CALIFORNIA STATE UNIVERSITY, NORTHRIDGE

MedTrack: A Mobile Application for Patients, Physicians and Pharmacies; Track Your Med-Health on the go

A graduate project submitted in partial fulfillment of the requirements

For the degree of Master of Science in Software Engineering

By

Nishika Malhotra

May 2016

The graduate project of Nishika Malhotra is approved:

Dr. Steven Fitzgerald	Date
Dr. Li Liu	Date
Dr. Adam Kaplan, Chair	Date

California State University, Northridge

ACKNOWLEDGEMENT

With deep sense of gratitude, I would like to thank my thesis advisor Professor Adam Kaplan, and the thesis committee members Professor Steven Fitzgerald and Professor Li Liu for their constant support and guidance throughout my work. They have been extremely generous with their time and rendered me all possible help to see this work complete. I am fortunate to have such a talented group of outstanding professors in my thesis committee.

A special note of thanks to committee chair and thesis advisor, Professor Adam Kaplan who always steered me in the right direction; his consistent participation, input and help in the project work led to its successful completion. He has been a sincere advisor and an inspiring force throughout this thesis.

I would also like to express heartiest gratitude to my husband who has been a moral guide and a mentor for me throughout this work, and his invaluable support has given me confidence to pursue my work.

Lastly, I would like to thank my family and friends, especially my parents who have always been there with their constant love and blessings.

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

MedTrack: A Mobile Application for Patients, Physicians and Pharmacies; Track Your Med-Health on the go

By

Nishika Malhotra

Master of Science in Software Engineering

This thesis is an effective and stable implementation of a software package that is integrated with mobile computing devices utilizing Android operating system. It tries to solve a key challenge in patient healthcare by leveraging the power of mobile communication in the Information Technology oriented world whilst exploring the advantages of cloud technology. It intends to be a step towards the health informatics revolutionizing patient health and healthcare systems worldwide.

The prototype developed as a part of this thesis offers an easy to use graphical user interface to its users who can be either patients, doctors or pharmacies. Application users can register themselves on the app and use its computing power to keep track of their medication regime if they are patients, or track patient's health if they are doctors, or register with the application as a pharmacy to get refill orders from the patients. The application allow patients to scan their medicines and activate system alarms based on the dosage schedule fed in by the user. The application sends real-time notifications to the patients on their android mobile devices, helping them to keep track of their medicine intake. It also integrates with the Walgreen pharmacy through their Prescription API to allow patients to order prescription refills on the go. The Walgreen Prescription API is used as a ‘proof-of-concept’ in the application. All the medical data registered with the application is stored securely in the AWS cloud using DynamoDB as the database.

CHAPTER 1

INTRODUCTION

This thesis presents the implementation of a software application to monitor and track health regime of a subject via a mobile platform. The presented prototype application allow patients to enter their medical dosages and scan the prescribed medicines, thereby, activating system alarms for medicine intake reminders. Alarm notifications are sent to the patient's mobile device for on-time medicine intake based on the dosage schedule. The application also allows patients to order prescription refills to the registered pharmacies. The presented infrastructure has the potential to accelerate beneficial changes taking place in the interactions among patients, healthcare professionals, and patients' own health practices. With such an infrastructure, individuals can obtain the necessary information to make desirable behavioral changes and thereby achieve healthier lifestyle choices.

There is a big need for an integrated IT platform where patient health can be tracked and monitored through the mobile world anytime anywhere [1]. Evolving health informatics systems are set to revolutionize health and healthcare programs worldwide. However, turning this vision into reality will take enormous effort from thousands of designers, analysts, software engineers, and medical professionals [2]. In the current global medical scenario, as new diseases emerge at an accelerating rate, the number of patients and their illnesses continue to grow. An increasing number of patients must juggle a number of different medicines prescribed for use multiple times per day. The more medicines patients take, the more difficult it can be for them to remember the medicine dosage and it's routine. It would be helpful to have a convenient place where patients can keep track of all their medicines, a schedule of their dosages, and timely refill reminders to get better results. There is also much information that is needed by a physician on a daily basis to keep track of a patient's health. The physicians can receive up-to-date information about any delays in medicine intake or missed dosages, and can send pharmacists orders for refills and prescriptions on the go. This can result in a large shift in the locus of action and control, from the physician's office to the patient's home. This shift will result in more individualized healthcare when medical care is necessary.

Due to technology advancements over time, personal health monitoring devices and systems have gained popularity and are easily accessible to consumers. Mobile devices such as smart phones are ideal candidates for a solution that could help patients and physicians. There has been an explosion of mobile health activities around the world [3]. With smartphones accessible to every individual globally, their ample processing power and wireless capability could be combined with secure data storage capacity and real time monitoring to develop a cloud application for the medical industry that will integrate patients, physicians and pharmacists. This application aims to strike this cause and track a patient's dosage routine, display the time and quantity of medicine to be taken via timely alarms and notifications, and transmit the data securely to a central server in the cloud. The data, once transmitted, will remain secure and accessible to the physician who could conveniently check their patient's routine and dosage levels and inform pharmacies of new orders and refills. Taking full advantage of this available information, and its fusion with secure health monitoring data, there can be a significant impact on the prevention steps and individual lifestyle choices that patients make.

This thesis is organized into the following chapters. Chapter 2 describes the related work carried out by other researchers and software developers. Chapter 3 elaborates on the technical aspect of this system and its implementation. It describes the mobile software development process and the platform chosen for development. Chapter 4 describes the application’s architecture and design. Chapter 5 discusses the system functionality, a working prototype and use case scenarios for the application. Chapter 6 mentions various kinds of application testing and the AWS cloud based testing done for the application. The last chapter, i.e. Chapter 7 presents a conclusion of this project work.

CHAPTER 2

RELATED WORK

As the health industry is growing so fast, smartphone apps are being used to keep track of health information. There are a range of systems that use remote monitoring systems for healthcare applications. There are systems used for rehabilitation like MARHS [4] and those use for modifying diet like Myca Nutrition [5]. Other applications like Smart Shoe uses proprietary or in house sensors to monitor gait [6] and biomarkers for heart failure patients and other general health parameters [7], [8], [9].

There are numerous smartphone applications in the market that tracks patients' blood-glucose levels and blood-pressure readings, or fitness apps that track calorie intake and workout routines. Consistent Self- Monitoring Blood Glucose (SMBG) has been proven to be a useful tool in improving glycemic levels in type-2 diabetic patients [10]. However, there is no app that caters to the need of patient's dosage routine and brings the patient, the physician and the pharmacy at the same platform. Barcode readers are being used in weight-loss apps such as 'Fooducate', that scans a product's barcode to pull out extra nutritional information and help the user make healthier food choices [11]. However barcode readers have not yet been fully utilized by medical applications. The application developed as a part of this thesis project attempts to utilize this technology, however, is different as it uses the barcode scanning ability of smartphones to scan medicines and keep track of their intake for patients.

While a smart medicine cabinet might check that the right medicines are moving on and off the shelves as scheduled, it can't tell if the patient is actually consuming them. The idea for this project came from digestible computer chips with built-in wireless transmitters developed by Proteus Biomedical, based in California [12]. These “ingestible event markers” get activated from the digestive fluids in the stomach, which then create ultralow power signals that can be picked up by a tiny recorder inserted under the skin or worn as a small adhesive skin patch. The recorder notes the date and time of the pill’s activation along with other information such as the type of drug, dose, and place of manufacture. It takes a snapshot of patient’s heart rate, respiratory rate, and other physiological measurements. The detector then sends the information to a server that combines it with data from other sensors, as well as information entered by the patient. The application developed in this thesis project is different as it is a mobile technology-based application and not an electronic device. It uses smartphones to track information like drug taken, dosage, time, and quantity of medicine remaining for the patients and then this information is used by physicians to track their patient’s health.

An intelligent Electronic Medical Record (EMR) application [13] exists for iPads and is used by physicians to easily get patient’s medical history, medical problems, and all the latest data since the patient’s last visit. This application, named Wand, is implemented as a web application using D3.js and Apache Cordova, and machine learning provided through an internal Web service API. The app can help doctors review the patient’s chart before examination, update patient history during a consultation, refill electronic prescriptions, while rounding in the hospital or review the next day’s appointments from home. Wand supports the physician’s workflow anywhere, and anytime. The concept of patient data collection and utilization of analytics and visualization to help physicians better understand their patient needs is used to develop the thesis project. However, the project is different from Wand as it is not an EMR application. However, it uses the patient data to track their medical routine. It is beneficial to physicians who can now keep a live check on their patients and their medications.

Time Station is another application [14] which is a barcode based time and attendance tracking app that runs on the iPhone. It allows employees to track attendance and time through a barcode scanning system. This is a cloud-based application, all the data remains on the secure cloud and this data can be used by managers to check on their employees whenever they desire. As it runs in the cloud there are no servers for users to maintain. This app inspired the idea to track patient medicines using barcodes. However, the thesis project is a medical app rather than a productivity app, and is an Android app rather than an iOS application.

A cloud based asset management application named EZ Office Inventory [15] tracks assets quickly and accurately using an Android or iOS device. Since it is a cloud based service, all the data is accessible at any time with extensive reporting through a web portal. Beyond capturing basic asset attributes and location data, it has a great feature to send reminders about these assets. Being in the cloud, it takes no time for any user to get this application up and running because of fast ramp-up time provided by the geographically distributed cloud networks having high elasticity, availability, load balancing, and efficient delivery chain. This thesis project leverages this idea that smartphones can be used to send reminders and notifications to patients, physicians and pharmacies. Barcodes can be used for tracking, and if they are integrated into the greater system they can also be used for services like notifications. The mobile application developed in this thesis is different from this asset management app as it is developed for the medical industry and the user entered data in combination with the data generated through barcode scans is used for patient health assessment and regulation.

Walgreens Pharmacy has an API open to developers that provides the utility to scan prescription bottles and send notification to Walgreen pharmacy for refill orders [16]. This API helps integrate the drug information and Refill by Scan feature into a mobile app to aid physicians and patients refill medicines in seconds. The Walgreen Prescription API is used in this thesis project as a ‘proof-of-concept’. If pharmacies open their platform to be integrated with health tracking applications used by the patients, it would solve a lot of problems that ill patients’ sometime face while getting their medicine prescriptions refilled in person at the pharmacy stores.

CHAPTER 3

TECHNICAL DETAILS

This chapter provides a broad overview of the technologies and computing resources selected for the project. The project utilizes Xamarin platform which helps to build native apps for multiple platforms on a shared C# code base. Android platform is integrated with C# .Net technology, and Cloud computing resources are used to provide the architectural backend for the work. The following description tries to establish the background for the technology used in this project.

3.1 Xamarin

Xamarin is a cross-platform framework with its own Integrated Development Environment, it works on C# within the .NET framework and allows to create native apps. Xamarin apps are built with standard, native user interface controls [17]. Apps not only look the way the end user expects, they behave that way too. Xamarin apps leverage platform-specific hardware acceleration, and are compiled for native performance. This can't be achieved with solutions that interpret code at runtime. With a C# shared codebase, developers can use Xamarin tools to write native Android, iOS, and Windows apps with native user interfaces and share code across multiple platforms. Application can be built with either Visual Studio or Xamarin Studio. Xamarin integrates with Visual Studio, Microsoft's IDE for the .NET Framework, and extending Visual Studio for Android and iOS development.

3.2 Android Platform

This is a customizable easy to use Operating System, a Linux kernel mobile platform. The Android mobile operating system is dependent upon the mobile device's processer capabilities for its performance. There are a few platforms available for the mobile environment: Android, iOS, Rim (Blackberry), Windows Mobile, and others. According to the NPD research firm, the leading North American market research company, the Android platform has the largest market share (53 percent) in the US [18].

3.3 Xamarin For Android

Xamarin.Android uses just-in-time compilation for sophisticated runtime optimization of an application's performance, meaning that the app is a native Android APK. It allows access to any Android API, including new form factors- supports 100% of Google's Android APIs in C#, enhances Java APIs with async support and .NET naming convention making coding relatively easier. It allows calls to existing Java code from C#. Xamarin stays up-to-date with the most current APIs from Google. Xamarin.Android (formerly known as Mono for Android), initially developed by Novell and continued by Xamarin, is a proprietary implementation of Mono for Android-based smart-phones. The Xamarin.Android stack consists of the following components [19]:

- Mono runtime
- An Android UI designer
- Libraries: Core .NET class libraries and libraries that bind the native Android/Java APIs
- SDK tools to package, deploy and debug

3.4 C#

C# as the language of choice for mobile app development is an efficient language for mobile app development. Asynchronous programming (async) keeps apps responsive. C# uses type inference to give developers more safety in fewer keystrokes, without boilerplate or verbose type annotations. Generics guarantee that collections and other compound types are used safely, without the need for casts or comments.

3.5 Xamarin for Visual Studio

Xamarin allows for native Android, iOS and Windows app development within Microsoft Visual Studio. Xamarin supplies add-ins to Microsoft Visual Studio that allows developers to build Android, iOS, and Windows apps within the IDE using code completion and IntelliSense. It also has extensions within Visual Studio that provide support for the building, deploying, and debugging of apps on a simulator or a device.

3.6 Amazon Web Services DynamoDB

AWS DynamoDb is a fast and NoSQL database which provides consistency and single digit millisecond latency at any scale. It is a flexible and a fully managed cloud database which supports both document and key-value store models. As the data volume grows and application demands performance increase, Amazon DynamoDb uses automatic partitioning and SSD technologies, automatically spreads the data and traffic for the tables over a sufficient number of servers; meeting the throughput requirements, maintaining consistency, and delivering low latency and high performance at any scale [20].

DynamoDB stores information as database tables, which are collections of individual items. Each item is a collection of data attributes. The items are analogous to rows in a spreadsheet, and the attributes are analogous to columns. Each item is uniquely identified by a primary key, which is composed of two attributes, called the hash and range. DynamoDB queries refer to the hash and range attributes of items to access. Advanced Key-value data stores such as DynamoDB achieve high scalability on loosely coupled clusters by using the primary key as the partitioning key to distribute data across nodes. The query capabilities are based on the default primary index and optional local secondary indexes of a DynamoDB table:

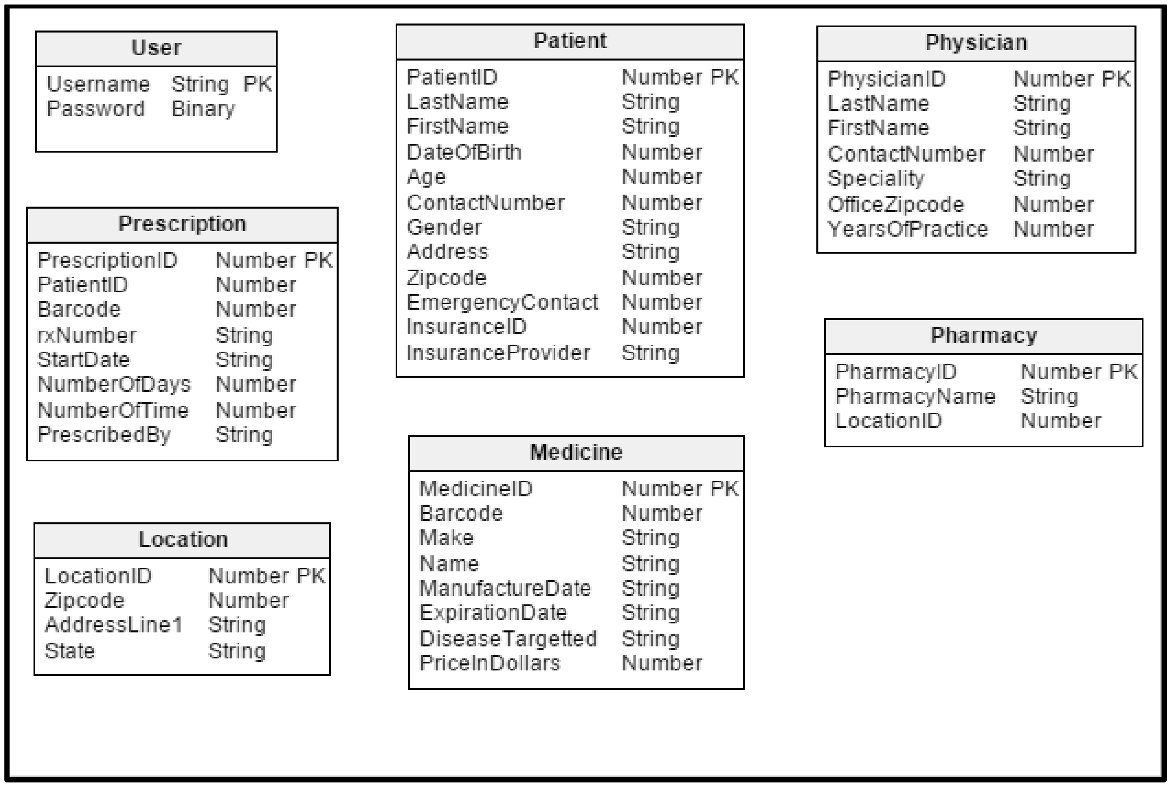
1. Primary Index: Two types of keys can be used for primary index querying: Simple Hash Keys and Composite Hash Key / Range Keys. Simple Hash Key gives DynamoDB the Distributed Hash Table abstraction. The key is hashed over the different partitions to optimize workload distribution. Composite Hash Key with Range Key allows to create a primary key that is the composite of two attributes, a “hash attribute” and a “range attribute.” When querying against a composite key, the hash attribute needs to be uniquely matched but a range operation can be specified for the range attribute.
2. Local Secondary Index: Local Secondary Indexes allows to create indexes on non-primary key attributes and quickly retrieve records within a hash partition (i.e., items that share the same hash value in their primary key).

Global secondary indexes allow to efficiently query over the whole DynamoDB table, not just within a partition as local secondary indexes, using any attributes (columns), even as the DynamoDB table horizontally scales. [21]

There’s an important distinction between a “query” and a “scan” in DynamoDB. A query is a search by ID. All query-searches without the ID parameter and a search by other parameters results in an exception. Searches based on parameters other than the ID are scan operations. An ID-based query is a lot faster as each record is guaranteed to have non-null ID field(s) and the IDs are indexed. As there's no enforced schema in DynamoDb a search based on any property must inspect – or scan – each record and check if it has that property [22]. A single Query or Scan request can retrieve a maximum of 1 MB of data; DynamoDB can optionally apply a filter expression to this data, narrowing the results before they are returned to the user [20].

3.7 Table Schema

The application's backend is built in DynamoDb using the AWS Management Console. There are seven tables in the database –User, Patient, Pharmacy, Physician, Medicine, Prescription, and Location. Figure 1 displays the DynamoDB table schema.



User		
Username	String	PK
Password	Binary	

Patient		
PatientID	Number	PK
LastName	String	
FirstName	String	
DateOfBirth	Number	
Age	Number	
ContactNumber	Number	
Gender	String	
Address	String	
Zipcode	Number	
EmergencyContact	Number	
InsuranceID	Number	
InsuranceProvider	String	

Physician		
PhysicianID	Number	PK
LastName	String	
FirstName	String	
ContactNumber	Number	
Speciality	String	
OfficeZipcode	Number	
YearsOfPractice	Number	

Prescription		
PrescriptionID	Number	PK
PatientID	Number	
Barcode	Number	
rxNumber	String	
StartDate	String	
NumberOfDays	Number	
NumberOfTime	Number	
PrescribedBy	String	

Pharmacy		
PharmacyID	Number	PK
PharmacyName	String	
LocationID	Number	

Medicine		
MedicineID	Number	PK
Barcode	Number	
Make	String	
Name	String	
ManufactureDate	String	
ExpirationDate	String	
DiseaseTargetted	String	
PriceInDollars	Number	

Location		
LocationID	Number	PK
Zipcode	Number	
AddressLine1	String	
State	String	

Figure 1 DynamoDB Table Schema for MedTrack

User table holds the username and password of every new user who registers with the MedTrack application. The password is stored in the hashed format using the SHA1 Cryptographic algorithm. The Username is the primary partition key for this table.

Patient table holds the profile of every registered user who adds himself as a Patient with the application. It auto generates the PatientID which is unique for every patient and is the primary partition key for the table, and LastName is the primary sort key for the table.

Physician table holds the profile of every registered user who adds himself as a Physician with the application. It auto generates the PhysicianID which is unique for every physician and is the primary partition key for the table, and LastName is the primary sort key for the table.

Pharmacy table holds the profile of every pharmacy registered with the application. Pharmacies can register their names and location including zip code, address and state with the application. The location details get added to the location table and the LocationID returned by the application for the specific address gets added to the specific entry in the Pharmacy table as a LocationID.

Location table holds the location addresses for pharmacies with their Address, State and Zipcode.

Medicine table holds the information about the prescribed medicines by the physicians. It has the medicine name, make, manufacture date, expiration date, disease targeted, the medicine barcode and the price in dollars. Each medicine when added to the table is allotted a MedicineID.

Prescription table in the database has all the prescriptions that are added by patients or physicians to track medicine regime. The table has the prescription’s Barcode, rxNumber, StartDate, NumberOfDays, NumberOfTime the prescription is to be taken, patientID of the patient, and PrescribedBy field representing the physician’s name who prescribed that prescription.

CHAPTER 4

SYSTEM ARCHITECTURE AND DESIGN

MedTrack is a mobile application that runs on Android Operating System based mobile devices. It is built with the minimum Android SDK version of 15 and the traget Android SDK version of 23. The application needs access permissions to the running system's camera, internet and wake lock to carry the necessary functionalities.

4.1 System Architecture

Medtrack mobile application is based on the three tier software architecture which comprises of the Presentation tier, the Business logic tier and the Database tier. Application's presentation tier is made up of the Android views, this is the user's interaction interface with the application where the user can input their data and receive messages or notifications from the application in real time. The Business tier comprises of the request processing and business logic of the application that is based on the C# .Net classes, libraries like ZXing for Barcode scanning, and the web service HTTP and JSON calls from the .Net code to the Walgreen Refill Prescription API based on the user's request. The Database tier comprises of the DynamoDB which is a NoSQL database on the Amazon Web Services cloud.

As the Figure 2 implies, in this system, client interacts with the Adroid presentation layer, inputs the data, and the control then passes on to the business layer which processes the user input, handles the request and sends it to the third party api if necessary and based on the response sends back the information and the control to the user. The data is saved on the cloud using AWS DynamoDb.

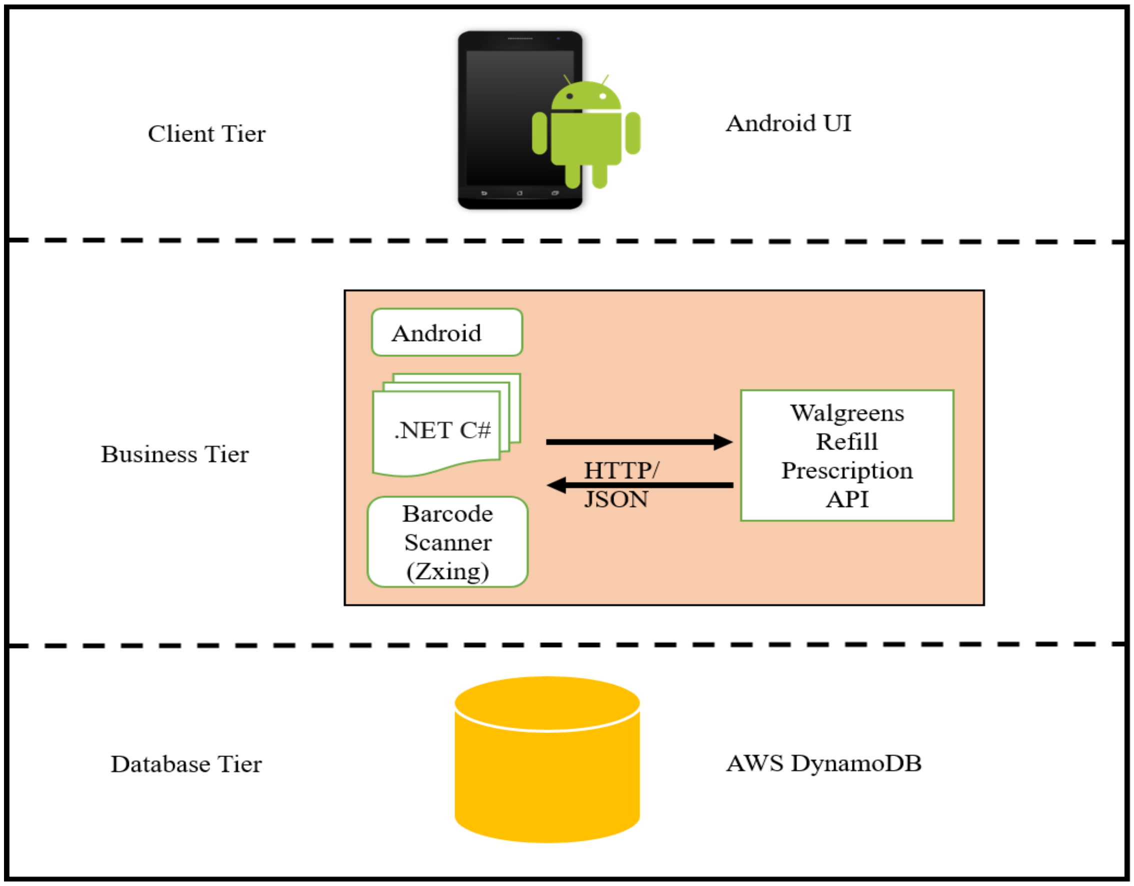


Figure 2 Software System Architecture

4.2 System Design

MedTrack application consists of the Entity classes –User, Location, Medicine, Prescription, Patient, Pharmacy and Physician, all of which directly correlate to the respective tables in the DynamoDb and the columns in the tables correspond to the fields of these entity classes. The screen displayed to the user on the Android device is built using the Layout .AXML files which accept the user input. The user entered data values are mapped from the presentation UI layer using the Android activity classes which define the presentation of the content on the screen. The Activity classes fetch the data entered in these layout screens, populate the values to the rescpective DynamoDB data table fields, and instantiate the Library classes that make a connection with the AWS DynamoDb to save the user data. There is one library class specific to each database table that has the logic to interact with the table and carry out the functionalities like add a new entity into the respective table. The DynamoDB client is configured from the library class with the Access Key, the Secret Key and the AWS region which then points to the database on the AWS cloud in that specific region.

Barcode scanning ability of this application comes from the integration of a third party barcode scanning library, ZXing.Net.Mobile. ZXing is an open-source, multi-format 1D/2D barcode image processing library implemented in Java, with ports to other languages like C# .Net. This library supports all the UPC-A, UPC-E, EAN-8, and EAN-13 barcodes which are mostly used in the medical industry on the drug prescriptions.

Broadcast Receiver is used to allow the application to interact with the Android system device and set up alarm notifications in realtime. Broadcast Reciever uses Observer Pattern to send off alarms.

WebView Activity of the Android development is used in this application to interact with the Walgreen Refill Prescription API. This activity sets off a HTTP request to the Walgreen URI with the required API key for authentication and other request parameters to launch their web view. HTTPWebRequest and HTTPWebResponse classes of the System.Net framework are used for HTTP communication, and the request and response objects are serialized and deserialized using the Newtonsoft.JSON library for C#.Net.

The reference libraries used for the development of this application are as below:

1. AWSSDK_Android : is a library that incorporates an open source AWS Software Development Kit distributed under an Apache Open Source license to build

Android applications with ease. It provides an easy access to a range of AWS components like DynamoDB, Mobile Analytics, Auto Scaling, S3, Amazon Lambda and more. It includes client side class libraries for working with the Amazon Web Services. This hides most of the lower-level coding for the web service interface communication, including authentication, request retries, and error handling, allowing the developer to establish a connection with the AWS cloud interface with ease.

2. Mono.Android: is a library provided by Xamarin to create .NET appliations that can run cross- platform, it provides a comprehensive set of facilities for application development. Xamarin.Android (formally known as Mono for Android) is an implemenation of Mono for Android based smartphones [19].
3. Newtonsoft.Json: is a high performance framework library for .NET. It provides a flexible JSON serializer for converting between .NET objects and JSON and JSON deserializer for converting JSON objects to .NET objects.
4. ZXing.Net.Mobile: is a C# .NET library based on the open source barcode library known as ZXing (Zebra Crossing), using the ZXing.Net Port. It works with Xamarin.iOS, Xamarin.Android, and Windows Phone.

CHAPTER 5

SYSTEM FUNCTIONALITY, PROTOTYPE AND USE CASES

Programming for this Android operating system based application is done using the Visual Studio 2015 Integrated Development Environment with the Xamarin plugin installed, and C# .Net as the coding language framework.

5.1 System Functionality

The system functionality is clearly depicted in the Figure 3. The MedTrack application starts when a user launches the application on an Android smartphone device, the welcome screen appears with the options to either ‘login’ an existing user or ‘register’ a new user to the application. Based on the option selected the respective android layout views appear on the application. If the user selects the ‘Register’ button, New User Registration screen appears with the username and password fields, the user enter the values of choice and clicks the ‘Register’ button. On successful registration the data is saved into the User table in the DynamoDB and the user is shown a success message, else error message is displayed. Successful user registration navigates to the Profile Registration screen with options to add a patient profile, a physician profile or a pharmacy profile. Clicking each of these buttons opens respective profile views with related fields to enter. This completes the registration and profile creation. The data entered in the Patient, Physician and Pharmacy profiles is saved to the Patient, Physician and the Pharmacy tables in the database respectively.

When a user clicks the ‘Log In’ button from the welcome screen, enters the username and password fields, and then clicks the ‘Login’ button, the application authenticates the credentials from the database and opens the main application screen with options: Add a

Prescription, Refill a Prescription, and Set Alarms to track the prescriptions. Clicking 'Add a Prescription' button opens the screen where the user is required to enter the details of his prescriptions that gets saved in the Prescription table in the database. Clicking 'Refill a Prescription' button opens the screen to enter the patient's id and scan the prescription's barcode that needs to be refilled. Once the barcode is scanned, the application finds the rxNumber corresponding to this drug and then takes the control to the Walgreen's refill prescription application view where a user can submit the refill request to the Walgreen pharmacy. Clicking on the 'Set Alarm' button sets the alarms for the user's prescription drugs and sends repeated timely notifications to the Android device for medicine intakes. It also prompts the user to scan the medicine's barcode to ensure that the user has taken the medicine and reflect the same in the database.

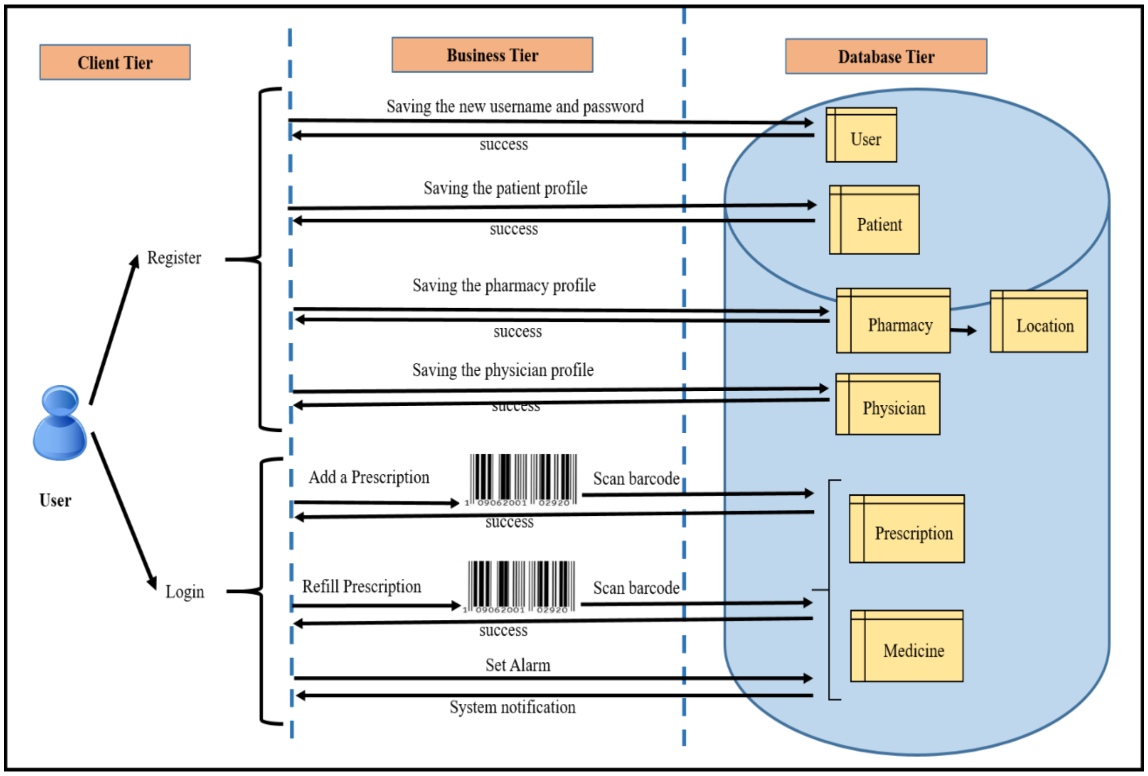


Figure 3 Software System Functionality

5.2 System Users

The MedTrack application user can either be a Patient, a Physician or a Pharmacy. The functionality of the application depends on the type of user and the purpose of use of the application. A typical use case scenario is discussed and represented in the Figure 4.

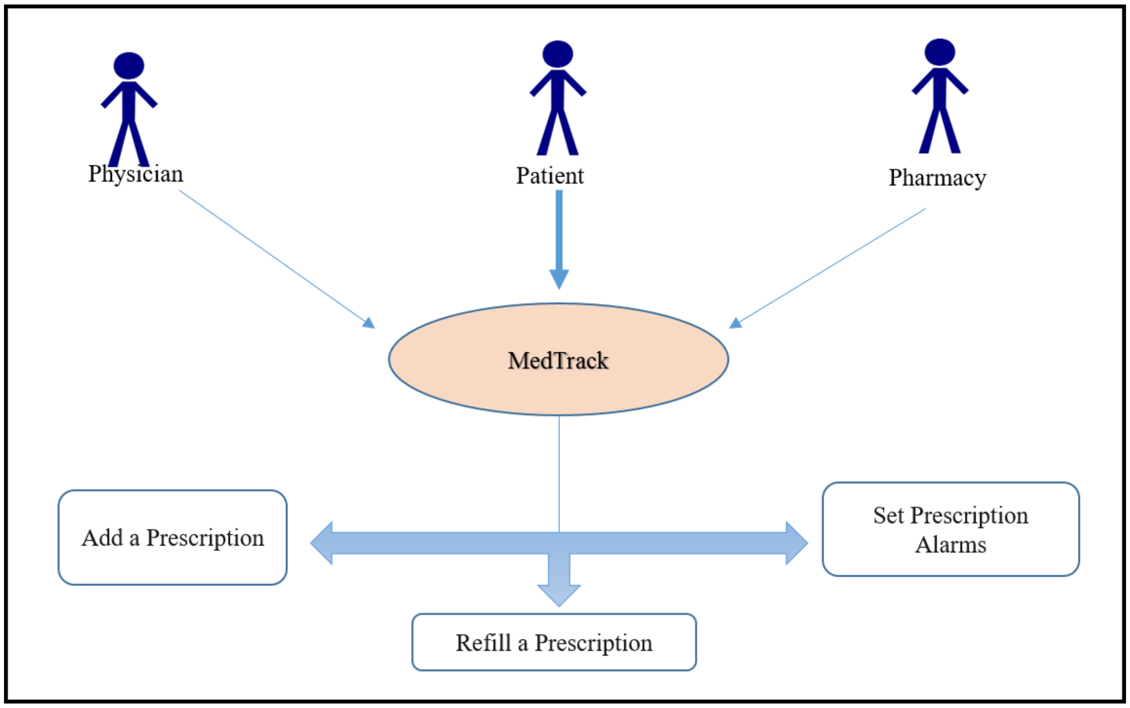


Figure 4 MedTrack User Diagram

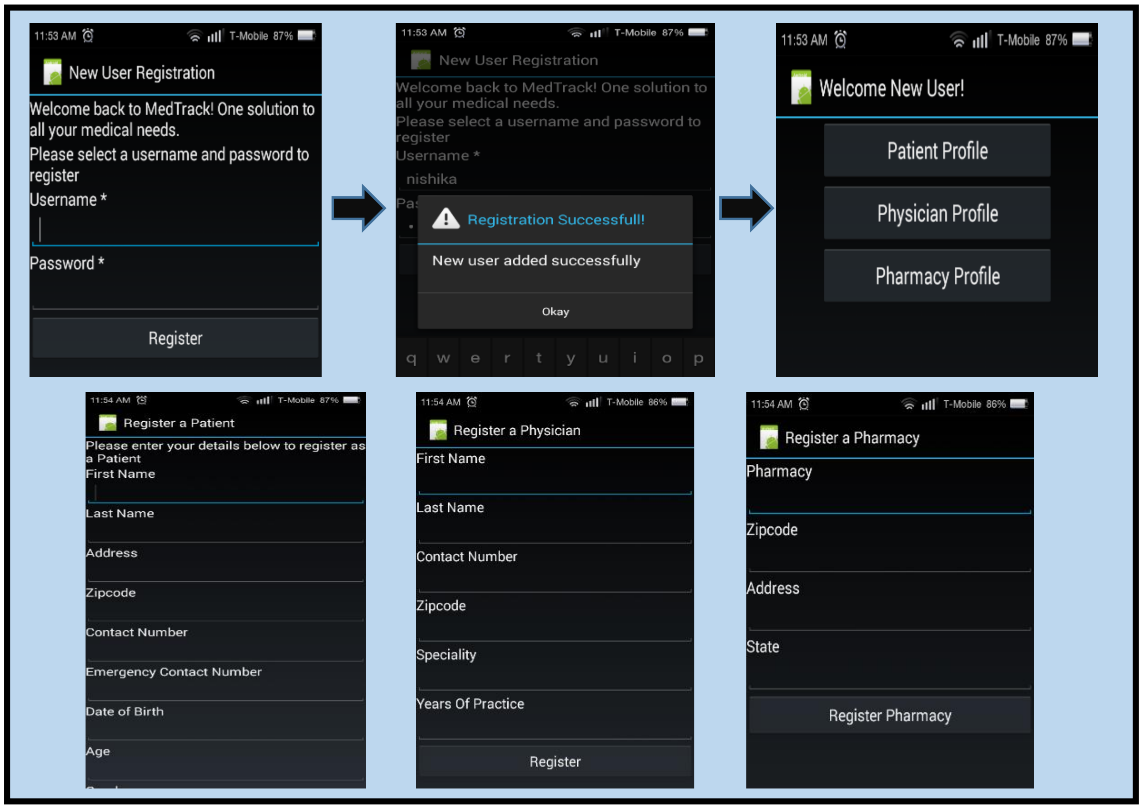
Patient: is the user of the application who adds his profile with the details like name, address, age, contact information, any existing diseases or disabilities, and insurance details.

Physician: is the user of the application who can add himself as a registered user to track his patient's medical regime.

Pharmacy: is the user of the application which if registered can tie up for online refill prescriptions from patients. Currently, Walgreen is the only available pharmacy for testing this functionality as they have their developer refill prescription API available as an open source API. Walgreen’s refill prescription API is used as a proof-of-concept in this project.

5.3 System Use Cases

1. Register as a New User: A new user registers on the MedTrack app by creating a new user account. The user enters the desired username and password and selects to Register. Then the user is prompted to enter the user profile as a Patient, or a Physician or a Pharmacy. The user selects either of the options and enters all the details asked by the application. The Figure 5 lists all the steps of this use case below.



Welcome back to MedTrack! One solution to all your medical needs.
Please select a username and password to register

Username * nishika

Registration Successfull!
New user added successfully

Please enter your details below to register as a Patient

Figure 5 Use Case Diagram- Register as a New User

2. Log In: An existing user or a registered user can login to the application and use its functionality. User enters the username and password and clicks the Login button.

If username or password is incorrect, an error message is displayed else if they are correct the user navigates to the welcome screen with options to Add a Prescription, Refill a Prescription or to Set Alarms for the Prescription. The Figure 6 depicts the flow of this use case.

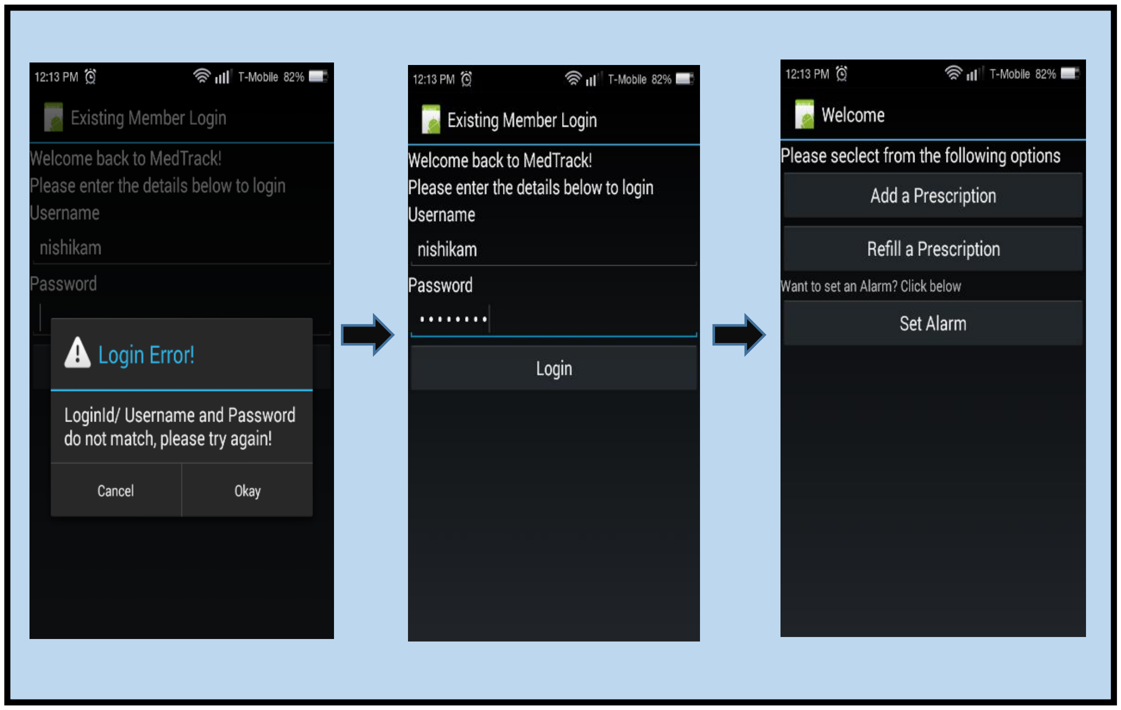
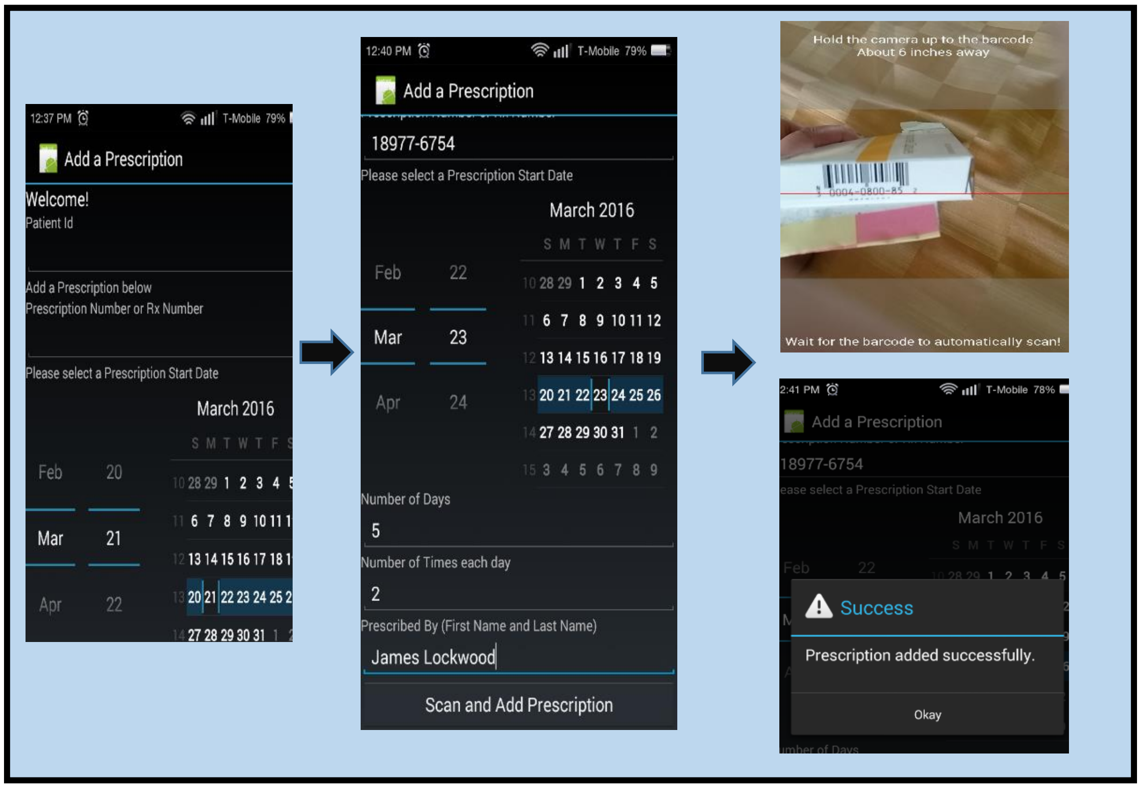

12:13 PM
T-Mobile 82%
Existing Member Login
Welcome back to MedTrack!
Please enter the details below to login
Username
nishikam
Password
Login Error!
LoginId/ Username and Password do not match, please try again!
Cancel
Okay
Login
Welcome
Please seclect from the following options
Add a Prescription
Refill a Prescription
Want to set an Alarm? Click below
Set Alarm

Figure 6 Use Case Diagram- Log In

3. Add a Prescription: A user logged in to the application clicks on the ‘Add a Prescription' `button on the Welcome screen. Add Prescription screen appears, user enters the required fields and clicks Scan and Add Prescription button on the screen. The Barcode scanner opens up, user scans the prescription barcode and the prescription details get added to the database. A success message is displayed to the user once the prescription is saved. Figure 7 clearly depicts the steps of this use case.


12:37 PM T-Mobile 79%

Add a Prescription

Welcome!

Patient Id

Add a Prescription below

Prescription Number or Rx Number

Please select a Prescription Start Date

March 2016

Feb 20 | Mar 21 | Apr 22

12:40 PM T-Mobile 79%

Add a Prescription

18977-6754

Please select a Prescription Start Date

March 2016

Feb 22 | Mar 23 | Apr 24

Number of Days

5

Number of Times each day

2

Prescribed By (First Name and Last Name)

James Lockwood

Scan and Add Prescription

Hold the camera up to the barcode About 6 inches away

Wait for the barcode to automatically scan!

2:41 PM T-Mobile 78%

Add a Prescription

18977-6754

ease select a Prescription Start Date

March 2016

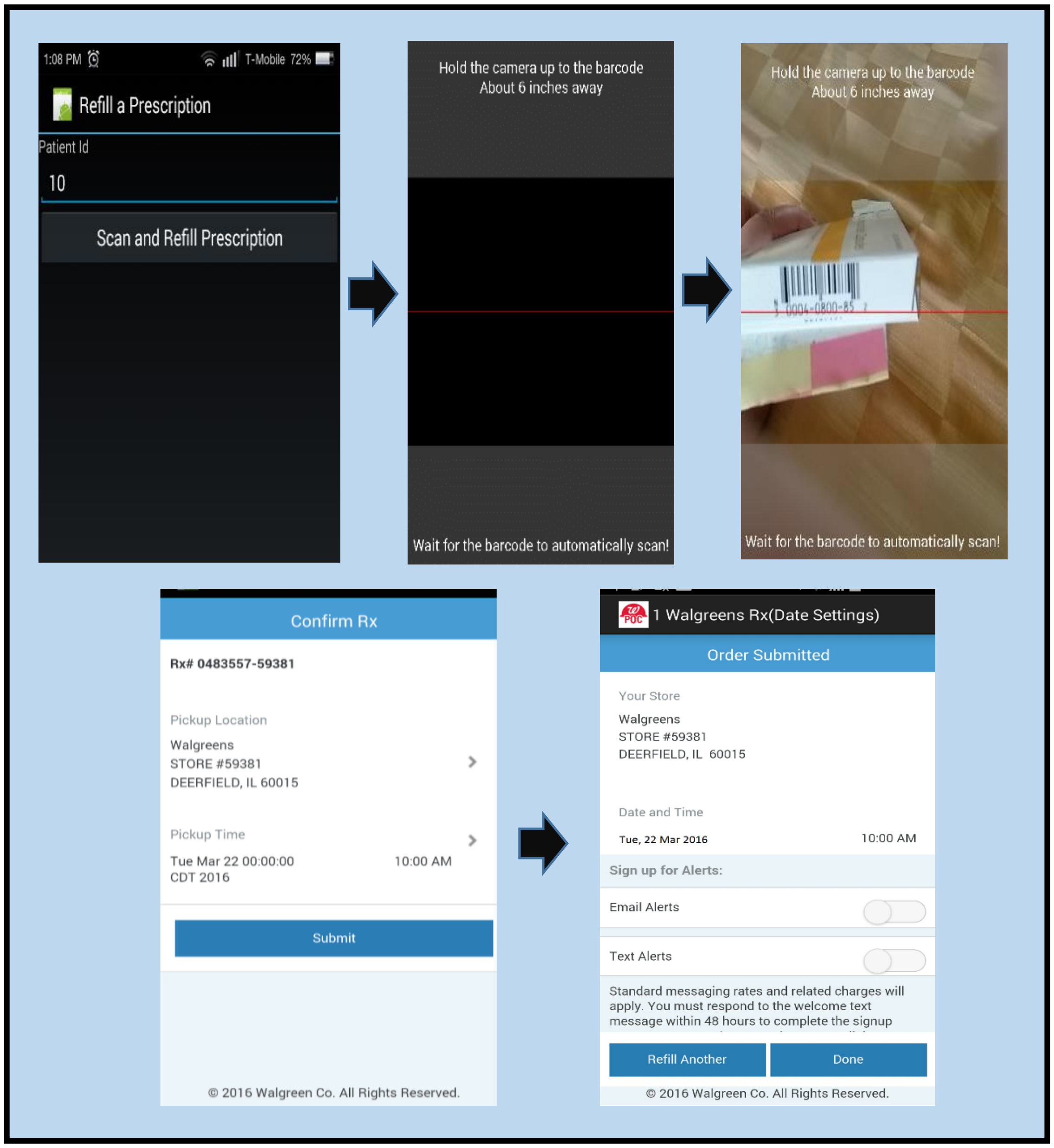
Success

Prescription added successfully.

Okay

Figure 7 Use Case Diagram- Add a Prescription

4. Refill a Prescription: A user logged in to the application clicks on the 'Refill a Prescription' button on the Welcome screen. Refill Prescription screen appears, user enters the Patient ID and clicks Scan and Refill Prescription button on the screen. The Barcode scanner opens up, user scans the prescription barcode that needs to be refilled and the prescription details along with the Rx number of the medicine goes to the Walgreen pharmacy registered with the application. The Walgreen Pharmacy web view opens up, user verifies the details and clicks Submit, and the refill order goes to the selected Walgreen location. The Figure 8 clearly depicts the steps of this use case.


1:08 PM T-Mobile 72%

Refill a Prescription

Patient Id

10

Scan and Refill Prescription

Hold the camera up to the barcode
About 6 inches away

Wait for the barcode to automatically scan!

Hold the camera up to the barcode
About 6 inches away

Wait for the barcode to automatically scan!

Confirm Rx

Rx# 0483557-59381

Pickup Location

Walgreens
STORE #59381
DEERFIELD, IL 60015

Pickup Time

Tue Mar 22 00:00:00 CDT 2016 10:00 AM

Submit

© 2016 Walgreen Co. All Rights Reserved.

1 Walgreens Rx(Date Settings)

Order Submitted

Your Store

Walgreens
STORE #59381
DEERFIELD, IL 60015

Date and Time

Tue, 22 Mar 2016 10:00 AM

Sign up for Alerts:

Email Alerts

Text Alerts

Standard messaging rates and related charges will apply. You must respond to the welcome text message within 48 hours to complete the signup

Refill Another

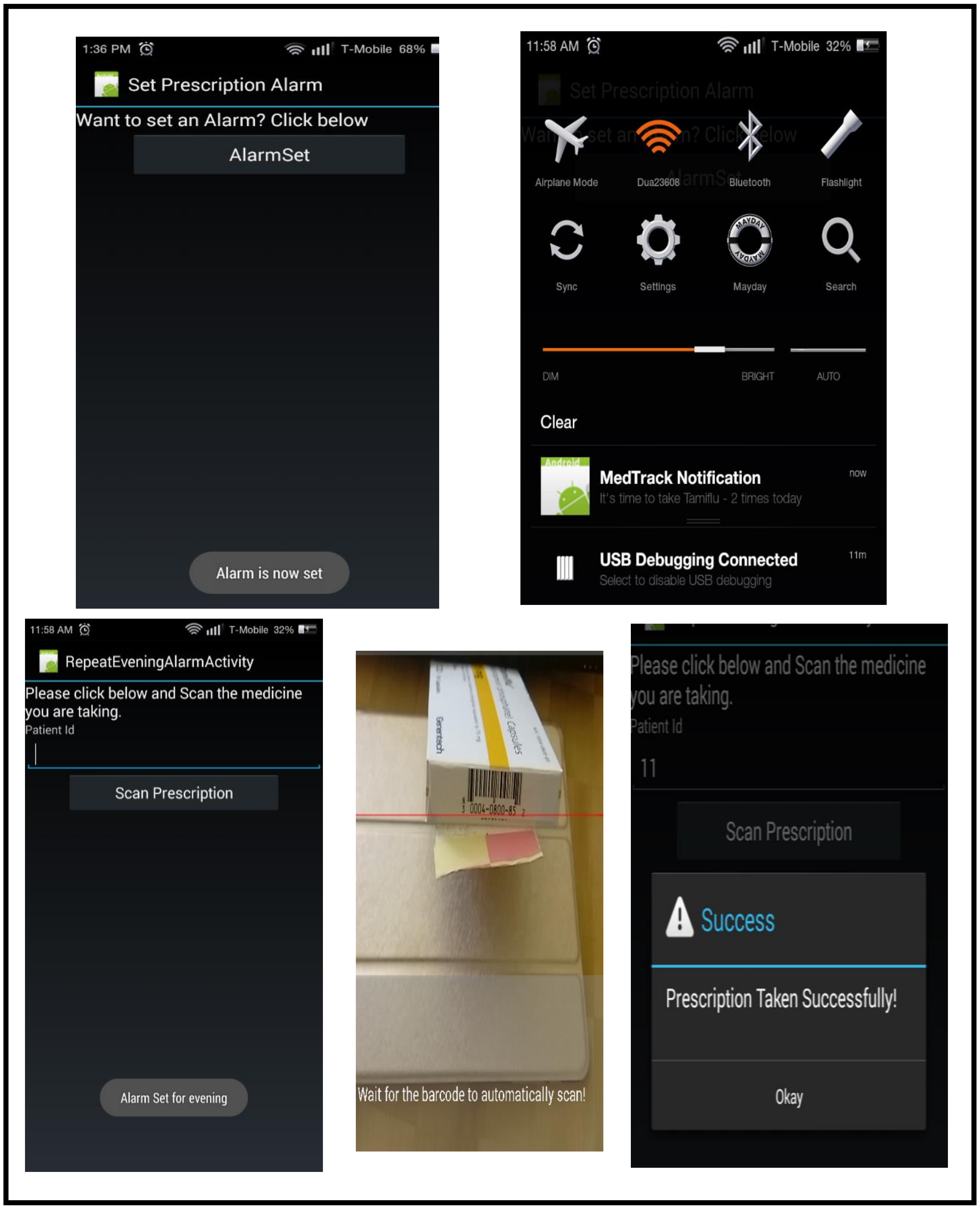
Done

© 2016 Walgreen Co. All Rights Reserved.

Figure 8 Use Case Diagram- Refill a Prescription

5. Set Alarm: A user logged in to the application clicks the ‘Set Alarm’ button from the Welcome screen, the Set Prescription Alarm screen opens up. User clicks the AlarmSet button and a message stating ‘Alarm is now set’ is displayed. The user receives system notifications for the alarm. Each time the alarm is set off the application prompts the user to scan the prescription barcode and take the medicine

at that time. User scans the prescription barcode and the alarm disappears. The Figure 9 clearly depicts the steps of this use case.


1:36 PM T-Mobile 68%

Set Prescription Alarm

Want to set an Alarm? Click below

AlarmSet

Alarm is now set

11:58 AM T-Mobile 32%

Clear

MedTrack Notification now

It's time to take Tamiflu - 2 times today

USB Debugging Connected 11m

Select to disable USB debugging

11:58 AM T-Mobile 32%

RepeatEveningAlarmActivity

Please click below and Scan the medicine you are taking.

Patient Id

Scan Prescription

Alarm Set for evening

Wait for the barcode to automatically scan!

Please click below and Scan the medicine you are taking.

Patient Id

11

Scan Prescription

Success

Prescription Taken Successfully!

Okay

Figure 9 Use Case Diagram- Set Alarm

5.4 System Activity Diagram

The application navigates through various activity views and states when the user first starts the application until the end. The android activity classes which render the User Interface on the Android device is represented by the respective Views in the Figure 10 below. These views are rendered as a result of an action triggered when the user interacts with the application in a certain way.

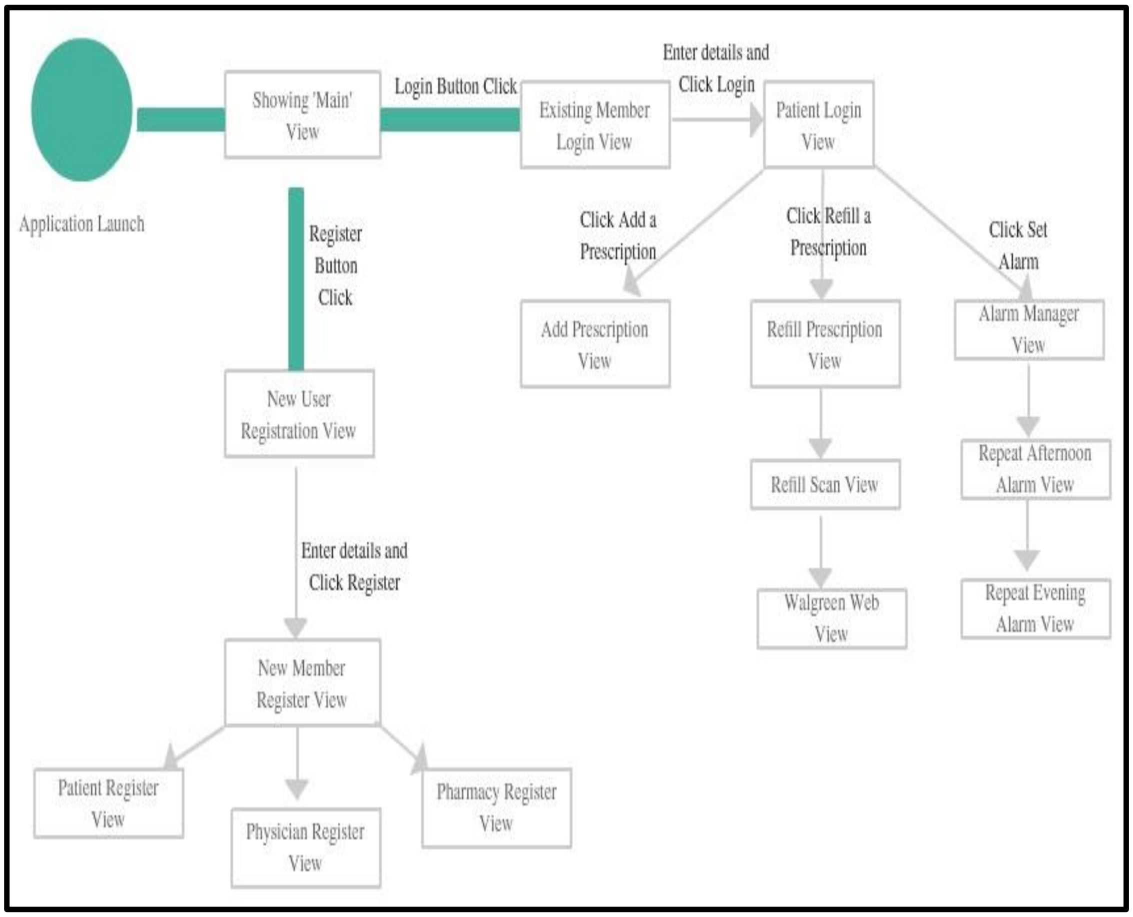


Figure 10 MedTrack Activity Diagram

5.5 System Prototype

The working prototype of the MedTrack application is presented as a video presentation of the application launch and navigation trough the various different screens of the application based on the user's interaction with the application.

5.6 System Performance

The client's software requirements are based on the general Android Operating System hardware requirements. MedTrack application is built for and executed on Android platform and thus does not mandate any special performance requirements. However, the minimum version of the Android platform on which the application will run is API level 15 that is Android 4.0.3 and 4.0.4 also known as ICE_CREAM_SANDWICH_MR1. The Android API level on which the application is designed to run without any restrictions is API level 23 that is Android 6.0-6.0.1 also known as MARSHMELLOW.

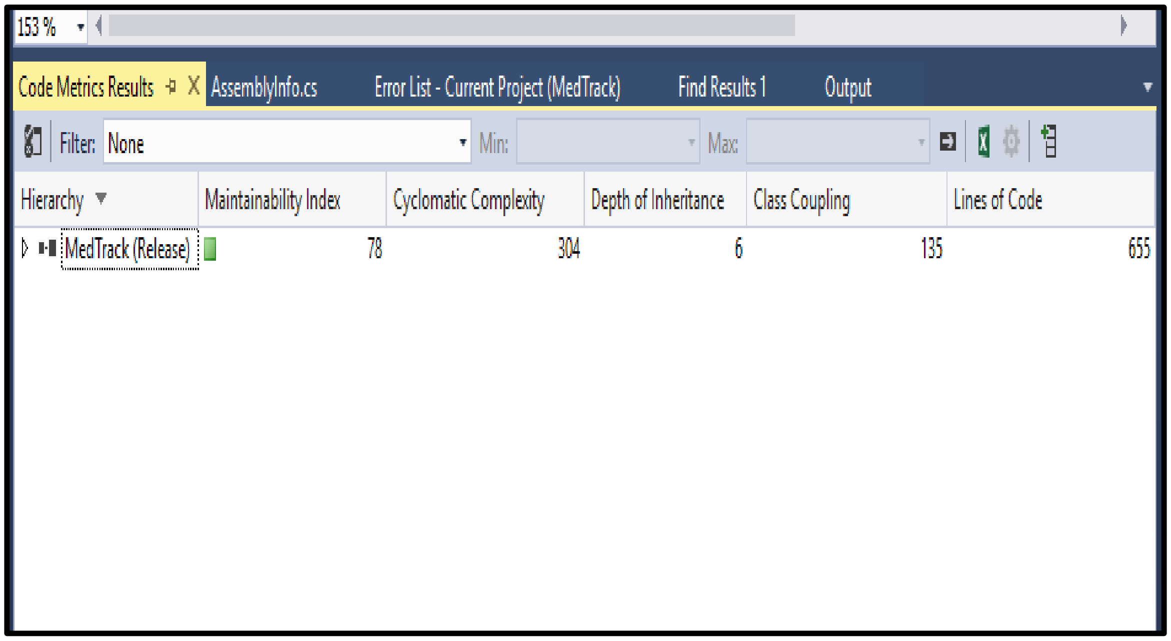
Other performance requirements of the smartphone device for best application functioning include a good high speed internet connection in the device, 3G is recommended, with 1 GB of disc space preferably, at least 700 MB is needed and a RAM of 64MB or more.

5.7 System Code Metrics

- Maintainability Index – This metric calculates an index value that represents the relative ease of maintaining the code. The value ranges between 0 and 100, where a high value means better maintainability. A green colored rating between 20 and 100 indicates that the code has good maintainability. This software shows a maintainability index of 78 as indicated in the Figure 11 below and this falls in range of easy to maintain code.

- Cyclomatic Complexity – This metric measures the structural complexity of the code. It is calculated from the number of different code paths in the flow of the program. A program that has a high value of cyclomatic complexity has a complex control flow and requires more tests to achieve good code coverage. This software implementation has a value of 304 as shown in the Figure 11 below.
- Depth of Inheritance – This indicates the number of class definitions that extend to the root of the class hierarchy. The deeper the hierarchy the more difficult it might be to understand where particular methods and fields are defined. This software implementation has a depth of inheritance metric value of 6 as shown in the Figure 11.
- Class Coupling – This code metric measures the coupling to unique classes through parameters, local variables, return types, method calls, generic or template instantiations, base classes, interface implementations, fields defined on external types, and attribute decoration. A good software design should have high cohesion and low coupling because high coupling indicates a design that is difficult to reuse and maintain because of its many interdependencies on other types. This metric value is shown in the Figure 11 below for this implementation.
- Lines of Code – This metric is used to find the approximate number of lines in the code. The count is based on the IL code and therefore does not tell the exact number of lines in the source code file. A very high count might indicate that the type or

method might be hard to maintain. This metric has a value of 655 for this software implementation and is shown in the Figure 11 below



Hierarchy	Maintainability Index	Cyclomatic Complexity	Depth of Inheritance	Class Coupling	Lines of Code
MedTrack (Release)	🟩 78	304	6	135	655

Figure 11 System Code Analysis

CHAPTER 6

APPLICATION TESTING

This section focusses on outlining the major testing procedures performed during and after the application development to ensure that the software functions and performs as required.

6.1 Unit Testing

This type of testing is the fundamental testing in the application development lifecycle. It is done to ensure each unit or module of the software functions as desired, and that the fundamental logic of individual units is correct. Running unit tests after every build helps to quickly catch and fix software regression errors introduced by code changes to the app.

6.2 Interface Testing

This testing is done to ensure that a specific module performs efficiently and effectively outside the application boundary with all interfaces specific to this module. This is assessed for all the interfaces of the application.

6.3 Performance Testing

Testing done to ensure that that the application and every module within it performs to industry standard expectations in terms of response time, availability, portability, and scalability.

6.4 Regression Testing

Testing done to ensure that applied changes to the modules and shared classes have not adversely affected previously tested functionality. Regression testing is performed after introduction of each new module, using permanent as well as temporary interfaces.

6.5 Integration Testing

Testing in which individual software module, software and hardware, and hardware elements are combined and tested as an integrated system. The purpose of integration testing is to ensure that all the non-functional design objectives are met and it ensures that the software, as a complete entity, complies with operational requirements.

6.6 User Acceptance Testing

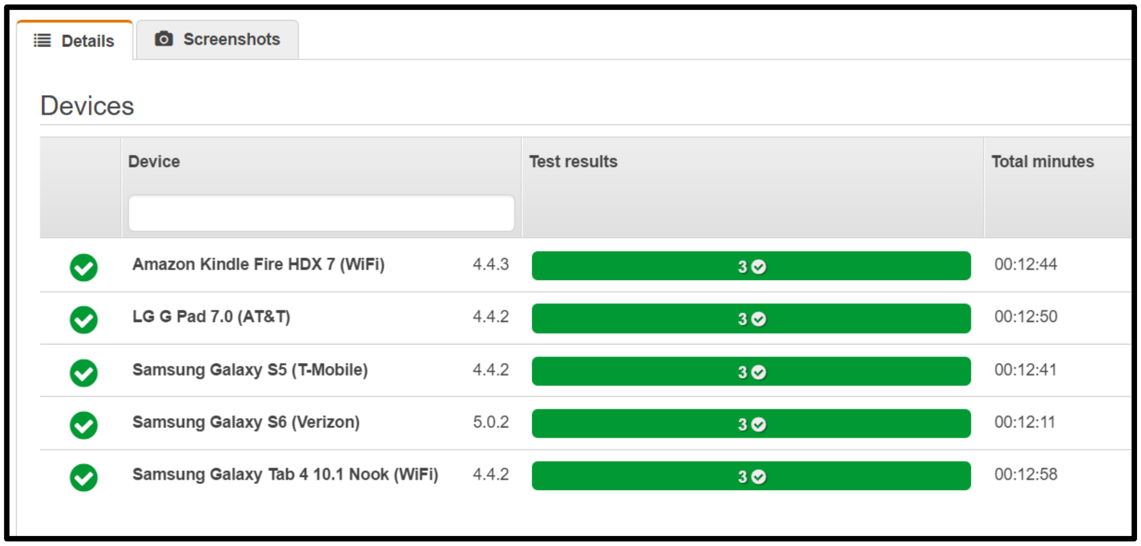
Testing done to determine whether or not a system application satisfies the user expectations based on the acceptance criteria set for the application. Acceptance testing certifies that contractual requirements and key objectives are met and that all components are correctly integrated in a comprehensive software package.

6.6 Amazon Web Service Device Farm

AWS Device Farm is used to test the application on a real smartphone Android device on the cloud. It is an app testing service to test android, iOS and Web apps on real devices. Device Farm supports native and hybrid Android, iOS, and Fire OS apps, including those created with PhoneGap, Titanium, Xamarin, Unity, and other frameworks [20].

Device Farm allows developers to upload applications on to the AWS cloud through the AWS Management Console. The developers choose which devices they want to test their app on and the device farm service automatically iterate through all the application screens and buttons. Later, this service generates a report for the developer to asses. This test report containing high-level results, low-level logs, pixel-to-pixel screenshots, and performance data is updated as tests are completed [23].

MedTrack is tested on the Device Farm on 5 Android OS based smartphone devices: Amazon Kindle Fire HDX 7, LG G Pad 7.0, Samsung Galaxy S5, Samsung Galaxy S6, and Samsung Galaxy Tab 4. All the tests in the Device Farm passes successfully. The Figure 12 shows the test results clearly.



Devices

	Device		Test results	Total minutes
✓	Amazon Kindle Fire HDX 7 (WiFi)	4.4.3	3 ✓	00:12:44
✓	LG G Pad 7.0 (AT&T)	4.4.2	3 ✓	00:12:50
✓	Samsung Galaxy S5 (T-Mobile)	4.4.2	3 ✓	00:12:41
✓	Samsung Galaxy S6 (Verizon)	5.0.2	3 ✓	00:12:11
✓	Samsung Galaxy Tab 4 10.1 Nook (WiFi)	4.4.2	3 ✓	00:12:58

Figure 12 AWS Device Farm Test Results

CHAPTER 7

CONCLUSION

Online and electronic patient tools like medical web-portals, mobile applications and wear gears facilitate communication and exchange of time sensitive data between patients, physicians and healthcare organizations. Online medical facilities like scheduling and reminders help patients to arrange and access health care that suits their needs and preferences, virtual consultation enables them to receive advice and health guidance from physicians and other providers regardless of the physical location, thereby improve healthcare operations. In this thesis the application and effectiveness of Information Technology in medicine has been demonstrated through a mobile application.

The application presented in this thesis allow patients to follow their prescribed health regime in a timely manner and also schedule their prescription refills with a pharmacy based on their convenience. The application’s software and mobile platform being Xamarin for Android allows easy access to Android API through C# libraries, provides runtime optimization enhancing application performance and.NET naming convention makes coding easier. Three tier system architecture used in this application allows separation of concern of the presentation, business logic and data layer. The system design provides an easy to use interface and easily maintainable functionality for all the use cases of the application. The application has been successfully tested in various android devices as well as on the cloud using Amazon Web Services. This mobile application can be used as a prototype to develop a fully integrated medical application in future.

REFERENCES

[1] mHealth: mobile technology poised to enable a new era in health care, *Ernst & Young*, © 2012 EYGM Limited.

[2] Shneiderman, B., Plaisant, C., and Hesse, B. W. (2013). Improving healthcare with interactive visualization. *Computer,* 46(5), 58-66.

[3] West, D. (2012). How mobile devices are transforming healthcare. *Issues in technology innovation*, 18(1), 1-11.

[4] Lee, S. I., Woodbridge, J., Nahapetian A., and Sarrafzadeh, M. MARHS: Mobility Assessment System with Remote Healthcare Functionality for Movement Disorders. *ACM SIGHIT* International Health Informatics Symposium (IHI), January 2012.

[5] Dorman, K., Yahyanejad, M., Nahapetian, A., & Sarrafzadeh, M., McCarthy, W., and Kaiser, W. Nutrition Monitor: A Food Purchase and Consumption Monitoring Mobile System. Conference on Mobile Computing, Applications, and Services (MobiCASE), October 2009.

[6] Noshadi, H., Ahmadian, S., Dabiri, F., Nahapetian, A., Stathopoulus, T., Batalin, M., Kaiser, W., and Sarrafzadeh, M. Smart Shoe for Balance, Fall Risk Assessment and Applications in Wireless Health. Microsoft eScience Workshop, December 2008.

[7] Suh, M., Chen, C., Woodbridge, J., Tu, M., K., Kim, J., I., Nahapetian, A., Evangelista, L., S., and Sarrafzadeh, M. A Remote Patient Monitoring System for Congestive Heart Failure. Springer Journal of Medical Systems (JOMS), Vol 35, No. 5, pp. 1165-1179, June 2011.

[8] Wu, W.H., Bui, A.A.T., Batalin, M.A., Au, L.K.,Binney, J.D., and Kaiser, W. J. MEDIC: Medical Embedded Device for Individualized Care, Artificial Intelligence in Medicine, Vol. 42, issue 2, pp 137-152, Feb,2008.

[9] Wu, W.H., Bui, A.A.T., Batalin, M.A., Liu, D., and Kaiser, W. J. Incremental Diagnosis Method for Intelligent Wearable Sensor Systems, IEEE Transactions on Information Technology in Biomedicine, vol.11, no.5, pp.553-562, Sept. 2007.

[10] Tran, J., Tran, R., and White, J. R. (2012). Smartphone-based glucose monitors and applications in the management of diabetes: an overview of 10 salient "apps" and a novel smartphone-connected blood glucose monitor. *Clinical Diabetes*, *30*(4), 173-178.

[11] Corcoran, K. (2014). Fooducate.

[12] Smith, J. M. (2011). The doctor will see you ALWAYS. *Spectrum, IEEE*, *48*(10), 56-62.

[13] Ruchikachorn, P., Liang, J. J., Devarakonda, M., and Mueller, K. Watson-Aided Non-Linear Problem-Oriented Clinical Visit Preparation on Tablet Computer.

[14] Time Station: https://www.mytimestation.com/

[15] EZ Office Inventory: http://www.ezofficeinventory.com/

[16] Pharmacy Prescription API, January 6, 2013 © Copyright Walgreen Co.

https://developer.walgreens.com/api/pharmacy-prescriptions/apimethod/pharmacy-prescription-api

[17] PhoneGap or Titanium or Xamarin - Which Cross-Platform Framework Should You Choose? http://www.cygnet-infotech.com/blog/phonegap-or-titanium-or-xamarin-which-cross-platform-should-you-choose#sthash.e5eS4Ibo.dpuf © 2016 Cygnet Infotech

[18] The NPD Group: For Once-Strong Smartphone Makers, 2011 Was the Year of New Beginnings. [Online]. 18 Nov 2012 © 2016 The NPD Group, Inc. https://www.npd.com/wps/portal/npd/us/news/pressreleases/pr_111213/!ut/p/c5/04_SB8K8xLLM9MSSzPy8xBz9CP0os3g3b1NTS98QY0MDVzcLA09T_zBn1zBfY38vQ_1I_SjjeBc3Sw8PN28TQ4sgS6C8q7-hi2dQgJGFpZF-QXagIgDXKdJi/

[19] Mono (Software) https://en.wikipedia.org/wiki/Mono_(software)

[20]Amazon Web Services Developer Guide © 2016

[21] Werner, V. (2013). Taking DynamoDB beyond Key-Value: Now with Faster, More Flexible, More Powerful Query Capibilities. *All Things Beautiful*

http://www.allthingsdistributed.com/2013/12/dynamodb-global-secondary-indexes.html

[22] Nemes, A. (2015) Using Amazon DynamoDb with the AWS .NET API. *Tips and tricks in C# .NET*

http://dotnetcodr.com/2015/01/22/using-amazon-dynamodb-with-the-aws-net-api-part-1-introduction/

[23]Lardinois, F. (2015), Amazon Launches AWS Device Farm, Lets Developers Test Android And Fire OS apps on Real Devices

http://techcrunch.com/2015/07/09/amazon-launches-aws-device-farm-lets-developers-test-android-and-fire-os-apps-on-real-devices/