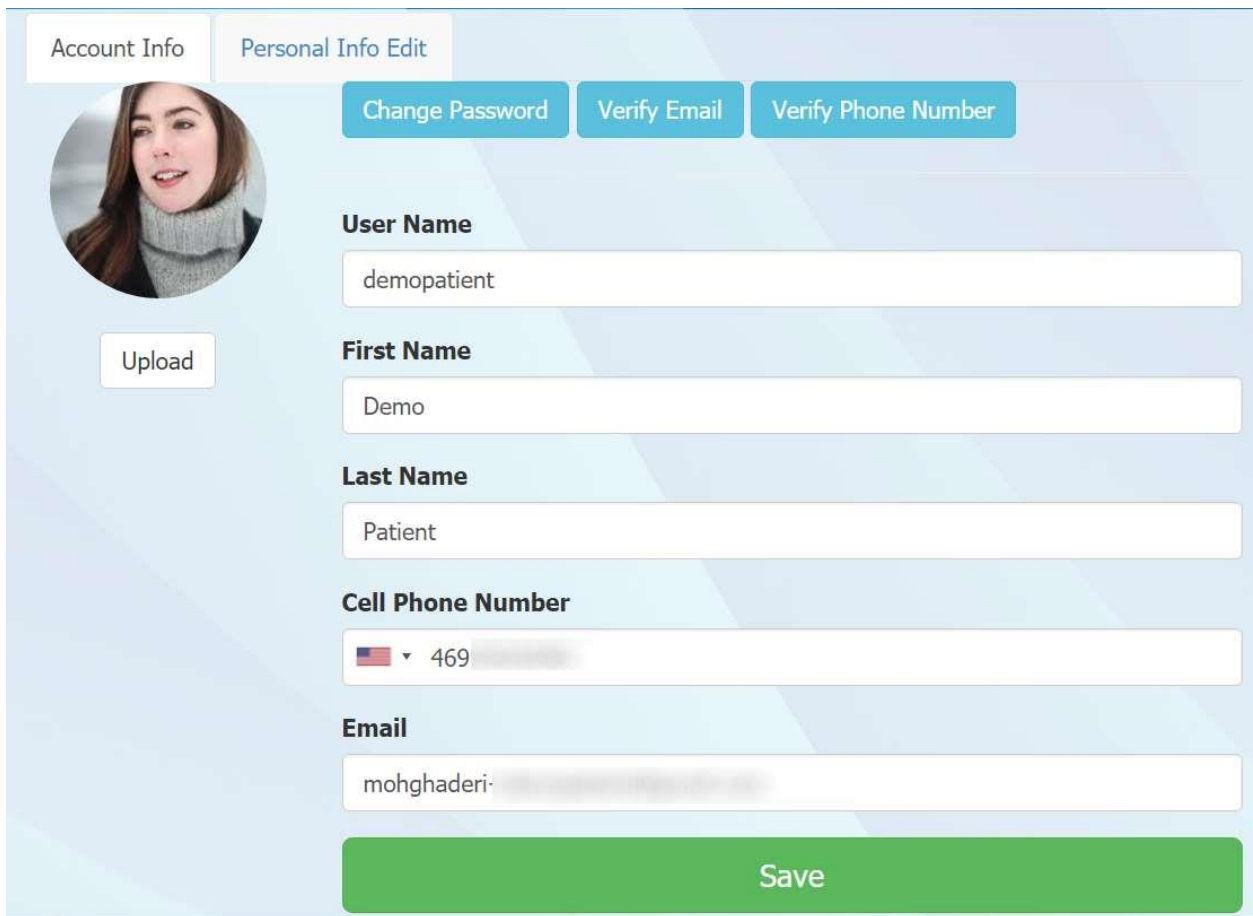
Demo Patient

Sign

Figure 38. Screenshot of Patients' Medical History page (Signature tab)

4.1.8 Manage Account and Profile

Each user has its own account and profile. The account keeps the information about the user such as first name, last name, phone number and email. The user can also choose a profile picture. The form for managing the account is showed in Figure 39.



The screenshot displays a web interface for managing a user's account. At the top, there are two tabs: "Account Info" (selected) and "Personal Info Edit". Below the tabs, on the left, is a circular profile picture of a woman with a grey turtleneck sweater, with an "Upload" button underneath. To the right of the profile picture are three buttons: "Change Password", "Verify Email", and "Verify Phone Number". The main form area contains several input fields with labels: "User Name" (containing "demopatient"), "First Name" (containing "Demo"), "Last Name" (containing "Patient"), "Cell Phone Number" (with a dropdown menu showing a US flag and "469" followed by a masked number), and "Email" (containing "mohghaderi-" followed by a masked email address). A large green "Save" button is positioned at the bottom right of the form.

Figure 39. Screenshot of Account Info page

Patients should enter their demographic data before booking an appointment. This can be done using Personal Info Edit page by filling out the information such as birthday, gender, country and address. This page is shown in Figure 40.

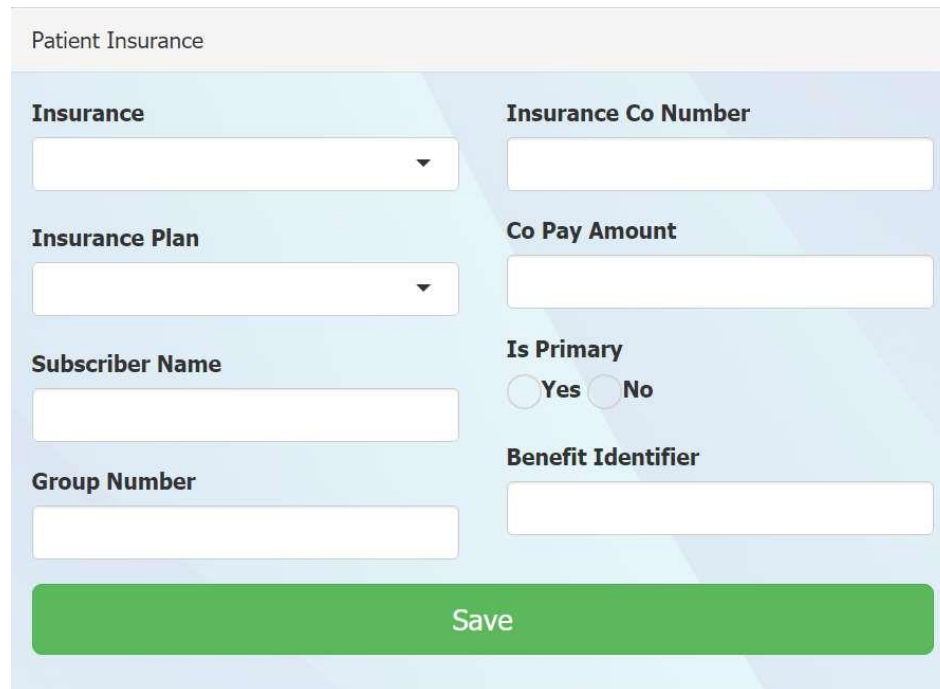
The screenshot shows a web form titled "Personal Info Edit" with a tab labeled "Personal Info". The form contains the following fields:

- Birthday**: A text input field with a green checkmark.
- Gender Type**: A dropdown menu showing "Male" with a downward arrow.
- Country**: A dropdown menu showing "United States" with a downward arrow.
- State**: A dropdown menu showing "Texas" with a downward arrow.
- Address Line1**: A text input field containing "18383 Gallery dr" with a green checkmark.
- Address Line2**: A text input field with a green checkmark.
- City**: A text input field containing "Dallas" with a green checkmark.
- Zip Code**: A text input field containing "75252" with a green checkmark.

A large green button labeled "Save" is positioned at the bottom of the form.

Figure 40. Screenshot of Personal Info Edit page

Patients can also enter their insurance information. This information will be visible to their doctors when they are going to request a payment. Insurance plan, subscriber name, group number, insurance co-number, co-pay, beneficiary identifier and whether this is the primary insurance account define the insurance information for a patient. Figure 41 and Figure 42 show the screenshots of the related data entry forms.

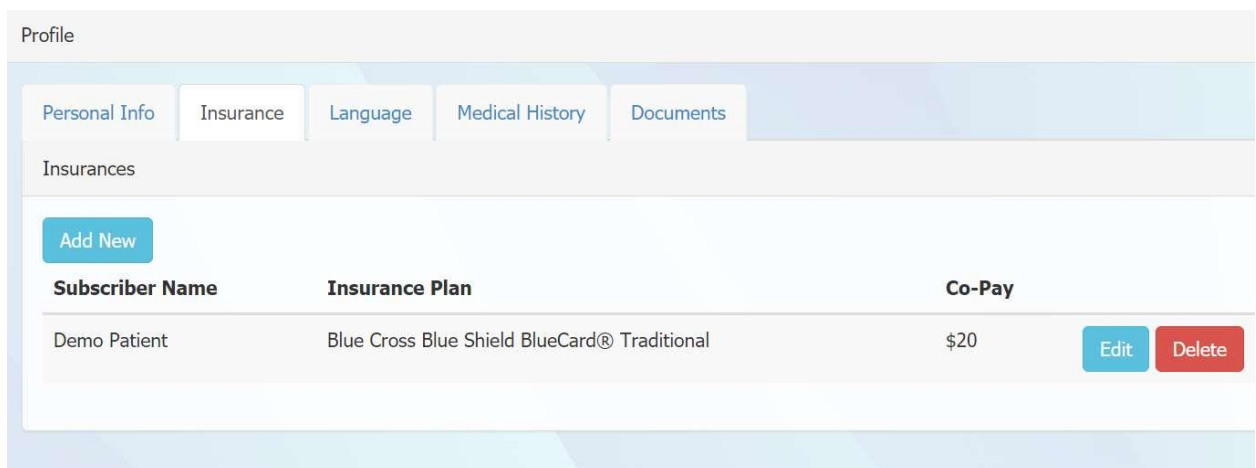


Patient Insurance

Insurance <input type="text"/>	Insurance Co Number <input type="text"/>
Insurance Plan <input type="text"/>	Co Pay Amount <input type="text"/>
Subscriber Name <input type="text"/>	Is Primary <input type="radio"/> Yes <input type="radio"/> No
Group Number <input type="text"/>	Benefit Identifier <input type="text"/>

Save

Figure 41. Screenshot of Patient Insurance page



Profile

Personal Info Insurance Language Medical History Documents

Insurances

Add New

Subscriber Name	Insurance Plan	Co-Pay	
Demo Patient	Blue Cross Blue Shield BlueCard® Traditional	\$20	<input type="button" value="Edit"/> <input type="button" value="Delete"/>

Figure 42. Screenshot of Profile page (Insurance tab)

Both doctors and patients can define their languages. We have designed this part for supporting the globalization of the platform. This information may be useful to simplify the process of searching doctors or patients. A complete list of languages is provided in the platform, too. Figure 43 shows the language selection page.

The screenshot shows a web interface for a 'Profile' page. At the top, there's a header 'Profile' and a navigation bar with tabs: 'Personal Info', 'Insurance', 'Language' (which is active), 'Medical History', and 'Documents'. Below the tabs, the section is titled 'Languages'. The main content area has a heading 'Please select languages that you are familiar with'. Under this, there's a label 'Language' followed by a dropdown menu. Below the dropdown is a green 'Add to list' button. Further down, there's a section titled 'Selected Items'. This section contains a table with two rows: 'Persian' and 'English'. Each row has a red 'Delete' button next to it.

Selected Items	
Title	
Persian	Delete
English	Delete

Figure 43. Screenshot of Profile page (Languages tab)

Patients can also upload their medication documents to their profile. We have used Amazon Simple Storage Server (S3) to store the files. All files will be uploaded directly to S3 server. All files are stored using the built-in encrypted feature and are not publicly accessible. Before accessing a file by a doctor or a patient, they should request a download link. This request checks whether the user is allowed to access the file or not. If the user is permitted to download the file, the system generates a temporary download link that is valid only for thirty minutes. In other words, files are kept in a secure environment to be HIPPA compliant. The upload process is done using Ajax and it shows a progress bar to provide a better user experience. Figure 44 shows a screenshot of this tab.

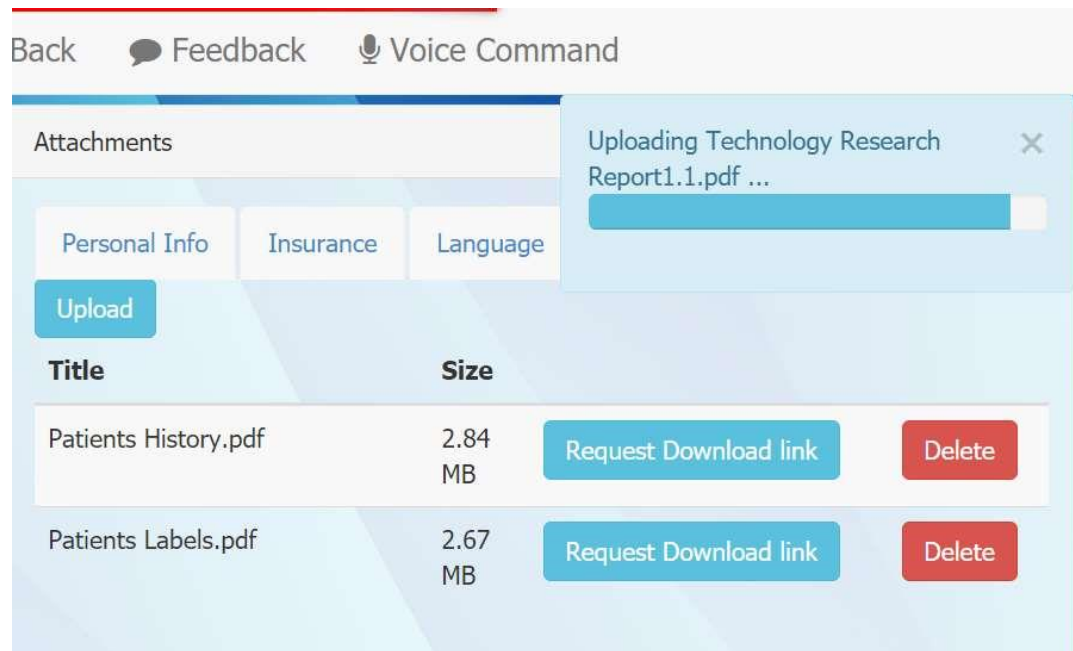


Figure 44. Screenshot of Profile page (Documents tab)

Doctors don't need to fill out their demographic information. They will need to enter their practice information so that the patients can find them. In addition, the system can have an administrative process to verify doctors' information such as their practice certificate. As our design has the assumption of being used as a research platform, we haven't implemented certificate verification process. Doctors' can enter their practice and license information such as degree, residency place, medical school, clinic phone number, registration number, license numbers, board certification year, clinic address, military experience, hospital privilege and personal statement in this part. This page is shown in Figure 45.

Doctor and Clinic information

Degrees <input type="text" value="MD"/> ✓	Registration Number <input type="text" value="111111111"/> ✓	Has Revoked License <input type="radio"/> Yes <input checked="" type="radio"/> No ✓
Residency Place <input type="text" value="UTSW"/> ✓	License Numbers <input type="text" value="111111111"/> ✓	Has Military Experience <input type="radio"/> Yes <input checked="" type="radio"/> No ✓
Medical School <input type="text" value="UTSW"/> ✓	Board Certification Year <input type="text" value="1987"/> ✓	Hospital Privileges <input checked="" type="radio"/> Yes <input type="radio"/> No ✓
Clinic Phone Number <input type="text" value="469-230-2558"/> ✓	Clinic Address <input type="text"/>	Federal D E A Information <input type="text"/>

Professional Statement

✓

Save

Figure 45. Screenshot of Doctor and Clinic Information page

As each state of the United States has different laws about medical practices, doctors need to get practice certificates from them separately. Doctors can keep a list of states that they are permitted practice in our system. The related page for defining this list is shown in Figure 46. Patients can see this information as a part of doctors' profiles. This information can also be used later to match patients and doctors and give appropriate warnings when the doctor is not allowed to practice in a state where the patient resides.

Certified States

Please select states that you are allowed to visit patients in the United States

U S State

Add to list

Selected Items

Title
Texas

Delete

Figure 46. Screenshot of Certified States page

Doctors also can specify the type of insurances they accept from their patients. The patients can see this information as a part of their doctor's profile. Selecting an insurance plan is as easy as choosing the company and available plans. The related page is shown in Figure 47. Our system doesn't have a complete list of all insurance plans, but the data can be modified in the database.

Insurance Companies and plans

Please select insurance plans that you accept from patients

Insurances

- American Family Insurance
- American Specialty Health
- American Republic Insurance
- American Medical Security (AMS)
- American Dental Professional Services
- Universal American
- United American Insurance Company
- Loyal American Life Insurance Company

Add to list

Health Partners of Kansas

Add to list

HealthEOS by Multiplan

Add to list

Acceptable Plans

Title
Blue Cross Blue Shield - BlueCard® Traditional
Blue Cross Blue Shield - BlueCard® PPO/EPO

Delete

Delete

Figure 47. Screenshot of Doctors' Acceptable Insurance Plans page

4.1.9 Manage Payments

Online payment is a part of telemedicine practices. We have used PayPal API to provide this feature. In addition, we have implemented an example of a use case scenario. The first step for the doctor to receive payments is to create a PayPal account. Afterward, she can enter her PayPal email using “Payment Info” page in the system (Figure 48). Doctors can ask for the payment from the Payment tab of an appointment details page (Figure 49).

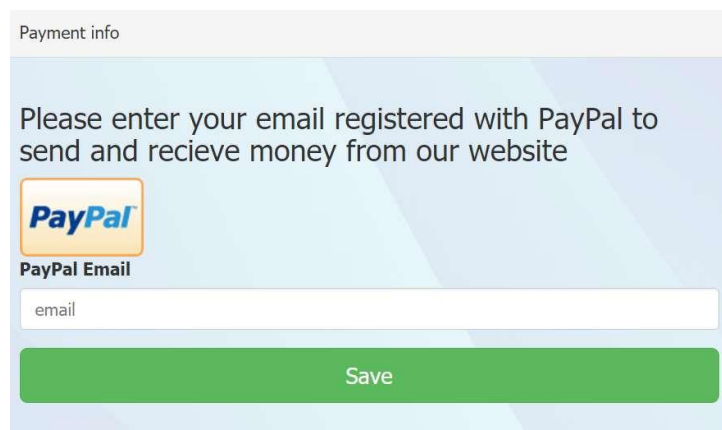
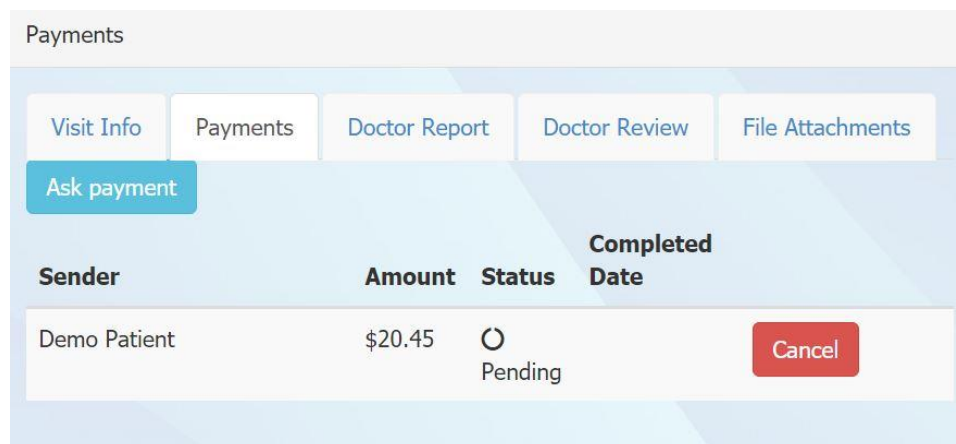


Figure 48. Screenshot of Doctors’ Payment Info page



Sender	Amount	Status	Completed Date
Demo Patient	\$20.45	Pending	

Figure 49. Screenshot of Visit Details page (payments tab)

As it was mentioned before, the Details page is accessible from “My Appointments” or “Today’s Appointments” pages. “Ask for payment” button redirects the doctor to the “Ask for Payment” page (Figure 50).

Insert new payment

Enter the amount that patient should pay

Amount

\$ 20.45

Service Charge Amount

\$ 0.2045


This is an extra money that we charge for processing the online payments

Save

Figure 50. Screenshot of Doctors’ Ask for Payment page

After specifying the amount of the money that the patients need to pay, it will be added to the payments list of the doctors (Figure 51) and patients (Figure 52).

Received payments

Sender	Amount	Status	Completed Date
 Demo Patient3	\$100	<input type="radio"/> Pending	

Cancel

Figure 51. Screenshot of Payments list (Doctor’s view)

The patient can use Pay button to pay the requested fees. The system redirects the patient to PayPal website to do the process. The design of the payment feature is flexible enough to easily be replaced by other payment services.



Payments			
Receiver	Amount	Status	Completed Date
 Demo Doctor 3	\$100	 Pending	Pay

Figure 52. Screenshot of Payments page (Patient's view)

It is unlikely that a research project requires payment, but the integration with other web services is very important. As the processing of online transactions is an important part of telemedicine applications, and it usually needs to be done using other services such as PayPal or Stripe, we have developed this to show how the system can be integrated with other web services.

4.1.10 Visit (Video Communication)

One of the most important use cases of the system is online visiting. Patients shall be able to book an appointment with their doctors and visit them online. In addition, doctors can book an appointment with their patients. Doctors can find their appointments in “Today's Appointments” page (Figure 28) and call their patients. The system shows a waiting screen to the doctor (Figure 53) sends a message to the user using the real-time communication channel (SignalR).

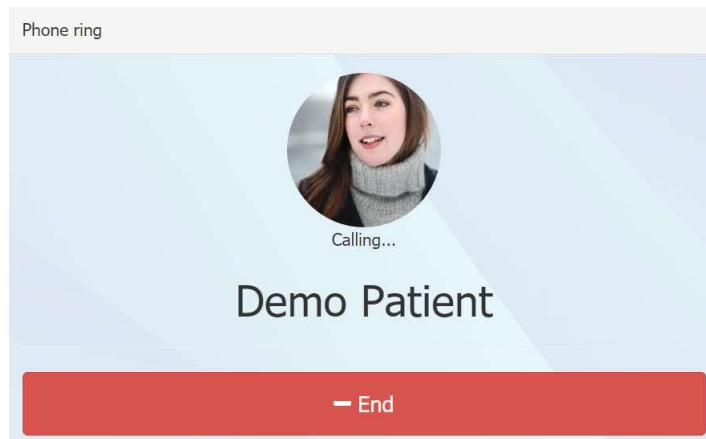


Figure 53. Screenshot of Calling Patient page

If the patient is online then the system informs him that the doctor is calling (Figure 54). If he was not online, then the system sends a push notification message to his mobile device. This message will be shown in the notification bar of the patient's smartphone. The patient's Android or iOS devices need to be registered with his email to receive notifications. By clicking on the message, the operating system will open the mobile application and the patient can answer the call. If the patient couldn't answer the call, the system will notify him about the missed call, and the doctor can call him again later.

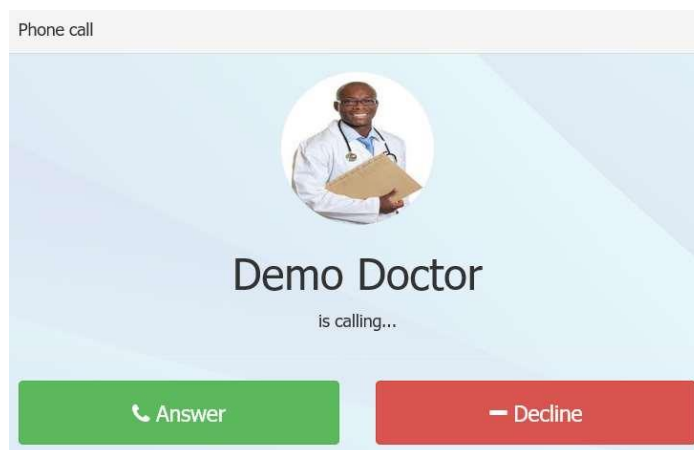


Figure 54. Screenshot of Received Call page

The doctor and the patient will be redirected to “Video Communication” page (Figure 55) when the patient answers the call.



Figure 55. Screenshot of Video Communication page

We have used WebRTC to enable video communication. WebRTC uses a fast peer-to-peer connection, and the video quality is good. Since WebRTC is still in its experimental phases, we have implemented an Adobe AIR flash client that can be used for more serious use cases. Video communication part is implemented in an Iframe so that it can be easily replaced by any new technology.

4.1.11 General system features

During a research, the user may face issues or have some ideas to improve the user experience. Therefore, we have implemented a feature in the system to receive feedbacks from the users. A

feedback can be a new idea or a bug report. This helps to reduce the gap between the system developers and the users. Feedback page is shown in Figure 56.

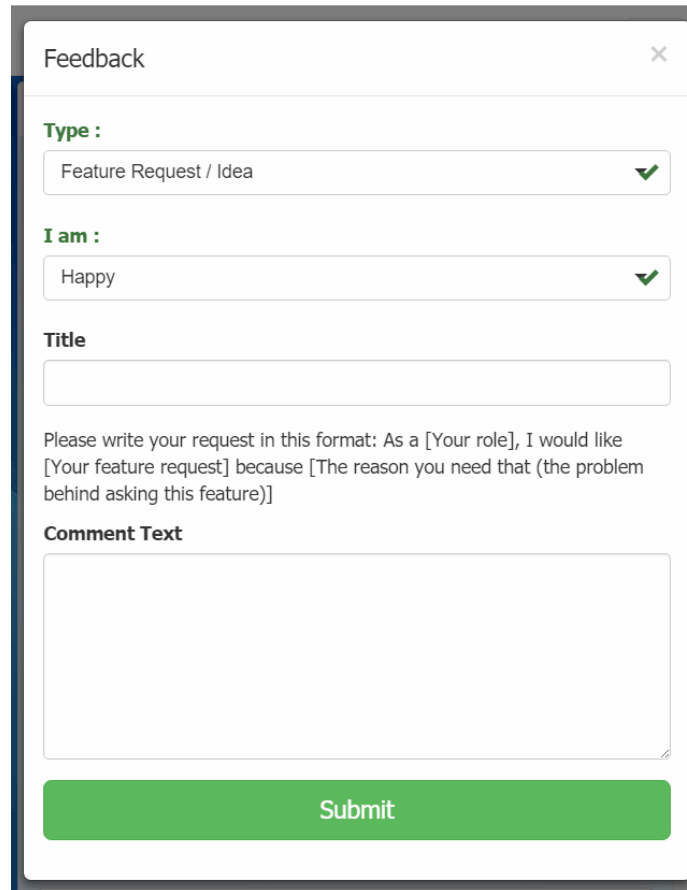
The image shows a web form titled "Feedback" with a close button (X) in the top right corner. The form contains several fields: a "Type" dropdown menu with "Feature Request / Idea" selected and a green checkmark; an "I am :" dropdown menu with "Happy" selected and a green checkmark; a "Title" text input field; a paragraph of instructional text: "Please write your request in this format: As a [Your role], I would like [Your feature request] because [The reason you need that (the problem behind asking this feature)]"; a "Comment Text" text area; and a green "Submit" button at the bottom.

Figure 56. Screenshot of Feedback page

To simplify the process of data entry, the system has a voice recognition feature to convert the user's speech to text. This is very helpful when the user is going to write several paragraphs. The current implementation is based on Web Speech API [71], and Google Chrome and Firefox 49+ are the only browsers that support it now. In order to use this feature in mobile applications, the developers should use native features of the operating system or online APIs such as iSpeech (ispeech.org), Google Cloud Speech API (<https://cloud.google.com/speech/>) or IBM Watson

(<https://www.ibm.com/watson/developercloud/speech-to-text.html>). Figure 57 shows an example of using the voice recognition feature in Google Chrome for writing a report.

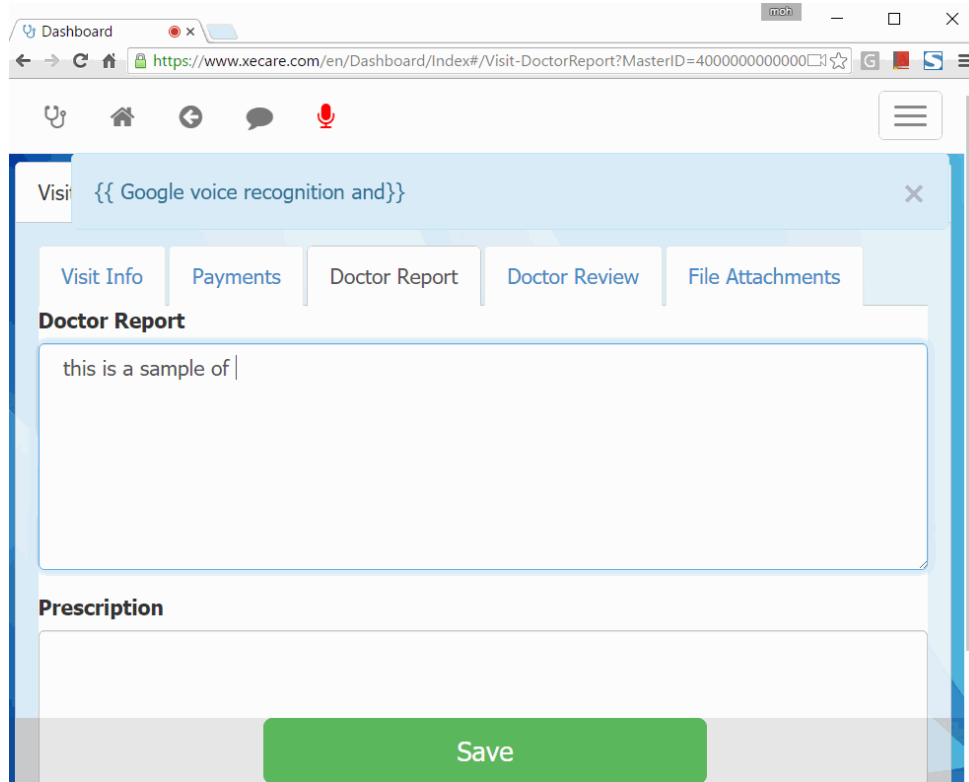


Figure 57. Screenshot of Voice Recognition feature

4.2 Augment Reality Module

Since web frameworks are limited to do performance-intensive processing, it is necessary to have native code in some research works. For example, real-time processing of the image data requires very fast code that runs natively on the platform. Transferring the images (video) over the server is probably not a good choice in many scenarios as it puts a heavy transfer load on the channel and processing load on the server, so the processing should be done on the client-side. As our web framework is completely API based, it is easy to be integrated with these native applications. It is

noteworthy to mention that this module is a very small piece of the work in comparison to the whole proposed platform.

The Berkeley University is working on a native Android framework for telemonitoring [13]. Thus, we are not going to duplicate their work. However, we would like to show more how a native Android application can have advanced features such as image processing that is not possible in web frameworks. We implement an Augmented Reality application to show breast tumors of on the patient's breast using Java and C++ (Android NDK - <https://developer.android.com/ndk/index.html>). Since Android, iOS, and almost any other native programming framework support web views, the implemented code here can be used along with the web application that we have made. In addition, the required data can be transferred through an encrypted channel to the API server.

4.3 Augmented Reality Process

The process has four steps depicted in Figure 58. The overall system takes the video from the scene and it produces overlay images on top of real video by rendering the prepared 3D model. This 3D model needs to be rendered from the same direction that camera looks at the scene. For this purpose, a marker is attached to the patient's body at a certain point. Afterward, the system detects the marker in the video sequence and estimates the camera direction and renders the 3D model from a correct angle with the proper scale on top of real video for producing AR. Each of these modules is explained briefly as the following:

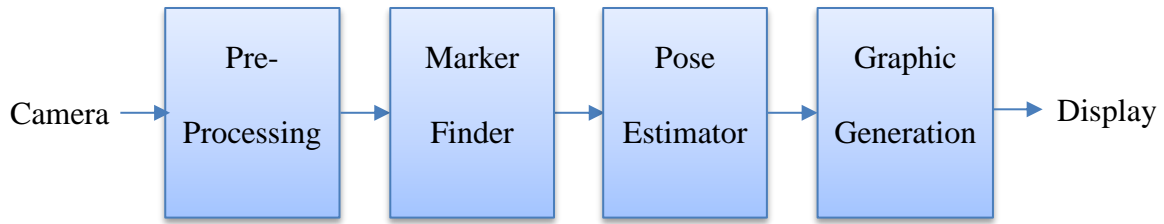


Figure 58. Overall architecture of the augmented reality module

4.3.1 Pre-Processing

The camera's image is the input to this module and it produces a binary image as output. The first step is converting the input image to a grayscale image (I_g). The next step is converting the inverted grayscale image to a binary image (I_b). This can be done by comparing values of pixels with a threshold:

$$I_b(x, y) = \begin{cases} 0 & I_g(x, y) \geq T \\ 1 & \text{otherwise} \end{cases} \quad (1)$$

where T is a threshold. For getting an informative binary image, the value of T needs to be chosen in an optimum way. In this work, Otsu algorithm [72] is used for finding the optimal threshold. In Otsu algorithm, the process of making binary image is considered as an automatic clustering task with two classes. Then, the algorithm finds the threshold by finding the optimum for classifying two classes using Fisher discriminant analysis (FDA). FDA maximizes the ratio of intra-class variance to inter-class variance.

4.3.2 Marker Finder

In marker based AR applications, a marker (fiducial) is added to the image for estimating the relative position of the camera to the scene and registering the 3D model according to the scene.

A marker is a predefined object in the scene which can be identified easily. Usually, markers are asymmetric rectangular black and white object which are added to an object in the scene. Since they are black and white; they can achieve the maximum contrast in an image. Figure 59 shows two of such markers. We picked the U pattern as a representative of the up direction as it is easier to remember, and the other pattern to test our system with a more complex pattern.

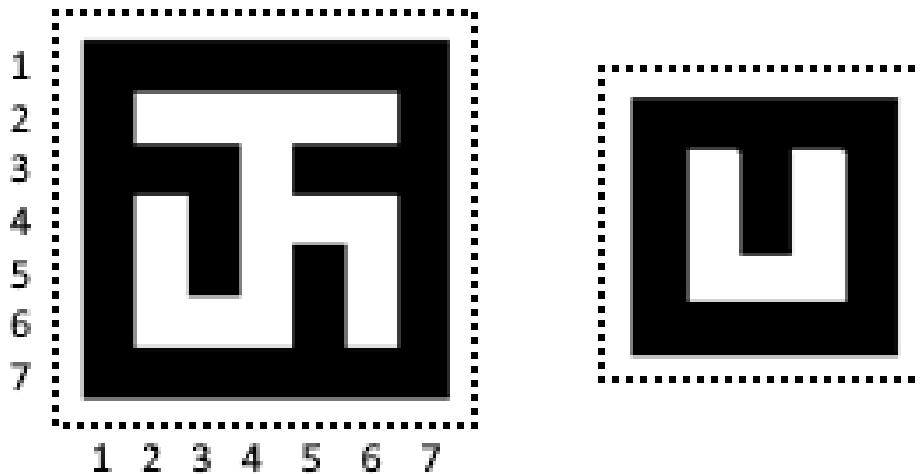


Figure 59. Two examples of markers used in this work. Left marker is a grid of 7x7 and the right one is 5x5. Dotted lines are the cut lines and are not parts of the markers

Markers are defined as squares because it is easier to define squares with polygons and find them. The size of marker whether it is 5x5 or 7x7 or 4x4 doesn't make any difference. The important point is that the selected pattern should not be symmetric so that the algorithm can find the rotation of the marker. Each marker has a blank square around it, and there is a pattern inside of the black square to avoid a mismatch between the marker and objects inside doctors' office. Since the algorithm works with black and white pictures, it is important to keep a white border around the marker when making the marker to make sure that it works well on women with dark skins.

Since the marker can appear in the image from different angles, it can be considered as a polygon with four vertices. So, polygon finding algorithm is used for finding the marker in the binary image. The first step in finding markers is finding the all closed contours in the binary image. This is done by topological structural analysis algorithm proposed in [73]. This algorithm finds the region between outer borders of inner borders of holes. After finding polygons, there is a list of candidate polygons for a marker.

A polygon should have the following conditions to be considered as a candidate marker: (i) have exactly four vertices, (ii) be convex, and (iii) the distance between consecutive points should not be very small. In addition, from two candidates that are close to each other, the one with smaller perimeter should be removed as we never have two very close or overlapping markers in this application. In short, this module gets all potential polygons from the previous module as an input and removes anything that doesn't have the above conditions.

For detecting the marker from candidates list, there are two steps: (i) obtain the frontal view of the rectangle by removing the perspective projection, and (ii) search for the marker pattern in the frontal view of the polygon. For detecting the pattern a template matching is done for recognizing the marker. Figure 60 shows this process.

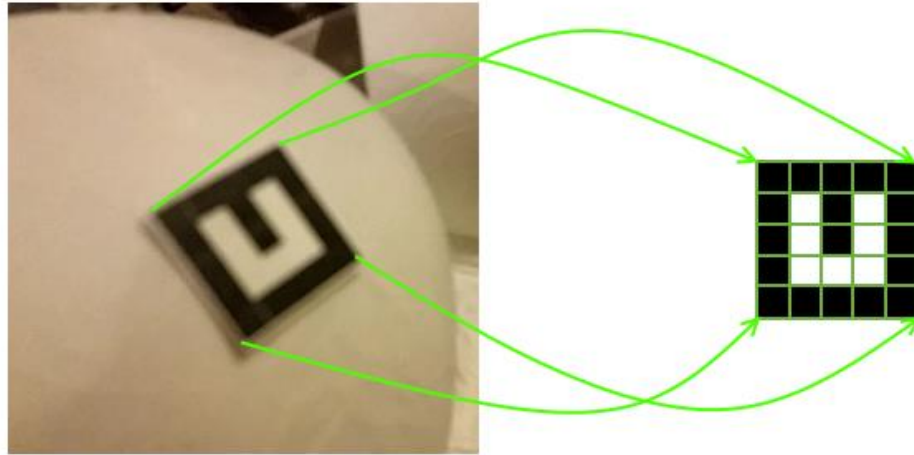


Figure 60. Frontal view of a marker. Left picture obtained from a camera and the right is the frontal view. Corners of a potential marker in the camera picture is mapped to the frontal view to process detection in the next step.

The pattern of each marker is represented by a matrix of 1 (for white and 0 (for blacks). For example, matrix M is representative of our U marker. $M = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}$. Matrix M is 3x3 and not 5x5

because we don't keep the border as we know it is always black.

To do the pattern match, the number of black pixels and white pixels are counted in each square of the black and white picture. If the number of blacks is more than whites, then the cell is considered as black and vice versa. The obtained matrix is compared with matrix M , and if it matches, then it has found a match. All four rotations (up, down, right, left) of the pattern are considered in matching a pattern.

4.3.3 Position Estimator

The last step is estimating the angle that camera is looking at the scene. By extracting this information, the 3D model can be rendered in the correct angle. This problem includes estimating 3D information using 2D information and usually referred as PnP problem [74]. For arranging the

PnP equations, the homogenous coordinate system is used. For this purpose, the homography transform needs to be estimated.

It is preferred to work with the homogenous coordinate system instead of Cartesian coordinate system when estimating the homography transform. Suppose $q_c = [x, y, z]^t$ denotes position of a point in 3D Cartesian system; then, its homogenous equivalent is obtained as $q_h = [x, y, z, 1]^t$. Conversely, the corresponding Cartesian representation of a point in homogenous system like $q_h = [x, y, z, \lambda]^t$ is obtained as $\left[\frac{x}{\lambda}, \frac{y}{\lambda}, \frac{z}{\lambda}\right]^t$.

Using the homography transform the problem can be written as the following matrix form:

$$p = CHq \quad (2)$$

where p is the homogeneous coordinate of a point in 2D and q is its corresponding homogeneous coordinate in 3D. The matrix $C_{3 \times 4}$ is the camera calibration matrix used to compensate the barrel distortion added by camera's lens and the matrix $H_{4 \times 4}$ is the homography transform. The camera calibration is out of the scope of this paper [47]. The matrix H is unknown but has the following form:

$$H = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & T_1 \\ R_{21} & R_{22} & R_{23} & T_2 \\ R_{31} & R_{32} & R_{33} & T_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

where R and T represent rotation and translation, respectively. For estimating the elements of H at least 4 pairs of corresponding points in 2D and 3D are needed. We use the position of four corners of markers for this purpose.

4.3.4 Graphic Generation

This module renders a pre-stored 3D model and adds texture to the model for displaying. It uses the extracted pose from the pose estimation module to render the 3D model in the correct angle and distance to the camera. This model can be created by experts using a combination of breast modeling methods or simply by obtaining it from 3D devices. Figure 61 shows the 3D model of a typical tumor used in this study. After rendering the 3D model, it is alpha blended to the real video to create the overlay image. This image is shown to the doctor on the screen of an Android device, a smartphone or on a head mounted display like Google Glass [46].

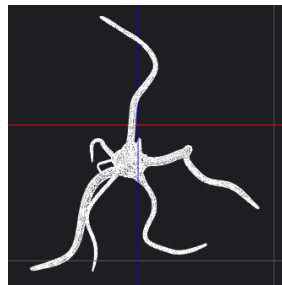


Figure 61. The 3D model of a tumor used to be shown on the breast

4.4 Chapter Summary

We propose a web-based platform that covers many common features of a healthcare research problem. This base can be customized to develop these research applications. This chapter provides an overview of the most of the platform functional features. We have shown and described these features by providing screenshots of the base platform application. We also introduced a design for more advanced modules by giving an example of an Augmented Reality module. Representative applications will be discussed in the next chapter and quality (non-functional) features will be explained in more detail in chapter five.

CHAPTER 5

REPRESENTATIVE APPLICATIONS

5.1 Breast Cancer Application

In this section, we explain how the platform can be used to create a new application. The main usage of this application is the risk assessment of the breast cancer. The potential usage of the platform is the breast cancer early detection and engagement with doctors. The following groups of the patients are going to be users of this application:

1. Women older than 20 who are not diagnosed with Breast Cancer
2. Women older than 40 who are not diagnosed with Breast Cancer
3. Women who are diagnosed with Breast Cancer

The first group can access system to get information about the breast cancer and be engaged in doing breast self-examination. The second group can use the system in the same way as the first group; in addition, they can do the risk assessment and schedule a mammography with their doctors. The third group can do everything that the second group may do. They can also use it for post-treatment reporting. For example, a woman who had a surgery or receives chemotherapy may enter her daily information in the system so that her doctor can be informed about her health conditions during and after treatment.

Breast self-examination

Breast self-examination is considered as one of the most important ways of detecting breast cancer. It is recommended that every woman should start it after age 20 onwards every month. The woman

may not know how to perform this examination correctly. This problem can be addressed by providing adequate training movies.

Risk Assessment

Women older than 40 should do the risk assessment with their doctors. Doctors are the best sources to decide about the patient's risk of developing breast cancer. Thus, we need to provide a risk assessment questionnaire for the patients to fill out so that their doctors can access this information and take further actions. If any action is necessary, doctors should schedule appointments with their patients in this system and visit them online.

5.1.1 Customizing the base

In this section, we explain how the existing features in the framework can be utilized to create this application. Figure 62 shows the customized version of the main menu for our breast cancer app.

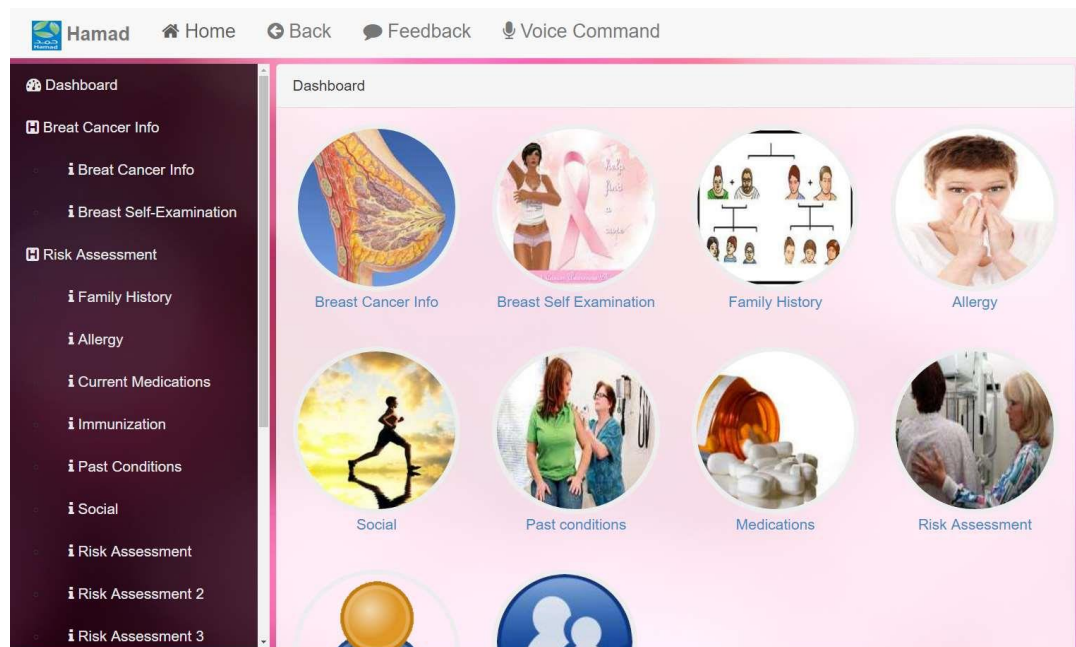


Figure 62. Screenshot of Patient's Menu in Breast Cancer Representative Application

It is clear from the requirements that many use cases are similar to what we have described in the base application. So, we can customize the base and reuse many components. Customizing the base is as simple as adding a file to the “Site” folder of the framework. The framework tries to find the file in this folder first. If it couldn’t find the file, then it searches for in the base platform. Therefore, overriding base files is as simple as having the same file name in the “Site” folder.

The following code shows the code in PatientMenu.cshtml file that is used to design the dashboard menu of this application:

```
@{ ViewBag.ViewTitle = "Dashboard"; }
@Html.Partial("~/Views/Dashboard/_Layout/_MasterViewHeader.cshtml")

@helper MenuItem(string caption, string link, string ngClick, string thumbFileName)
{
    <div class='col-sm-3 col-xs-4 col-md-3 col-lg-2'>
        <div class="text-center" style="height:210px">
            <a class="" href="@link" data-ng-click="@ngClick">
                
                <div class=''>@caption</div>
            </a>
        </div>
    </div> <!-- col-6 / end -->
}

@functions {
    private static string CurrentUser(string path)
    {
        return "OpenPath(' + path + "', {MasterID: CurrentUserID})";
    }
}
```

```

<div class="row" ng-controller="DashboardController">

    @MenuItem("Breast Cancer Info", "#BreastCancerInfo-BcInfo", null,
"breastlearning.jpg")

    @MenuItem("Breast Self Examination", "#BreastCancerInfo-BseTraining", null,
"bse.jpg")

    @MenuItem("Family History", null, CurrentUser("BreastCancer-PatientLabel"),
"familyhistory.jpg")

    @MenuItem("Allergy", null, CurrentUser("BreastCancer-Allergy"), "allergy.jpg")
    @MenuItem("Social", null, CurrentUser("BreastCancer-Social"), "social.jpg")
    @MenuItem("Past conditions", null, CurrentUser("BreastCancer-PasMed"),
"pastconditions.jpg")

    @MenuItem("Medications", null, CurrentUser("BreastCancer-CurrentMed"),
"medications.jpg")

    @MenuItem("Risk Assessment", null, CurrentUser("BreastCancer-RiskAssessment"),
"riskassessment.jpg")

    @MenuItem("My Profile", "#Person-ProfileView", null, "Profile.png")
    @MenuItem("My Account", "#User-ProfileEdit", null, "MyAccount.jpg")

</div> <!-- row / end -->

@Html.Partial("~/Views/Dashboard/_Layout/_MasterViewFooter.cshtml")

```

This file is located in “{root}\Sites\hamad\Views\Dashboard\” folder. As you can see in the code, everything is customizable in the user interface. Similarly, to create the sidebar with new page links, all the developers need to do is to create a file and call it “_MainSideBar.cshtml.”

5.1.2 Education module

An important aspect of the application is to provide some information about the breast cancer according to the requirements. Since the framework is web-based, the developer can use all advantages of the web for content creation and management. We don’t need to have a sophisticated

educational application. The content can be created using any web editor, and it can be added to the application inside an iframe or as a content of a web view. Figure 63 shows how a YouTube video is embedded in the page to help patients learn about the Breast Self-Examination (BSE).

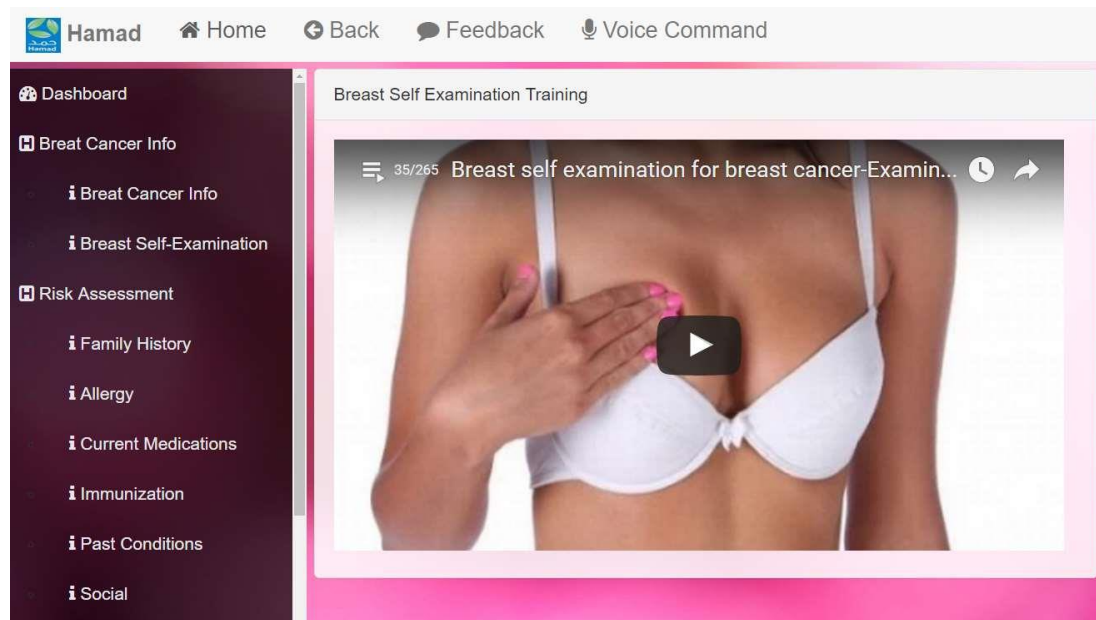


Figure 63. Screenshot of BSE Training in Breast Cancer Representative Application

The content of this page is defined as follows:

```
@{ ViewBag.ViewTitle = "Breast Self-Examination Training"; }
@Html.Partial("~/Views/Dashboard/_Layout/_MasterViewHeader.cshtml")

<div class="embed-responsive embed-responsive-16by9">
  <iframe class="embed-responsive-item"
    src="https://www.youtube.com/embed/xpbcXKhk8S0"
    frameborder="0"
    allowfullscreen
    onload="initView();"
    style="width:98%; height:98%">
  </iframe>
</div>
@Html.Partial("~/Views/Dashboard/_Layout/_MasterViewFooter.cshtml")
```

It is not necessary to put the content in an iframe. Since all files are actual HTML files, the content can be any HTML page. Thus, the developer has enough flexibility to use content type.

5.1.3 Reusing Medical History module for Risk Assessment

Potential patients need to register in the system and fill out their medical history forms and questionnaire related to the breast cancer assessment. The questions are obtained from the actual forms that are being used in Hamad Medical Corporation of Qatar. Since these questionnaires are very similar to the Medical History module of the framework, the developer can re-use these modules to create more forms. In this example, the developer only defines the front-end and links it to Medial History controllers. Figure 64 shows the created user interface for entering the risk assessment data.

Hamad Home Back Feedback Voice Command

Risk Assessment (1) Risk Assessment (2) Risk Assessment (3) Sign

1) Personal History of breast cancer? ☐ Yes ☒ No

2) Family history of breast cancer? ☐ Yes ☒ No

3) Others ☐ Yes ☒ No

Risk factor for breast cancer
 Age at your first period years

Number of pregnancies: Age at First pregnancy: Number of live births: Vaginal: C-Section:

Number of abortions/miscarriages: ☐ Yes ☐ No

Save

Figure 64. Screenshot of Risk questionnaires in Breast Cancer Representative Application

Creating this form is as easy as defining an AngularJS controller that inherits all functions from the Medical History base class. The following lines need be added to the JavaScript code.

```
entityControllerFactory.create("MedicalHistory", "BCRisk", function ($scope, sParams)
{
    BaseControllers.MedicalHistoryBaseAddFormServices($scope, sParams, "BCRisk");
});
```

Afterward, we need to set the AngularJS controller in the view. The following code shows how ng-controller tag is set to our newly defined controller:

```
<form role="form"
    ng-controller="MedicalHistoryBCRiskController"
    id="MedicalHistoryBCRiskFrm"
    onsubmit="return false;">
...

```

As the medical history module saves the JSON object defined in the AngularJS model, it is not necessary to develop more code on the client-side. If special validation is necessary before saving the form, the developer can override the \$scope.save function in the defined controller. There can be a model checker on the server-side code to check the validity of the model if necessary.

It is noteworthy to say that current version of JavaScript is not an Object-Oriented programming language and does not support inheritance. Thus, we are using functions to mimic the inheritance behavior. The new version of JavaScript (ECMA 1.6) and TypeScript have this feature, but they were not available when we started development of this project.

5.1.4 Re-using the base modules

Since the rest of the application is very similar to the base telemedicine features, all the developer needs to do is to put the links in the sidebar panel. For example, the following code in `_MainSideBar.cshtml` file will provide links to the views of profile management of the base.

```

</li>
    @MenuItem("My Profile", "fa-male", "Person-ProfileView")
    <ul>
        <li>@MenuCurrentUser("Insurance", "fa-info", "PatientInsurance-
GridInsideProfileTab")</li>
        <li>@MenuCurrentUser("Language", "fa-info", "User_Language-
TwoKeyInsideProfileTab")</li>
    </ul>
</li>

```

As mentioned before, the framework tries to find those views in the Site folder and since they do not exist there, it uses the base views of the platform. Therefore, the new app can get any functionality of the base code without extra coding.

5.2 Post-Surgical Care Application

In this section, we will explain some parts of another representative application. Once a patient is diagnosed with breast cancer there are multiple ways to treat the disease based on the patient's conditions such as surgery, radiation therapy, chemotherapy, hormone therapy and targeted therapy. It is recommended that patients track their health conditions and keep doctors informed about it when receiving treatments. For these patients, a special portal is needed so that they can access and enter the information about their health such as mood, intaken calories by foods, burnt calories and sleep hours. Doctors should be able to see this information and visit their patients if it is necessary.

5.2.1 Implementation

The implementation of this application is very similar to the Breast Cancer representative application. A view of the Patient's Menu of this app is shown in Figure 65.

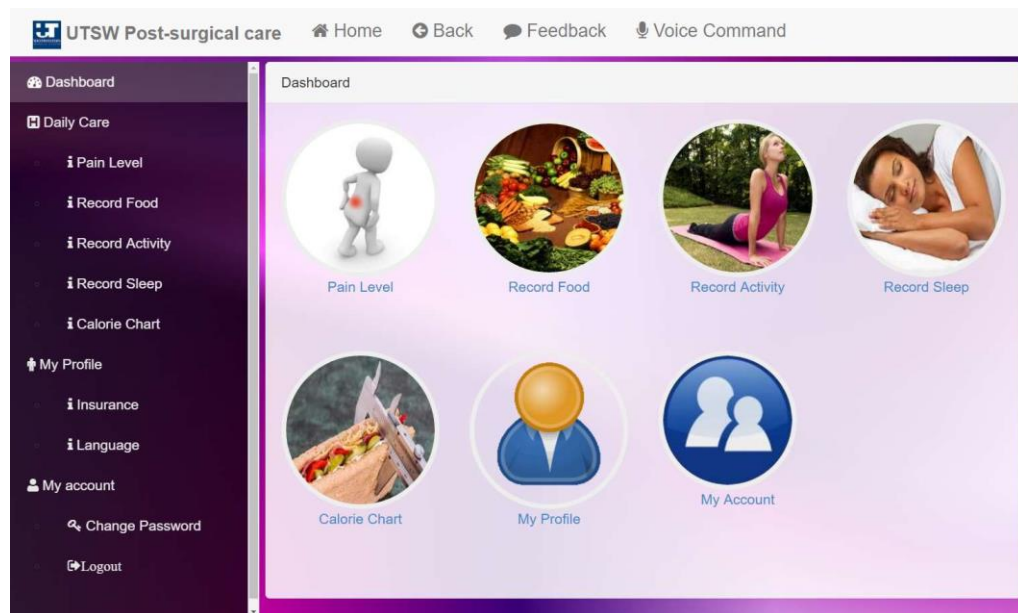


Figure 65. Screenshot of Patient's Menu in Post-Surgical Care Application

We start customizing the base platform by modifying the main menu. This part explained in more detail in the previous example; therefore, we skip the explanation here.

5.2.2 Pain Level Form

Pain Level Form lets the user record his mood. The doctor can see the records of the mood changes during the time period of the study. Pain Level is one single value related to the user's health condition. We have called these kinds of values Vital Values. The framework provides the backend (server-side) for recording and saving these values, and it is very easy to reuse it.

The first step is to define the UI controller in JavaScript files. The following code should be written in the related JS file:

```
entityControllerFactory.create("VitalValue", "PainLevelForm", function ($scope,
sParams) {
    BaseControllers.AddEditFormServices($scope, sParams);
    //http://wbotelhos.com/raty
```

```

$scope.dataValueRatyOptions = {
    path: "../../Content/Images/smiley/",
    starHalf: 'mood0.png',
    starOff: 'mood0.png',
    starOn: 'mood6.png',
    number: 10,
    target: '#precision-hint',
    targetKeep: true,
    hints: ['1', '2', '3', '4', '5'],
    size: 60
};

$scope.onBeforeInsert = function () {
    $scope.model.PersonID = $scope.CurrentUserID;
    $scope.model.VitalTypeID = 4; //pain level
    $scope.model.IsManualEntry = true;
}

});

```

In this example, we are going to use a JavaScript component called Raty to show how the framework user interface can be integrated with AngularJS component. Adding this component is as easy as including its files in the web page. After including the code of the directive to the base JavaScript files, we can use it in our UI controller. Since we are going to use an AngularJS directive to access this component, the coding of the UI is very clean and simple. We define the options in our controller and reference it to the web page. Statement “\$scope.dataValueRatyOptions” keeps the required configuration for this component. Finally, we set the parameters in the model that is not shown on the user interface.

Statement “BaseControllers.AddEditFormServices(\$scope, sParams)” provides an inheritance-like functionality to eliminate coding many similar tasks in the edit forms. This is the framework’s

feature that can help the developers to write less code for their user interface. A view of this form is shown in Figure 66.

Figure 66. Screenshot of Pain Level Form

The next step is to define the HTML part of the UI. Thus, we add PainLevelForm.cshtml file to the VitalValue folder.

```
@{
    var entityName = "VitalValue";
}
@{ ViewBag.ViewTitle = "Pain Level"; }
@Html.Partial("~/Views/Dashboard/_Layout/_MasterViewHeader.cshtml")
<form role="form"
    ng-controller="@{(entityName)}PainLevelFormController"
    id="@{(entityName)}PainLevelFormFrm"
    onsubmit="return false;">
    <div class="row">
        <div class="col-md-12">
            <h3>Please select your pain level from 1-10</h3>
            <div class="hidden-sm hidden-xs">
                <div data-ng-rtty="dataValueRatyOptions"
                    ng-model="model.DataValue" style="width:500px"></div>
            </div>
        </div>
    </div>
```

```

        <select id="precision-hint" class="input hint" ng-
model="model.DataValue">
            <option value="">--</option>
            <option value="1">1</option>
            ... // Code removed for readability
            <option value="10">10</option>
        </select>
    </div>
</div>
<hr />
<div class="form-group">
    <button class="btn btn-lg btn-success btn-block"
ng-click="Save($event)">
        Save
    </button>
</div>
</form>
@Html.Partial("~/Views/Dashboard/_Layout/_MasterViewFooter.cshtml")

```

After assigning the AngularJS controller name in the form tag (`ng-controller="VitalValuePainLevelFormController"`), it binds the data model defined in the JS file to the UI.

To provide a better user experience on small screens, we can simply make this component invisible by adding “hidden-sm hidden-xs” to its CSS class. Please note that Raty component and our HTML Select box gets and sets the same value (i.e. “`ng-model="model.DataValue"`”).

5.2.3 Pain Level Chart

Visualization of data is usually provides a better interpretation of the data. Therefore, we would like to show a chart of the entered pain level values to the patient or his doctor. There are many JavaScript chart components that can be used to draw diagrams, and we are not interested in developing one from scratch. Therefore, we use HighCharts.com HighStock product to draw our

diagram. This is an open-source component that is free to use for research purposes. As there is no official AngularJS directive for this component, we are going to use JQuery to create this component. Figure 67 illustrates how the pain level data can be shown in a time-period.

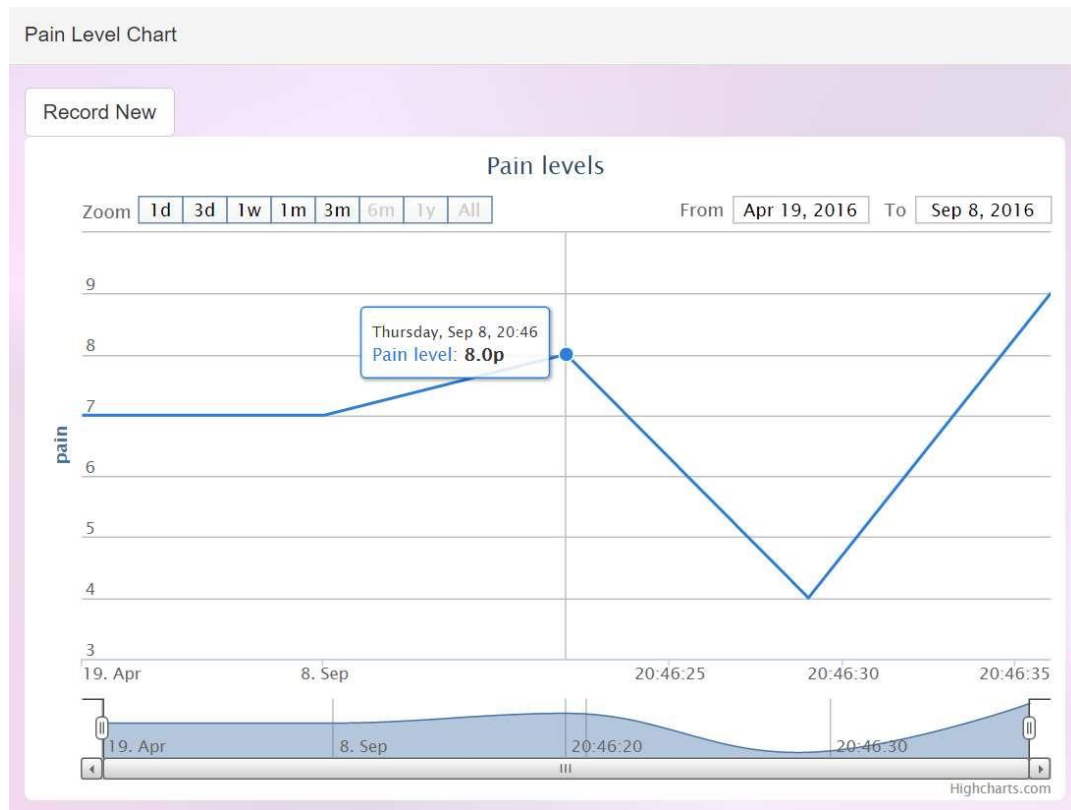


Figure 67. Screenshot of Pain Level Chart

Similar to the other forms, we need to program the UI controller first. The following code is all we need to write in the related JavaScript file:

```
entityControllerFactory.create("VitalValue", "PainLevelChart", function ($scope,
sParams) {
    if ($scope.MasterID)
        $scope.UserID = $scope.MasterID;
    else
        $scope.UserID = $scope.CurrentUserID;
    $scope.getBaseChartConfig = function(data) {
        return chartConfig = {
```

```

        chart: {
            zoomType: 'x'
        },
        rangeSelector: {
            ... // Code Removed for readability
        },
        yAxis: {
            title: {
                text: 'pain'
            }
        },
        title: {
            text: 'Pain levels'
        },
        series: [{
            name: 'Pain level',
            data: data,
            tooltip: {
                valueDecimals: 1,
                valueSuffix: 'p'
            }
        }]
    };
} // end getBaseChartConfig

$scope.init = function () {
    $scope.ExecInline("GetAllChartDataJson"
        , { UserID: $scope.UserID, VitalTypeID: 4 }
        , function (p) {

            var data = p.response.data;
            DeltaCompress.decompArrayObj(data, 0);
            for (var i = 0; i < data.length; i++)
                data[i][0] = data[i][0] * 1000; // convert to milliseconds
        })
    }
}

```

```

        var chartConfig = $scope.getBaseChartConfig(data);
        var containerId = "painlevelchart";
        $('#'+ containerId).highcharts("StockChart", chartConfig);
        $scope.chart = $('#'+ containerId).highcharts();
    });
} // end init
});

```

The first part of the code, “getBaseChartConfig” function, creates the JSON object that needs to be used to initialize the chart component.

The second part, “init” function, gets the data from the server and creates the chart component. Statement `$scope.ExecInline("GetAllChartDataJson"` calls a function with the provided name on the server to get the data based on the provided user identifier and the Vital Value Enumerate type. This function also compresses the data to reduce the bandwidth usage and the response time (it is because less data will be transferred). Statement “`DeltaCompress.decompArrayObj`” decompresses the data. To convert seconds to milliseconds, we need to multiply the time field by 1000. Finally, we use JQuery function to find the container tag in the HTML code and replace it by the chart component.

Using JQuery in an AngularJS application has several disadvantages. It reduces testability and makes it hard to have several instances of the same controller on the page. We have used JQuery here to show how easily unsupported components can be used in the UI.

The UI definition is simple for this form because we only need a placeholder for a ready-to-use chart component. The following code shows the HTML design behind this form:

```

@{
    var entityName = "VitalValue";
}
@{ ViewBag.ViewTitle = "Pain Level Chart"; }

```

```

@Html.Partial("~/Views/Dashboard/_Layout/_MasterViewHeader.cshtml")
<form role="form"
    ng-controller="@{(entityName)}PainLevelChartController"
    id="@{(entityName)}PainLevelChartFrm"
    onsubmit="return false;">
    <a href="#VitalValue-PainLevelForm" class="btn btn-default">Record New</a>
    <div class="row">
        <div class="col-md-12">
            <div id="painlevelchart" style="height: 450px; min-width: 400px"></div>
        </div>
    </div>
</form>
@Html.Partial("~/Views/Dashboard/_Layout/_MasterViewFooter.cshtml")

```

We can bind our JavaScript code with the AngularJS controller using “ng-controller” attribute. Statement “<div id=’painlevelchart’ ...>” provides a placeholder to draw the chart, and “” redirects the user to the insert form. This form is shown in Figure 66.

To access the chart form, we need to put the hyperlink on our menu. “#VitalValue-PainLevelChart” shows the chart for the current patient. Doctors can see their patient’s chart by accessing the form using this hyperlink: “#VitalValue-PainLevelChart?MasterID={PatientID}” where Patient ID parameter is the user identifier of that patient. Therefore, we can reuse the code for both doctors and patients.

5.2.4 Record Sleep Form

The implementation of Record Sleep form is very similar to the previous data entry forms (e.g. Pain Level form) that was explained before. Thus, we skip the explanation here. The only

difference is that this form uses DialyActivity entity of the framework instead of VitalValue.

Figure 68 shows a screenshot of this form.

Figure 68. Screenshot of Record Sleep Form

The following is the programming part of the form in JavaScript:

```
entityControllerFactory.create("DailyActivity", "SleepForm", function ($scope,
sParams) {
    BaseControllers.AddEditFormServices($scope, sParams);
    // model for duration calculator
    $scope.uimodel = {
        hours: 0,
        minutes: 0,
        seconds: 0
    };
    $scope.onBeforeInsert = function () {
        $scope.model.UserID = $scope.CurrentUserID;
        $scope.model.DailyActivityTypeID = 7; // Sleep
        $scope.model.IsManualEntry = true;
        $scope.model.Calorie = 0;
    };
});
```

```

        $scope.model.DurationSeconds = ($scope.uimodel.hours * 3600) +
        ($scope.uimodel.minutes * 60) + $scope.uimodel.seconds;
    }
    $scope.onSuccessInsert = function () {
        $scope.GotoView("DailyActivityChart-CalorieChart");
    }
});

```

This form shows the way we can define the controller when the user interface model is different from the server-side model. For instance, the duration value in the server is stored as seconds, but it is easier for the user to enter it in the form of hours, minutes and seconds. These kinds of calculations can be done in “onBeforeInsert” function of the framework.

5.2.5 Physical Activity Form

Physical Activity form lets the user record his physical activity. A view of this form is shown in Figure 69. This form is similar to other data entry forms except the part that calculates the calorie and fills out the patient’s data automatically. Some parts of the JavaScript code behind the form is as follows:

```

entityControllerFactory.create("DailyActivity", "FitActivityForm", function ($scope,
sParams) {
    BaseControllers.AddEditFormServices($scope, sParams);
    var fitActivityService = EntityService.create("FitActivity", sParams.$http,
sParams.ngAuthSettings);
    var personService = EntityService.create("Person", sParams.$http,
sParams.ngAuthSettings);
    // model for calorie calculator
    $scope.uimodel = {
        ... // removed code for readability
    };

    $scope.init = function () {

```



```

$scope.ViewNewForm(); // insert form
... // removed code for readability
personService.GetByID({
    id: $scope.CurrentUserID,
    successFn: function (e) {
        var person = e.response.data;
        if (person.GenderTypeID === 0)
            $scope.uimodel.gender = "male";
        else
            $scope.uimodel.gender = "female";
        $scope.uimodel.age = new Date().getFullYear() - new
Date(person.Birthday).getFullYear();
    }
});
$scope.model.RecordDateTime = new Date();
}
... // removed code for readability
$scope.$watch(
    function() {
        return angular.toJson($scope.uimodel);
    }
    , function (newValues, oldValues, scope) {
        $scope.CalculateGrossCalorie();
    }, true);
... // removed code for readability
$scope.CalculateGrossCalorie = function () {
    ... // removed code for readability
    $scope.calculatedCalorie = caloricexpenditure;
    return caloricexpenditure;
}
});

```

The service objects created using “EntityService.create” provides access to the server-side data.

The architectural design of the framework doesn’t allow rendering the data inside the view in the

server-side code. This is because we need to keep the separation between the data and the UI. It also helps to cache all UI views and increase the performance by reducing the response time.

Figure 69. Screenshot of Record Physical Activity Form

This form can get the patients' demographic data using "GetByID" function of the "personService" object. It updates the model after receiving the data from the server. As changing any data on the form affects the calorie calculation, we need to watch the changes and update the calculation on the model. The function defined in "\$scope.\$watch" does this. It tracks the changes in the user interface and recalculates the calorie computations.

5.2.6 Record Food Forms

Another requirement of this representative application is to let the patients record their eating habits. The design of this part consists of a Master and a Detail page. The master page is called Food Group; it can record the time and type of the food. Figure 70 shows the master form. Food Group title in this form lets the user pick a general name for its food group to reuse it later. Each

Food Group can have several food items as its detail. The user can add food items (details) by clicking on “Add New” button. This button redirects him to “Add Food Item” page (shown in Figure 71).

Food Group

Food Serving Time Type
Breakfast ✓

Record Date Time User Local
2016/04/19 5:3:44

Food Group Title
Typical breakfast ✓

Items

[Add New](#)

Food	Amount	Calorie	Fat (g)	Carb (g)	Protein (g)	
Jams, preserves, marmalades, low sugar (all flavors)	20 gram	100	100	10	20	Edit Delete

[Save](#)

Figure 70. Screenshot of Food Group Add Form

This page is similar to our data-entry forms (e.g. Pain Level form). It has several fields for entering food name, serving amount, calorie, fat, protein and carb amount. Data entry of the nutrition facts should be done by the user, but it is possible to integrate this part with the nutrition API providers.

Please enter the food name and calories consumed

Food

Jams, preserves, marmalades, low sugar (all flavors) ✓

Serving Amount

20 ✓

Food Serving Type

gram ✓

Calorie

100 ✓

Fat

100 ✓

Protein

20 ✓

Carb

10 ✓

Save

Figure 71. Screenshot of Add Food Item Form

To provide a better visualization of the entered data, we will use a chart to show the calorie taken, burnt and its difference. A view of this form is shown in Figure 72.

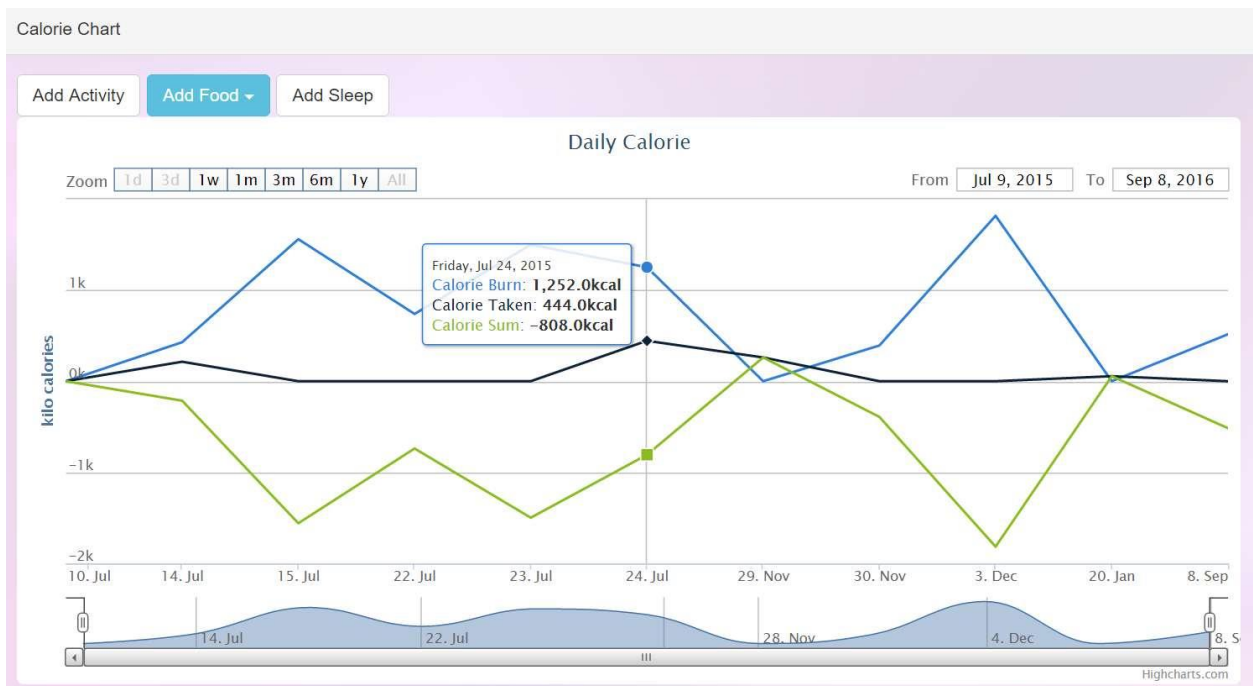


Figure 72. Screenshot of Calorie Chart Form

The client-side programming of this form is very similar to our Pain Level Chart form. In the server-side, however, we need to create a view in the database. The rest will be generated by the code generator.

The Analyzer module can do the computations and store it in a database table. Managing appointments and Video communication functionality can be added to this app by adding the links to the menus similar to our breast cancer app.

We have used some features of the platform in the representative apps such as Vital Values or Daily Activity that were not discussed in the base platform (i.e. previous chapter). It is noteworthy to mention that the features explained in the representative apps can also be reused for the development of other applications.

5.3 Chapter Summary

To demonstrate capabilities of our platform, we introduced two representative apps in this chapter. The first app was related to a research of breast cancer risk assessment tool to be designed and developed. In the second app, we have shown how the platform can be fully customized to support a post-surgical research; this app required patients to enter some data such as their mood (pain level), sleep hours, the taken calorie and burnt during the day. It also needed to draw several charts from the entered data. We have shown how the user interface can be integrated with other third-party components (e.g. charts) and how the current features of the platform can be reused to simplify the development process. The simplicity of the development process shows how easy it is to reuse the base platform functionality and develop new apps.

CHAPTER 6

EXPERIMENT AND RESULTS

6.1 Research methodology

This study had several different research problems; thus, we needed to validate each part using its appropriate research method: 1- As the proposed solution is mostly design and a framework and not a healthcare application, we didn't do a clinical trial. We have conducted a user study to validate this base platform functionality. In fact, by showing that the proposed functionality is working properly, we can infer that the applications that use the same features can function correctly as expected. 2- As it is very hard to validate the quality features, we explain the design considerations to ensure the quality later in the next section of this chapter. 3- The results of the experience of the augmented reality module is explained in more detail in another section of this chapter.

6.2 The proposed platform user study

To validate the base platform functionality, we have asked several potential users to follow several scenarios and rate their experience. Each user was asked to register in the system, try its functionality and fill out a survey related to the experience. We selected the users from the students who are not in the computer science or related majors as they are more familiar with the software and it may bias the conclusions. The following instruction is given to the patient users to follow:

- 1- Sign up in the website <https://www.xecare.com/> as patient. Please use your real email and phone number to get the notifications. Please write down your username and password on a paper for future use.
- 2- Fill out your personal information under “My account” page.
- 3- Add your insurance information (fake data)
- 4- Fill out Medical History forms with fake data. Allergy and first three questions of Social page is enough. Sign the data.
- 5- Book an appointment with your doctor. Your doctor is called “Demo Doctor.” he an African American man who is licensed to work in Texas. Verify that he can practice in Texas before adding him. His phone number starts with “469230”.
- 6- Logout from the system and login again.
- 7- Answer all survey questions.

And the following steps was required for doctor users to follow:

- 1- Sign up in the website <https://www.xecare.com/> as a doctor. Please use your real email and phone number to get the notifications. Please write down your user name and password on a paper for feature use.
- 2- Fill out your personal information under “My account” page.
- 3- Fill out your clinical information under “Professional Info” page. (Use may fake information)
- 4- Set in your calendar that you are available 10:30 A.M. and 11:00 A.M. on November 10th.
- 5- Book an appointment with your patient. Her first name is “Demo” and her last name is “Patient.”

6- Logout from the system and login again.

7- Answer all survey questions.

6.2.1 Discussion

The results of the surveys are shown in Table 2 and Table 3. These tables show that the implemented features are working properly. In each part, the users were able to complete the majority of the tasks without a problem. This indicates that the provided base application is functioning as expected.

Table 2. Survey results for the patient users

Feature	Outcome	Completed users
Sign up	I could see the Patient's menu.	5
Demographic information	I could see my account information.	5
	I could add my demographic information successfully.	
	I could see my profile information is completed based on the information I provided.	
Insurance information	I could see my insurance added to the list successfully.	5
Medical History - Allergy	I could enter my allergy information successfully.	5
Medical History – Social	I could enter my social behavior questionnaire successfully.	5
Medical History - Sign	I could sign my medical history form.	5
Finding doctor by phone number	I could see a list of three doctors when searching the provided number.	5

Feature	Outcome	Completed users
	I could see my doctor is licensed in Texas.	
	I could see “Demo Doctor” is added to my list.	5
Book an appointment with the doctor	I could see the appointment I booked is added to my appointments list.	5
Log out	I could successfully log out of the system and I saw the login page.	5
Login	I could enter the system again and see my main menu.	5

Table 3. Survey results for the doctor users

Feature	Outcome	Completed users
Sign up	I could see the Doctor’s main menu.	2
	I could see my account information.	2
	I could add my professional information successfully.	2
Calendar	I could define my availability calendar.	2
Book an appointment with the doctor	I could find my patient.	2
	I could book an appointment with my patient.	2

Although these features do not cover all the features that are available in the framework, it provides evidence of the usability of the applications developed on this platform. Features such as vital values, or time-series data are excluded from the experiments because it was only available in the representative applications and required many more steps to follow. In addition, they are using the exact same functionality of the framework that are manually and automatically tested.

To have a better understanding of the proposed platform, we would like to compare it with similar works such as Apple CareKit (and ResearchKit) and Berkley Telemonitoring. Table 4 provides a brief comparison between the proposed platform and these platforms.

Table 4. Comparison of our platform with Apple CareKit and Berkley Telemonitoring

	Apple CareKit	Berkley Telemonitoring	Our system
Target User	iOS Developers	Medical Professional	Less experienced Web Developers
Languages	SWIFT, Objective-C	Java	HTML, CSS, C#, JavaScript
Server-side	N/A	Very basic, Java	Comprehensive, ASP.NET
Client-side	iOS native app	Android Native app	Cordova for Android and iOS
Main features	Task management, Symptom tracker, charts, alerts, contacts, survey	Vital signs reading from devices via Bluetooth, Survey	(Discussed earlier)

It can clearly be seen that each platform is good for some sort of applications. The target user (developer) in Apple CareKit is iOS developers who know SWIFT or Objective-C languages. However, Berkley Telemonitoring focuses on Medical professionals. The app in this platform should be developed using Java. Our platform is designed for less experienced developers. The code generators, several base classes, the generalization of common scenarios in medical research apps and reducing the required code make it easier for less-experienced developers to develop apps

using our platform. We believe that this is more flexible than generating the whole code and requires less effort than developing many things from scratch. Furthermore, our platform has many modules that are not available in both Apple CareKit and Berkley Telemonitoring. Its server-side functionality is an advantage for those who are going to conduct research works with the requirements of larger data gathering and analysis.

In short, if the research is going to be done using iOS smartphone with no server-side requirement (or very minimum data handling on the server-side), AppleCare Kit would be a better choice. If Android smartphones are the target and the app requires connecting to many Bluetooth devices, Berkley Telemonitoring seems to be a better solution. If the application requires a fully-functional server, our platform is the best. Nonetheless, our platform can be used as a server-side option for both Berkeley Telemonitoring and Apple CareKit. In other words, the combination of the solutions can be considered when required.

6.3 Quality Features

It is not easy to validate the quality features of a software framework. Thus, we emphasize how the quality of the system is considered in the design and the development of the framework. In each section, we define the meaning of the quality measure, and we explain how we considered it in the design and implementation of our framework.

6.3.1 Security and Privacy

Authentication is to ensure that the user is who he/she declares himself/herself to be. A unique username and password is the most common approach to implementing it. We are using an API-based authentication standard, OAuth 2.0. It generates access tokens from the server and sends it

to the client. The client will use this token to access the system. This also helps to support a single-sign-on as the token can be accessible from other systems. This way the user can login to one system and get the token and access the other system using the same token. Furthermore, it is possible to have a web-based authentication for the other systems like what Facebook and Google have for authenticating the users.

Authorization defines access policy to any resource of the system (including services and data) for each user. In our implementation, the security is a part of the service layer. We have designed some predefined security classes for common scenarios in the platform. ServiceSecurityException class is developed to be used for throwing security exceptions. Table 5 provides a few security checker classes that are implemented in the platform.

Table 5. Basic Security checkers classes in the platform

Class Name	Checks that the service is called by
OwnerDoctorSecurity	a doctor who is assigned to the record
OwnerPatientOrADoctorSecurity	the patient who is assigned to the data or any doctor
OwnerPatientOrOwnerDoctorSecurity	the patient or a doctor who is assigned to the data
DoctorRoleSecurity	a user who is a doctor (any doctor)

The following code is a sample of the usage in the service layer:

```
public void CancelVisit(long visitId)
{
    vVisit visit = GetByIDV(visitId, new GetByIDParameters());
    ((VisitBR)BusinessLogicObject).CancelVisit(visit);
}
```

```
OwnerPatientOrOwnerDoctorSecurity.Check("Cancel a visit", visit,  
vVisit.ColumnNames.PatientUserID, vVisit.ColumnNames.DoctorID);  
... // Some code removed for readability  
}
```

In this example, `OwnerPatientOrOwnerDoctorSecurity` class is used to check if the user who is going to cancel an appointment is the related doctor or the related patient. It means that one doctor cannot cancel an appointment of other doctors mistakenly. It is true that doctors cannot see appointments of other doctors in the user interface, so they shouldn't be able to call this service from the UI. However, since we have a total separation of the layers, all of the security and business rules should be checked on the server-side. APIs can be called from other systems and a potential bug in those systems may result in calling a service with incorrect parameter values. In addition, JavaScript clients are not generally safe and a potential hacker can easily modify the parameter values to cause problems in the system.

HIPAA Compliance

Since HIPAA compliance has more to do with the software system than the architectural design of the software, we try to keep it short here. All requests going to the web server are encrypted using HTTPS and Secure Sockets Layer (SSL) certificates. Data stored on the database server is stored on encrypted drives. The connection between the database server and the web server is encrypted using SSL. Backups are configured to create a backup frequently. Security layer checks authorization of the user to access the data. The integrity of the data can be checked using automated audit trails. The logging system also logs all the service accesses. All of the files stored in the file server (Amazon S3) are encrypted. The access to the files is provided using tokens and the log information of the user who requested it is saved in the database.

6.3.2 Extensibility

In this study, extensibility is defined as the ability to add more features without doing major changes to the framework. We have considered several things to make this platform extensible. The user interface is all based on the web, and the web has a very extensible nature. New components can easily be added to the user interface by including their JavaScript files. The AngularJS framework is also very flexible in the implementation of new components. On the server-side, a Factory class is responsible for creating all entity classes like services. This class can easily be customized to load services from other assemblies. In addition, the code generator is a great help to add new server-side components based on the database design.

6.3.3 Reliability

From the architectural point of view, all APIs are designed stateless so that the traffic can go to different servers. Therefore, the state of the software is not shared between servers to make a point of failure on the server. In fact, there is no one point of failure in the design. The database reliability is a concern of the DBMS providers and deployment. So, the configuration of such systems should avoid centralized deployment. This makes the final system reliable.

6.3.4 Scalability

Scalability is defined as the ability of the system to serve more users as the number of the users grows. In the software architecture, distributing the software layers on separate platforms is one of the best practices. So, we have designed our architecture based on the n-tier design. This helps to distribute the layers among separate servers. The API is completely stateless; therefore, a load balancer can distribute the load on several servers. New servers can be added to the system

whenever the traffic is high, and in a low traffic time, the unnecessary servers can be safely terminated. In addition, the user interface doesn't render any user-related data. Thus, it can easily be used behind the load balancers. NoSQL databases may be necessary to be used in the data access layer to handle a large amount of data.

6.3.5 Maintainability

We have tried to keep a high quality for the maintainability in the architecture. The code generator helps to make major modifications without being concerned about the programming errors as the code is generated from one base. Therefore, changing the models from the database and generating the code can prevent developers' mistakes in naming variables or creating models.

The code is well commented so that new developers can easily understand and modify the code. There are more than 20,000 lines of comments in the provided base code. The majority parts of the code have automated test cases that help to prevent unwanted bugs in changes. More than 300 test cases are defined for automatically test the code. If these test cases pass whenever the developer applies some changes to the code, he can ensure that it didn't break any other part of the code.

Furthermore, the platform configuration is separated from the code. The following code shows some parts of the configuration file of the platform:

```
<applicationSetting>
  <security enabled="true" accessHelperClassFullName="" />
  <log enabled="true" writeToLog4NetEnabled="true" mainWriterClassFullName=""
  backupWriterClassFullName="" />
  <exceptionHandling
  exceptionTranslatorClassFullName="Framework.Common.Exceptions.ExceptionTranslator,
  Framework" showDebug="true" />
  <localization cultureName="en-US" tzdbFile=" " />
```

```

    <project title=" " projectConfigProviderClassFullName=" " namespacePrefix=" "
factoryClassFullName=" " useSchemaWithEntityName=" "
serverPrefixDigitsForIDGenerator="6" serverInfoConfigFile=" " emailVerificationUrl="
" continueQuickRegisterUrl=" " phoneNumberVerificationUrl=" " websiteUrl=" "
googleApiServerKey=" " appleCertificatePassword=" " videoServiceHost=" "
videoMeetingName=" " videoDefaultConnectionType=" "
resetPasswordNewRequestWaitSeconds="60" resetPasswordCodeExpireSeconds="600"
resetPasswordUrl=" " companyAddress=" " mainDomainName=" " />

    <paypal mainAccount=" " endpoint="api.sandbox.paypal.com" clientId=" " secret=" "
userName=" " password=" " signature=" " applicationID=" " connectionTimeout="12000"
requestRetries="2" isSandBox="true"
senderRedirectUrl="https://www.sandbox.paypal.com/webscr&cmd=_ap-
payment&paykey=" senderRedirectMobileUrl=" " cancelPage=" " returnPage=" "
webRootUrl=" " />

</paypal>

<twilio accountSid=" " authToken=" " fromNumber=" " enabled="false">

</twilio>

<amazonCloud>

    <s3 bucketName=" " regionEndpoint=" " mainAccount=" " accessKeyID=" "
secretAccessKey=" " successRedirectUrl=" " defaultUploadAcl=" "
defaultSignedUrlExpirationMinutes="5" defaultServerSideEncryption="AES256">

    </s3>

</amazonCloud>

<mobilePush realtimeNotifyClassFullName=" ">

</mobilePush>

<databaseContexts>

    <context name="default" settingsPath=" " isTransactional="true"></context>

</databaseContexts>

</applicationSetting>

```

As the settings are separated from the code, it is easier to maintain the system.

6.3.6 Performance

Performance is one of the main quality features. Lack of performance may cause negative side effects on the software. We have considered several points to increase the performance of our system. Web bundling features are used to bundle several files and serve them in one request. This reduces the number of requests from the browser and decreases the wait time. The UI is designed

and served separately from the data. Therefore, all pages can be loaded from a cache server or even stored in the client-side application. Single Page Application design is considered for the client-side user interface. It only loads the requested page using Ajax. All data is served using the API in JSON format. Thus, each view only loads the minimum amount of data that is required to show the view. Frequently accessed data and services may also be cached in the memory to improve the performance.

6.3.7 Portability

We defined portability as the ability to run the user interface on different operating systems with minimum modifications. The architectural design is based on the web; thus, the applications run on iOS, Android, Windows or Linux as long as they have a supported browser. Responsive design is one of the features of web user interfaces that makes it easier to use the same code for different devices and screen sizes. All of the designed user interfaces are responsive, and can easily be seen on the screen of a smartphone, a tablet or a desktop computer.

The web standards don't support many native device APIs such as Bluetooth connections. These features need to be implemented in the native code. We have designed a client application based on Apache Cordova platform that runs our platform, and it allows running the native codes by including its plugins. Since these plugins are written in the platform native code, they can use almost any feature that is available in the operating system. For example, the web browser doesn't let the user connect to a Bluetooth device, but our web platform can access the Bluetooth devices using its Cordova native application.

6.3.8 Reusability

There are several considerations in the platform design to increase reusability. It is easy to inherit from the provided base classes, and their methods can easily be overridden by the developers. There are also some events in the classes that make it easier to customize the functionality of the class. We have also implemented a special feature for the user interface to let the developer reuse all or some parts of the currently designed user interface. This feature is explained in more detail in the section related to the representative applications. The framework library also has many classes that can be reused directly such as encryption or compression.

6.4 AR Module Experiment

To test the AR module, we printed the markers shown in Figure 59. We also made a 3D model of a sample tumor. In the earlier stages, we used a bra to represent the breast and the real sense. We placed the marker on the left breast and adjust the tumor view on the right side. The smartphone application deployed on Samsung Galaxy S5 that is an Android phone. Figure 73 demonstrates the screenshots of the application from different perspectives. The application works smooth on this phone when it is configured for 800x600 pixels resolution, but it is not smooth (less than ten frames per second) in 1920x1080 pixels resolution.

We made markers with bigger size and enlarged the 3D model to be viewable in the figures reported in this paper. In general, smaller markers can be used with the same results. To obtain accurate results, the size of the printed marker should be adjusted to the size of 3D model and camera calibration.

6.4.1 Discussion

The proposed system is a new tool to help breast tumor visualization. Based on our experiments, this visualization has several potentials such as better screening and surgery planning. Another potential application is using this tool as a communication tool between radiologist and doctors. A radiologist who has expertise in dealing with 3D models can provide annotated 3D models by using one 3D model or a combination of several models for example obtained from 3D imaging system (e.g. x-ray, ultrasound, and MRI), and send the model to the doctor. Doctors can view this on a plastic model in their office for screening or on the patient body for surgery planning. It simplifies viewing of the models from different perspectives.

Google Glass didn't meet our expectations for this tool. Our hypothesis was to provide an AR experience for doctors to look at the tumors while looking at the real-world view. However, the way Google Glass handles it does not exactly give the both views at the same time. In order for the user to see the augmented world, he needs to look at the Glass screen, and it distracts him from perceiving the world with his eyes. In fact, it is not possible to have the both view simultaneously. In addition, the low-resolution screen of Glass (640x480 pixels) makes it very hard for doctors to see things well. In a nutshell, we can't confirm the previous findings of the usefulness of Google Glass for surgery. Even though not bad, it is not even good enough for a diagnosis.

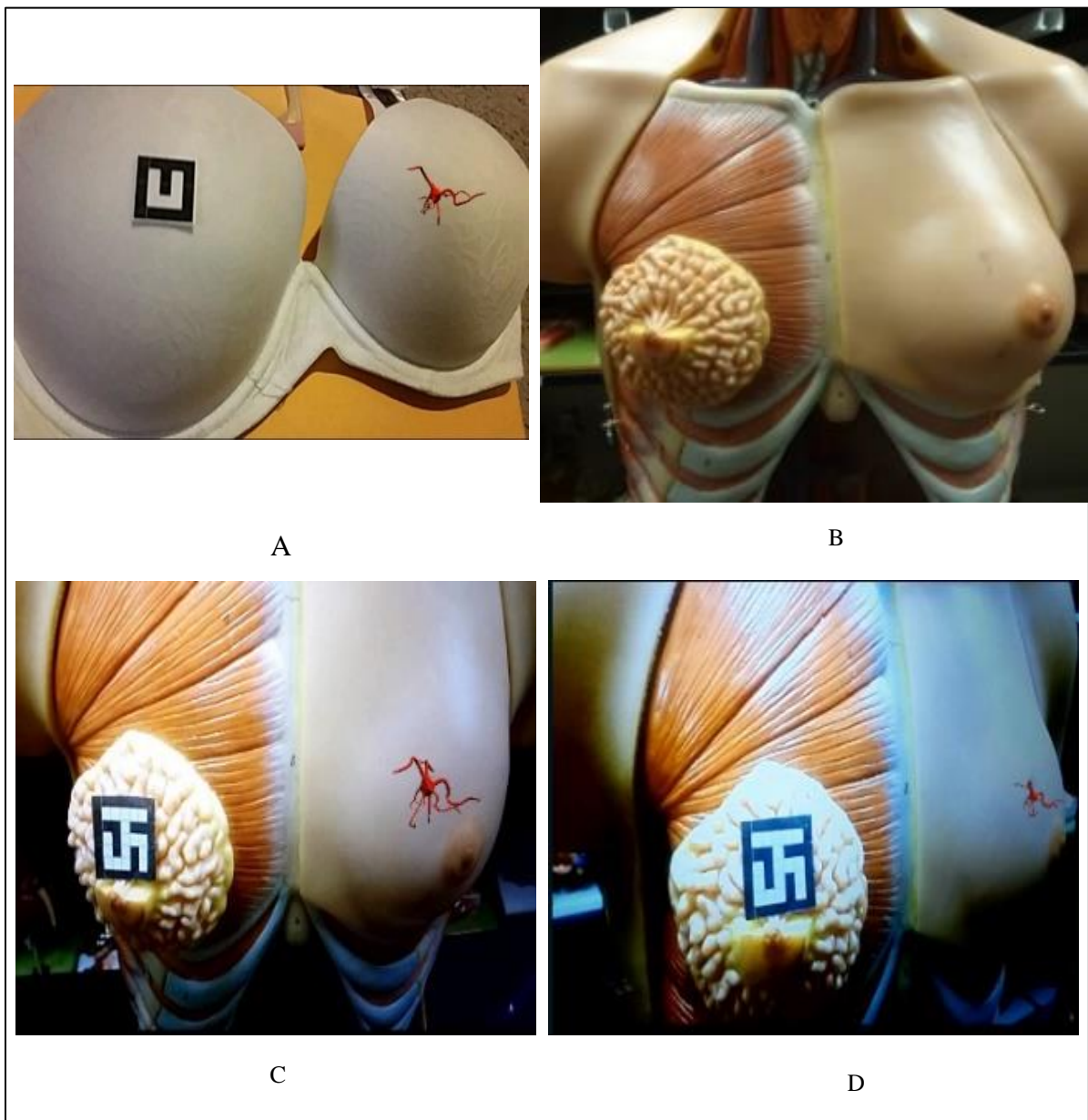


Figure 73. Experiments of the augmented reality system. (A) Screenshot (what a physician sees on the screen) tested on a bra. (B) The original plastic model without AR (C) Screenshot on a plastic model. (D) The same as B, but from a different angle.

There are limitations in our methodology. First, we assumed that the marker put on the body will remain fixed during the study for real patients. Obviously, misplaced marker will cause inaccuracy

in the registered model. This marker-based approach may be replaced by a marker-less one. Secondly, we didn't do a comprehensive study on the accuracy in this proof of concept research as several research works have already studied some aspects of this problem [76], [77], [78]. Thirdly, more tests and feedbacks from users are needed for head-mounted displays. A device like Epson Moverio BT-200 Smart Glasses [79] that doesn't block the user's view is perhaps more preferable than a smartphone.

6.5 Conclusion

Currently available mobile apps for disease management are like islands in the ocean of healthcare apps. They are not fully integrated into the caregiving process. Hence, telemedicine solutions are trying to find their position in healthcare. In this study, we proposed a flexible software platform that can be used to develop medical research applications. We explained the main features available in the base platform to develop disease management research applications, and we investigated essential features for such system. These features include management of account, profile, a list of doctors, a list of patients, doctors' availability calendar, appointments, medical history, payments and video communication. In addition, we have introduced some features such as feedback services and voice recognition.

We have shown how developers can take advantage of the platform to create the required questionnaire forms based on the medical history feature and let the user enter the data. This can significantly reduce the cost of obtaining data, and let the researchers focus more on analyzing the data. In addition, they can add value to the patients by providing information about the diseases and help them to more effectively connect with their doctors.

It is true that the proposed platform is not an all-inclusive framework, but it helps to significantly reduce the time of developing similar applications. The design of the software architecture is flexible enough to let the developer change and customize the platform based on the requirements. It is also very easy to integrate the system with other systems both on the client-side and the server-side.

The integration of telemedicine solutions with disease management systems is also discussed in this research work. We also presented a representative application that helps users to do breast cancer risk analysis with the help of their doctors.

6.6 Future Work

This study has some limitations similar to many other research works. We have assumed a constant Internet access in the design. In other words, our design is not offline-first. Many mHealth applications need to work offline. The proposed native mobile application works as a wrapper for the website, and it doesn't work offline. Although HTML5 applications can be used offline using manifest files, the future of such feature is not clear. It is better that the future works consider offline-first design. The integration with currently available frameworks such as Berkeley Telemonitoring system is also desirable.

We have started with a totally different requirements set. Therefore, not all parts of the code and design are perfect for the proposed usage. Our requirements to make the server-side code compatible with Java programs put many restrictions in the design and choosing of the components. In addition, the design of the framework layers caused a lot of complexity. This increases the learning curve, and it is in contrast with the goal of being used by less experienced

developers. Most of the applications probably don't need such complexity. Therefore, removing such complexity would make it easier for new developers to adopt the framework.

The Analysis module performs on a separate server, and it runs independently from the apps. It is better to provide some of its functionality in the framework and integrate it with the developed application. For example, a messaging queue between the analyzer and the application can trigger the running calculations so that the Analysis module doesn't need to check the database frequently. We haven't considered standardizations as a part of the design because it is a very time-consuming task. Making standard APIs can be a great help for integrating the developed applications. For example, Open mHealth project (openmhealth.org) provides several schemas for sending and receiving vital signs. Adopting these standards and defining new standards can be very beneficial in long-term usage of the developed applications.

Moreover, we are going to add more modules to the project. This will reduce the development time of the new applications. Our future plan is to use this platform for several research projects to identify new requirements and implement them in the framework. For example, the Bluetooth connectivity in the applications can be generalized and it can be customized for different applications. This helps the developers to develop their applications faster.

6.7 Chapter Summary

We asked several potential users of the system to interact with the system to validate the proposed base platform functionality. They have been asked to sign up to the system and book an appointment. The results show that the users can use the basic functionality of the proposed framework without any problems. As the design and execution are similar in other features, we can conclude that they can perform as well as the basic features. We also considered quality features such as extensibility or reliability in the evaluation. For each feature, we explained what design decision are made and implemented to meet the expected quality. We also show the experimental results of the augmented reality module for visualization of the breast cancer tumors. This visualization helps doctors to view a 3D model of the tumors on the body of the patient. Image processing and capability to include more advanced modules in the framework makes it useful for future research studies [12].

REFERENCES

- [1] C. J. Murray and A. D. Lopez, “Measuring the global burden of disease,” *New Engl. J. Med.*, vol. 369, pp. 448–457, 2012.
- [2] D. J. Hunter and K. S. Reddy, “Noncommunicable diseases,” *New Engl. J. Med.*, vol. 369, pp. 1336–1343, 2011.
- [3] I. Kalsekar, S. Record, K. Nesnidal, and B. Hancock, “National estimates of enrollment in disease management programs in the United States: an analysis of the National Ambulatory Medical Care Survey data,” *Popul. Health Manag.*, vol. 13, pp. 183–188, 2009.
- [4] T. Bodenheimer, E. Chen, and H. D. Bennett, “Confronting the growing burden of chronic disease: can the US health care workforce do the job?,” *Health Aff.*, vol. 28, pp. 64–74, 2008.
- [5] A. Stuckey, “Breast cancer: epidemiology and risk factors,” *Clin. Obstet. Gynecol.*, vol. 54, pp. 96–102, 2009.
- [6] P. Pisani, F. Bray, and D. M. Parkin, “Estimates of the world-wide prevalence of cancer for 25 sites in the adult population,” *Int. J. cancer. J. Int. du Cancer*, vol. 97, pp. 72–81, Jan. 2002.
- [7] R. Siegel, D. Naishadham, and A. Jemal, “Cancer statistics, 2012,” *CA: Cancer J. Clin.*, vol. 62, pp. 10–29, 2011.
- [8] V. Williams, J. Price, M. Hardinge, L. Tarassenko, and A. Farmer, “Using a mobile health application to support self-management in COPD: a qualitative study,” *Br. J. Gen. Pract.*, vol. 64, pp. e392–e400, 2012.

- [9] M. Hardinge, H. Rutter, C. Velardo, S. A. Shah, V. Williams, L. Tarassenko, and A. Farmer, "Using a mobile health application to support self-management in chronic obstructive pulmonary disease: a six-month cohort study," *BMC Med. informatics Decis. Mak.*, vol. 15, 2013.
- [10] N. F. BinDhim, A. M. Shaman, L. Trevena, M. H. Basyouni, L. G. Pont, and T. M. Alhawassi, "Depression screening via a smartphone app: cross-country user characteristics and feasibility," *J. Am. Med. Informatics Assoc.*, p. amiajnl-2014, 2012.
- [11] L. S. Wallace and L. K. Dhingra, "A systematic review of smartphone applications for chronic pain available for download in the United States.," *J. opioid Manag.*, vol. 10, pp. 63–68, 2011.
- [12] M. Hussain, A. Al-Haiqi, A. A. Zaidan, B. B. Zaidan, M. L. M. Kiah, N. B. Anuar, and M. Abdulnabi, "The landscape of research on smartphone medical apps: Coherent taxonomy, motivations, open challenges and recommendations.," *Comput. methods programs Biomed.*, vol. 122, pp. 393–408, Dec. 2015.
- [13] D. Aranki, G. Kurillo, A. Mani, P. Azar, J. V. Gaalen, Q. Peng, P. Nigam, M. P. Reddy, S. Sankavaram, Q. Wu, and R. Bajcsy, "A Telemonitoring Framework for Android Devices," in *2016 IEEE First International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE)*, 2016, pp. 282–291.
- [14] W. E. Todd and D. B. Nash, *Disease management: a systems approach to improving patient outcomes*. American Hospital Pub., 1996.
- [15] S. Fox and M. Duggan, "Mobile health 2012," *Washington, DC: Pew Internet & Am. Life Proj.*, 2011.
- [16] J. A. Cafazzo, M. Casselman, N. Hamming, D. K. Katzman, and M. R. Palmert, "Design of an mHealth app for the self-management of adolescent type 1 diabetes: a pilot study," *J. Med. Internet Res.*, vol. 14, 2011.
- [17] B. Holtz and C. Lauckner, "Diabetes management via mobile phones: a systematic review," *Telemed. e-Health*, vol. 18, pp. 175–184, 2011.

- [18] M. de Ridder, J. Kim, Y. Jing, M. Khadra, and R. Nanan, "A systematic review on incentive-driven mobile health technology: As used in diabetes management," *J. Telemed. telecare*, p. 1357633X15625539, 2014.
- [19] M. Suh, C.-A. Chen, J. Woodbridge, M. K. Tu, J. I. Kim, A. Nahapetian, L. S. Evangelista, and M. Sarrafzadeh, "A remote patient monitoring system for congestive heart failure.," *J. Med. Syst.*, vol. 35, pp. 1165–1179, Oct. 2011.
- [20] A. Pandey, S. Hasan, D. Dubey, and S. Sarangi, "Smartphone apps as a source of cancer information: changing trends in health information-seeking behavior.," *J. Cancer Educ. Off. J. Am. Assoc. Cancer Educ.*, vol. 28, pp. 138–142, Mar. 2013.
- [21] J. Anhøj and C. Møldrup, "Feasibility of collecting diary data from asthma patients through mobile phones and SMS (short message service): response rate analysis and focus group evaluation from a pilot study," *J. Med. Internet Res.*, vol. 6, 2002.
- [22] A. Weaver, A. M. Young, J. Rowntree, N. Townsend, S. Pearson, J. Smith, O. Gibson, W. Cobern, M. Larsen, and L. Tarassenko, "Application of mobile phone technology for managing chemotherapy-associated side-effects," *Ann. Oncol.*, vol. 18, pp. 1887–1892, 2005.
- [23] Y. H. Min, J. W. Lee, Y.-W. Shin, M.-W. Jo, G. Sohn, J.-H. Lee, G. Lee, K. H. Jung, J. Sung, and B. S. Ko, "Daily collection of self-reporting sleep disturbance data via a smartphone app in breast cancer patients receiving chemotherapy: a feasibility study," *J. Med. Internet Res.*, vol. 16, 2012.
- [24] E. Basch, A. Iasonos, A. Barz, A. Culkin, M. G. Kris, D. Artz, P. Fearn, J. Speakman, R. Farquhar, and H. I. Scher, "Long-term toxicity monitoring via electronic patient-reported outcomes in patients receiving chemotherapy," *J. Clin. Oncol.*, vol. 25, pp. 5374–5380, 2005.

- [25] H. Jagos, V. David, M. Haller, S. Kotzian, M. Hofmann, S. Schlossarek, K. Eichholzer, M. Winkler, M. Frohner, M. Reichel, W. Mayr, and D. Rafolt, “A Framework for (Tele-) Monitoring of the Rehabilitation Progress in Stroke Patients: eHealth 2015 Special Issue.,” *Appl. Clin. informatics*, vol. 6, pp. 757–768, Dec. 2015.
- [26] J.-H. Cho, Y.-H. Choi, M.-J. Kang, H.-S. Kim, J.-A. Shin, J.-H. Lee, and K.-H. Yoon, “Urgent Need of Ubiquitous Healthcare for Chronic Disease Management: Focused on Diabetes for the First Step,” in *2010 10th IEEE/IPSJ International Symposium on Applications and the Internet*, 2010, pp. 379–382.
- [27] R. Rosso, G. Munaro, O. Salvetti, S. Colantonio, and F. Ciancitto, “CHRONIOUS: an open, ubiquitous and adaptive chronic disease management platform for chronic obstructive pulmonary disease (COPD), chronic kidney disease (CKD) and renal insufficiency,” in *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, 2009, pp. 6850–6853.
- [28] “Introducing ResearchKit,” 19-Oct-2016. [Online]. Available: <http://researchkit.org/>.
- [29] “Introducing CareKit,” 19-Oct-2016. [Online]. Available: <http://carekit.org/>.
- [30] M. Paschou, E. Sakkopoulos, and A. Tsakalidis, “easyHealthApps: e-Health Apps dynamic generation for smartphones & tablets,” *J. Med. Syst.*, vol. 37, pp. 1–12, 2011.
- [31] J. L. Kelsey and G. S. Berkowitz, “Breast cancer epidemiology,” *Cancer Res.*, vol. 48, pp. 5615–5623, Oct. 1988.
- [32] K. McPherson, C. M. Steel, and J. M. Dixon, “ABC of breast diseases. Breast cancer-epidemiology, risk factors, and genetics,” *BMJ*, vol. 321, pp. 624–628, Sep. 2000.
- [33] S. Shapiro, P. Strax, and L. Venet, “Periodic breast cancer screening in reducing mortality from breast cancer,” *JAMA*, vol. 215, pp. 1777–1785, Mar. 1971.

- [34] A. M. Leitch, G. D. Dodd, M. Costanza, M. Linver, P. Pressman, L. McGinnis, and R. A. Smith, "American Cancer Society guidelines for the early detection of breast cancer: update 1997.," *CA: Cancer J. Clin.*, vol. 47, pp. 150–153, Jun. 1997.
- [35] L. S. Caplan, B. L. Wells, and S. Haynes, "Breast cancer screening among older racial/ethnic minorities and whites: barriers to early detection," *J. Gerontol.*, vol. 47 Spec No, pp. 101–110, Nov. 1992.
- [36] H. D. Nelson, K. Tyne, A. Naik, C. Bougatsos, B. K. Chan, and L. Humphrey, "Screening for breast cancer: an update for the US Preventive Services Task Force," *Ann. Intern. Med.*, vol. 151, pp. 727–737, 2007.
- [37] J. Heo, M. Chun, K. Y. Lee, Y.-T. Oh, O. K. Noh, and R. W. Park, "Effects of a smartphone application on breast self-examination: a feasibility study," *Healthc. informatics Res.*, vol. 19, pp. 250–260, 2011.
- [38] Y. Koibuchi, Y. Iino, H. Takei, M. Maemura, J. Horiguchi, T. Yokoe, and Y. Morishita, "The effect of mass screening by physical examination combined with regular breast self-examination on clinical stage and course of Japanese women with breast cancer," *Oncol. reports*, vol. 5, pp. 151–156, 1996.
- [39] A. Khokhar, "Short text messages (SMS) as a reminder system for making working women from Delhi Breast Aware," *Asian Pac J Cancer Prev*, vol. 10, pp. 319–22, 2007.
- [40] "Detailed Breast Cancer Risk Calculator - Dr Halls," 05-Aug-2016. [Online]. Available: <http://halls.md/breast/risk.htm>.
- [41] J. L. Bender, R. Y. K. Yue, M. J. To, L. Deacken, and A. R. Jadad, "A lot of action, but not in the right direction: systematic review and content analysis of smartphone applications for the prevention, detection, and management of cancer," *J. Med. Internet Res.*, vol. 15, Dec. 2013.
- [42] M. H. Mobasheri, M. Johnston, D. King, D. Leff, P. Thiruchelvam, and A. Darzi, "Smartphone breast applications—What's the evidence?" *Breast*, vol. 23, pp. 683–689, 2012.

- [43] J. Ricke and H. Bartelink, "Telemedicine and its impact on cancer management," *Eur. J. Cancer*, vol. 36, pp. 826–833, 1999.
- [44] K. Collie, M. A. Kreshka, S. Ferrier, R. Parsons, K. Graddy, S. Avram, P. Mannell, X. Chen, J. Perkins, and C. Koopman, "Videoconferencing for delivery of breast cancer support groups to women living in rural communities: a pilot study," *Psycho-Oncology*, vol. 16, pp. 778–782, 2005.
- [45] S. Pruthi, K. J. Stange, G. D. Malagrino, K. S. Chawla, N. F. LaRusso, and J. S. Kaur, "Successful implementation of a telemedicine-based counseling program for high-risk patients with breast cancer," *Mayo Clinic Proceedings*, vol. 88. Elsevier, pp. 68–73, 2011.
- [46] T. Starner, "Project glass: An extension of the self," *Pervasive Comput. IEEE*, vol. 12, pp. 14–16, 2011.
- [47] D. L. Baggio, *Mastering OpenCV with practical computer vision projects*. Packt Publishing Ltd, 2011.
- [48] L. A. Shluzas, G. Aldaz, J. Sadler, S. Joshi, L. Leifer, and D. Pickham, "Mobile Augmented Reality for Distributed Healthcare Point-of-View Sharing During Surgery," 2012.
- [49] O. J. Muensterer, M. Lacher, C. Zoeller, M. Bronstein, and J. Kübler, "Google Glass in pediatric surgery: an exploratory study," *Int. J. Surg.*, vol. 12, pp. 281–289, 2012.
- [50] S. Feng, R. Caire, B. Cortazar, M. Turan, A. Wong, and A. Ozcan, "Immunochromatographic diagnostic test analysis using Google Glass," *ACS nano*, vol. 8, pp. 3069–3079, 2012.
- [51] O. M. Jeroudi, G. Christakopoulos, G. Christopoulos, A. Kotsia, M. A. Kypreos, B. V. Rangan, S. Banerjee, and E. S. Brilakis, "Accuracy of remote electrocardiogram interpretation with the use of Google Glass technology," *Am. J. Cardiol.*, vol. 115, pp. 374–377, 2013.
- [52] S. Vallurupalli, H. Paydak, S. K. Agarwal, M. Agrawal, and C. Assad-Kottner, "Wearable technology to improve education and patient outcomes in a cardiology fellowship program-a feasibility study," *Health Technol.*, vol. 3, pp. 267–270, 999.

- [53] M. H. Schreinemacher, M. Graafland, and M. P. Schijven, "Google glass in surgery," *Surg. Innov.*, vol. 21, pp. 651–652, 999.
- [54] S. Nicolau, L. Soler, D. Mutter, and J. Marescaux, "Augmented reality in laparoscopic surgical oncology," *Surg. Oncol.*, vol. 20, pp. 189–201, Sep. 2011.
- [55] M. Heydarzadeh, M. Nourani, and J. Park, "An augmented reality platform for CABG surgery," *Biomedical Circuits and Systems Conference (BioCAS), 2015 IEEE*. IEEE, pp. 1–4, 2014.
- [56] M. A. Livingston, W. F. Garrett, G. Hirota, M. C. Whitton, E. D. Pisano, and H. Fuchs, "Technologies for augmented reality systems: realizing ultrasound-guided needle biopsies," *Proceedings of the 23rd annual conference on computer graphics and interactive techniques*. ACM, pp. 439–446, 999.
- [57] E. D. Pisano, H. Fuchs, A. State, M. A. Livingston, G. Hirota, W. F. Garrett, and M. C. Whitton, "Augmented reality applied to ultrasound-guided breast cyst aspiration," *Breast Dis.*, vol. 10, pp. 221–230, 1459.
- [58] Y. Sato, M. Nakamoto, Y. Tamaki, T. Sasama, I. Sakita, Y. Nakajima, M. Monden, and S. Tamura, "Image guidance of breast cancer surgery using 3-D ultrasound images and augmented reality visualization," *IEEE Trans. Med. Imaging*, vol. 17, pp. 681–693, Oct. 1998.
- [59] T. Carter, C. Tanner, N. Beechey-Newman, D. Barratt, and D. Hawkes, "MR navigated breast surgery: method and initial clinical experience." Springer, pp. 356–363, 2007.
- [60] "Mammograms and Other Breast Imaging Tests," 09-Feb-2016. [Online]. Available: <http://www.cancer.org/acs/groups/cid/documents/webcontent/003178-pdf.pdf>. [Accessed: 09-Feb-2016].
- [61] "3D Technologies Poised to Change How Doctors Diagnose Cancers," 09-Feb-2016. [Online]. Available: <http://www.fda.gov/ForConsumers/ConsumerUpdates/ucm416312.htm>.

- [62] S. M. Friedewald, E. A. Rafferty, S. L. Rose, M. A. Durand, D. M. Plecha, J. S. Greenberg, M. K. Hayes, D. S. Copit, K. L. Carlson, T. M. Cink, L. D. Barke, L. N. Greer, D. P. Miller, and E. F. Conant, "Breast cancer screening using tomosynthesis in combination with digital mammography.," *JAMA*, vol. 311, pp. 2499–2507, Jun. 2014.
- [63] Y. Zhang, Y. Zhou, X. Yang, P. Tang, Q. Qiu, Y. Liang, and J. Jiang, "Thin slice three dimensional (3D) reconstruction versus CT 3D reconstruction of human breast cancer," *Indian J. Med. Res.*, vol. 137, 2011.
- [64] G. Kurillo, A. Y. Yang, V. Shia, A. Bair, and R. Bajcsy, "New Emergency Medicine Paradigm via Augmented Telemedicine," in *Virtual, Augmented and Mixed Reality*, Springer International Publishing, 2015, pp. 502–511.
- [65] A. Tsalatsanis, E. Gil-Herrera, A. Yalcin, B. Djulbegovic, and L. Barnes, "Designing patient-centric applications for chronic disease management," *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*. IEEE, pp. 3146–3149, 2009.
- [66] T. Chomutare, L. Fernandez-Luque, E. Årsand, and G. Hartvigsen, "Features of mobile diabetes applications: review of the literature and analysis of current applications compared against evidence-based guidelines," *J. Med. Internet Res.*, vol. 13, 2009.
- [67] P. S. Gill, A. Kamath, and T. S. Gill, "Distraction: an assessment of smartphone usage in health care work settings.," *Risk Manag. Healthc. Policy*, vol. 5, pp. 105–114, Aug. 2012.
- [68] P. N. Otto and A. I. Anton, "Addressing Legal Requirements in Requirements Engineering," in *15th IEEE International Requirements Engineering Conference (RE 2007)*, 2007, pp. 5–14.
- [69] "Developers Guide to HIPAA Compliance," 02-Jun-2014. [Online]. Available: <https://github.com/truevault/hipaa-compliance-developers-guide>.
- [70] "E.164," *International Telecommunication Union (ITU)*, 01-Nov-2010. [Online]. Available: https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-E.164-201011-I!!PDF-E&type=items.

- [71] “Web Speech API Specification,” 19-Oct-2012. [Online]. Available: <https://dvcs.w3.org/hg/speech-api/raw-file/tip/speechapi.html>.
- [72] M. Sezgin, “Survey over image thresholding techniques and quantitative performance evaluation,” *J. Electron. Imaging*, vol. 13, pp. 146–168, 2002.
- [73] S. Suzuki, “Topological structural analysis of digitized binary images by border following,” *Comput. Vision, Graph. Image Process.*, vol. 30, pp. 32–46, 1983.
- [74] V. Lepetit, F. Moreno-Noguer, and P. Fua, “Epnnp: An accurate $O(n)$ solution to the pnp problem,” *Int. J. Comput. Vis.*, vol. 81, pp. 155–166, 2007.
- [75] “jPCT 3D engine - the free 3D solution for Java and Android,” 18-Feb-2016. [Online]. Available: <http://www.jpct.net/jpct-ae/>.
- [76] A. W. C. Lee, J. A. Schnabel, V. Rajagopal, P. M. F. Nielsen, and M. P. Nash, “Breast Image Registration by Combining Finite Elements and Free-Form Deformations,” in *Digital Mammography*, Springer Berlin Heidelberg, 2009, pp. 736–743.
- [77] M. A. Lago, F. Martinez-Martinez, M. J. Ruperez, C. Monserrat, and M. Alcaniz, “Breast prone-to-supine deformation and registration using a time-of-flight camera,” *Biomedical Robotics and Biomechatronics (BioRob), 2012 4th IEEE RAS & EMBS International Conference on*. IEEE, pp. 1161–1163, 2010.
- [78] A. Sotiras, C. Davatzikos, and N. Paragios, “Deformable medical image registration: A survey,” *Med. Imaging, IEEE Trans.*, vol. 32, pp. 1153–1190, 2011.
- [79] “Epson Moverio BT-200 Next Generation Smart Glasses,” 17-Feb-2016. [Online]. Available: <http://www.epson.com/moverio>. [Accessed: 17-Feb-2016].

Copyright Permissions:

Some parts of this dissertation were published in IEEE conferences:

© 2016 IEEE. Reprinted, with permission, from Mohammad Ali Ghaderi, Mehrdad Heydarzadeh, Mehrdad Nourani, Gopal Gupta and Lakshman Tamil, Augmented reality for breast tumors visualization, Aug 2016.

© 2015 IEEE. Reprinted, with permission, from Mohammad Ali Ghaderi, Gopal Gupta and Lakshman Tamil, Mohamed Abdallah and Khalid Qaraqe, mHealth Platform for Breast Cancer Risk Assessment, Oct 2015.

VITA

Mohammad Ali Ghaderi was born in Tehran, the capital city of Iran. Commodore 64 programs and games sparked his interest in programming when he was seven years old. He started computer programming more seriously with Visual Basic 4 in 1997 when he was thirteen years old. He obtained his Bachelor's degree from Shiraz Islamic Azad University in 2008. He graduated from the University of Tehran with a Master's degree in Information Technology in 2011. Information retrieval was his main research interest in the Master's program. After joining the Software Engineering Ph.D. program of the University of Texas at Dallas in 2012, he developed more interest in software engineering and healthcare applications because of their challenging nature and public benefits. He has published several papers in the international conferences and journals.