MEDROID - AN ANDROID APPLICATION FOR ONLINE PATIENT DOCTOR APPOINTMENT

By
Capt Ahmad Bilal
Capt Naveed Ahsan
Capt Muhammad Hafeez

Submitted to the Faculty of Computer Science, Military College of Signals

National University of Science and Technology, Rawalpindi in partial fulfillment for the requirements of a B.E in Software Engineering

May 2016

ABSTRACT

Medroid provides a common medical platform to doctors and patients. App users will be registered with a unique ID (separate for patients and doctors). Patients can search doctors by their name and specialty. They can also consult a doctor and will be able to book appointments of doctors online. Doctors will have a public diary in which their appointments schedule will be shown. Users can be searched through their name and specialty.

In order to provide a descriptive range of online doctor appointment on web is challenging. Taking a doctor appointment is a time consuming and costly procedure. The domain provides online appointment facility, information about highly approved doctors.

Taking doctor appointment is tired and time consuming process and no one is willing to wait for a long time in clinic. As the clinic services are not available any time and there is much difficulty for the patients who are far away from clinic.

As computer technology is developing very fast and due to the development of internet the world becomes a global village where people can access necessary information instantly. This application provides online doctor appointment through internet.

CERTIFICATE FOR CORRECTNESS AND APPROVAL

Certified that work contained in the thesis – Medroid an Android Application for online patient doctor appointment carried out by **Capt Ahmad Bilal**, **Capt Naveed Ahsan** and **Capt Muhammad Hafeez** under supervision of **Brig Dr. Fahim Arif** and co-supervision of **Asst Prof Waseem Iqbal** for partial fulfilment of Degree of Bachelor of Software Engineering is correct and approved.

Approved by

Asst Prof Waseem Iqbal

CSE DEPARTMENT MCS

DATED: _____ 2016

DECLARATION

No portion of the work presented in this dissertation has been submitted in support of another award or qualification either at this institution or elsewhere.

DEDICATION

In the name of Allah, the Most Merciful, the Most Beneficent
To our parents, without whose unflinching support and unstinting cooperation,
a work of this magnitude would not have been possible

ACKNOWLEDGEMENTS

We bow our head to Almighty ALLAH who bestowed upon us the vision and determination to transfer "a figment of our imagination" to reality.

No words to describe the respect for our parents, because without their love and support it would have been impossible to attain this target. God bless them Ameen!

We acknowledge the support and encouragement of our brothers and sisters throughout our studies. We also acknowledge the guidance of teachers who gave us knowledge.

We are especially indebted to our supervisor **Brig Dr Fahim Arif** and co-supervisor **Asst Prof Waseem Iqbal** for their guidance and supervision to complete this work. We are very thankful to them for giving time and expertise. Their suggestions, time-to-time assistance and painstaking attention were valuable for us. We all are proud of their kindness and sympathy.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Introduction

In order to provide a descriptive range of online doctor appointment on web is challenging. Taking a doctor appointment is a time consuming and costly procedure. The domain provides online appointment facility, information about highly approved doctors.

1.2 Need of online appointment

Taking doctor appointment is tired and time consuming process and no one is willing to wait for a long time in clinic. As the clinic services are not available any time and there is much difficulty for the patients who are far away from clinic.

1.3 A Brief Introduction

As computer technology is developing very fast and due to the development of internet the world becomes a global village where people can access necessary information instantly. This application provides online doctor appointment through internet.

1.4 App Objectives

The primary objectives that caused the development of the online doctor appointment are many, some of which are listed below:

1.4.1 Doctor Appointment Online

In this modern world people usually refer to use a digital means because they have not enough time and like to take doctor appointment online. They also want to see detail information about the doctors.

1.4.2 Better Patient Services

It means better and quick services. Web based patient service makes patient happy. Instead of going to clinic for taking appointment and wait for a long time, online doctor appointment provides the instant appointment and a lot of information. It saves money and time for patients.

1.4.3 Information Sharing Convenience and Control

It enhances data offering in the middle of patients and specialists without a moment to spare deliveries. Patients spare cash, online 24 hours a day and 7 days in week, no car influx and no group.

1.4.4 Manual System

You may go to various clinics & hospitals for taking relevant doctor appointment which is time consuming, but here is not the situation you

simply search for specific doctor whom you want. Availability means available to anyone, anywhere and anytime.
In order to cater for some of the problems faced due to manual processing, we proposed a project to develop an automated solution for the management of three of the sections of the clinic & hospital i.e.

- Admin section
- Patient section
- Doctor section

The above sections will be discussed in more detail later in the document.

Our project is a database based system as this is the most appropriate form of automation for management systems. We have provided proper security mechanism in order to forbid unauthorized access to the system. Only authorized user can log in to the system. An admin of a section is the most privileged user who can perform most of the operation on the data.
The system stores the necessary information about each patient of the clinic. Every patient will have his own profile. Each patient will be able to create a profile by entering all the correct information relevant to him. The patient can search the available doctors in the clinic and also see the profile related to him.

Technology to Be Used:
Different software's and tools were used during the completion of project, which include:

- JAVA
- Android
- MYSQL
- PHP

1.5 Literature Review

The project really shows the current trends. For the development of the Android application we have studied JAVA, PHP language and MySQL from different books. Internet was a great help as many good sites were available to take help from. We were familiar with PHP and MySQL.

The set of web and Android development tools that use are given below:

- PHP
- JAVA
- MySQL
- Android
- Apache web server

1.6 Intro to Java:

Java for Android

Java is a programing language which is used for many platforms and it is also used for android application development. Java is developed by Sun Microsystems which is now owned by oracle. Java came after C and C++ programming language. It is very powerful language due to its libraries.

Some of the Java's important core features are:

- Essay to learn and implement
- It is platform independent
- It is object-oriented

Android depends on Java there are many libraries of java used in Android SDK.

1.7 Computer Engineering

Computer Engineering is combination of Software Engineering and Electrical Engineering. It holds the working of software engineering and process involved in software engineering for making software.

So all those thing that are related to software are also related to software engineering.

We have also studied about well-engineered software from software related books i.e. What will be the characteristics of acceptable software? It has the following characteristics:

- Reliable
- Good GUI
- Efficient
- Good quality
- Cost-effective
- Functionality
- Maintainable

1.7.1 Software Engineering phases

There are four main phases used for software development. These are analysis, design, implementation, and testing. Following are the parts further explained.

1.7.1.1 Analysis

In this characteristic application is analyzed. This stage characterizes the issue that the client is attempting to comprehend. The deliverable result toward the end of this stage is a necessity report. Preferably, this report states in a reasonable and exact style what is to be fabricated. This investigation speaks to the "what" stage. The necessity archive tries to catch the necessities from the client's viewpoint by characterizing objectives and connections at a level expelled from the execution points of interest.

1.7.1.2 Design

In Design stage all the requirements are designed to form a shape on which further work is done to conclude its further outcome. Design is reconsider many times to completely fulfill the requirements.

1.7.1.3 Implementation

In this phase all the design and analysis phase data is implemented. The idea is converted into algorithms is formed to a working application in implementation phase.

1.7.1.4 Testing

Testing phase is defined as the checking of working of software which is implemented in the design phase and error in the software is debugged in this phase.

1.8 Use cases

A utilization case is a portrayal of how clients will perform operations on your application.

A utilization case incorporates two principle parts:

- The steps a client will take to perform a specific errand on your site.
- The way the Web webpage ought to react to a client's activities.
- A utilization case starts with a client's objective and finishes when that objective is satisfied.

A utilization case depicts a grouping of communications between a client and application, without determining the client interface

Every utilization case catches:

- The performing artist (who is utilizing the App?)
- The cooperation (what would the client like to do?)
- The objective (what is the client's objective?)

By and large, you compose the ventures in a utilization case in a straightforward account. This connects with individuals from the configuration group and urges them to be effectively included in characterizing the pre-requisites.

- Identify why going should be utilizing the App.
- Pick one of those on-screen characters.
- Define what that on-screen character needs to do on the app. Everything the on-screen character does on the site turns into a utilization case.
- For every utilization case, settle on the typical course of occasions when that performing artist is utilizing the site.

- Describe the fundamental course in the portrayal for the utilization case. Depict it as far as what the on-screen character does and what the framework does accordingly that the performing artist ought to be mindful of.
- When the fundamental course is portrayed, consider exchange courses of occasions and add those to "broaden" the utilization case.
- Repeat the steps 2 through 7 for all other performances.

1.9 Methodology

While working on software development the design methodology plays a very vital role in it and holds all the basic functionality.
It is always a difficult task to select something from a range of available alternatives. We studied all the major software process models. It's the job of software engineer to select the appropriate software process model for a particular project. In the end we decide to follow the agile methodology. We developed the project with proper guidance of our project supervisor. We had taken the role of analyst, designer, and programmer and in the end we have tested the program by our self.

1.10 Agile Methodology

Lithe improvement approaches have confidence in a speedy and more versatile advancement of programming which is the reason it is frequently utilized for app improvement when extension is dubious. It additionally has an abnormal state of client info as inputs from the clients are needed at all the phases of the methodology of programming advancement.

Lithe approaches take a shot at cycle's i.e. progressive estimate. The product improvement handle under this procedure begins with the essential suspicion that the whole deliverable has been seen adequately well so it can be separated into little assignments.

After this the assignment is further broken into littler assignments or emphases called sprints. Every sprint includes a little programming improvement lifecycle. It obliges inputs from the client, successive assessments, incorporation of obliged changes and acknowledgement testing. This is the motivation behind why little changes in the client's venture administration necessities can in principle be fused at any phase of the item advancement.

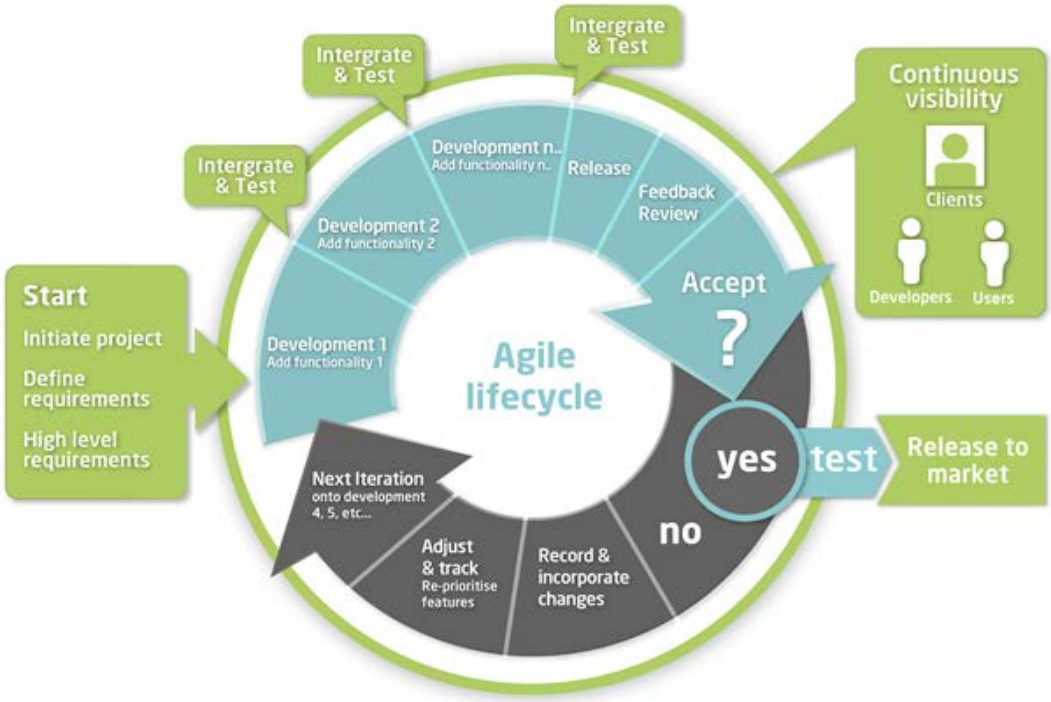


Figure 1-Agile Lifecycle

Chapter 2: Literature Review

2.1 Introduction

The existing system for healthcare is manual from which many people get some problem and it is not efficient. For increasing the efficiency an E-Health system is to be made to fulfill all the basic requirements for the appointment and other consultancy from doctor. As people around the world are turning towards internet so this project can fulfill the basic requirements and further it can be improved.

2.2 Drawbacks

Following are some drawback of the old system:

2.2.1 Transport

Patients have a lot of problem to go to the specific clinic where the patient wants to take an appointment. Similarly searching for specific specialty doctor is much difficult for patients. Other problems such as large distance, traffic jam, doctor's schedule, clinic locations etc.

2.2.2 Time Consuming Process

Taking doctor appointment manually is a time consuming process because you will have to go to the specific clinic and wait for a long time to get an appointment if the clinic staff is busy.

2.2.3 File Maintenance

A manual file or register was maintained by the clinic and hospital which is difficult to sort.

2.2.4 Inefficient Updating

Record is updated in very poor manner and human's errors are the major factor for the inefficiency.

2.2.5 Long Access Time

Manual system requires time to find a month old record if the patients are coming frequently.

2.2.6 Chance of Errors

In human manner of work chances are there for errors.

2.2.7 Redundancy

Redundancy occurs due to the old register manual system as its slow and human error can occur.

2.2.8 Integrity

Data is not secure on the manual system as anyone can change the entries and record.

2.2.9 Analysis Problems

It is difficult for one to do analysis of the data written. It is a slow method to achieve the task.

2.2.10 Backup Problem

There is not a single way to back up the data efficiently in the manual system and it is also slow method to back up the data.

2.2.11 Data Entry

Manually data entry is slow and human error occurs.

2.2.12 Data Recovery

Error that occurs accidentally and which remains unknown leads to erroneous result, which takes months to correct.

Chapter 3: Software Requirement Specification

3.1 Introduction

3.1.1 Purpose of SRS

This document is the Software Requirements Specification for the MEDROID online appointment system. It details the results of the analysis effort performed by the developers of **MEDROID**. This analysis effort reflects the intentions of **MEDROID**'s Business Plan, as well as providing the basis for design and prototyping of the online appointment system. Finally, this document will also be referenced during implementation and testing of the final system, to be performed at a later iteration.

3.1.2 Scope of product

The system initially provides for a Doctor/patient online appointment system. The schedule is a dynamically generated. It is created from information submitted by the doctor through online forms. The patient uses the system to find a doctor and view their schedule. The schedules show information such as the doctor's working hours, holidays etc. The patient then uses the system to request an appointment at an available time. The doctor looks at a master view of the schedule and approves the requested appointments. A message is sent to the patient to confirm the appointment that they requested.

3.1.3 References

Following document has already been submitted and approved.

- The project proposal document
- IEEE Std 1058-1998, IEEE Standard for Software Project Management Plans
- IEEE STD 830-1998, IEEE Recommended Practice for Software Requirements Specifications
- IEEE STD 1016-1998, Recommended Practice for Software Design Descriptions

3.2 General Description

3.2.1 Product Perspective

The proposed system allows doctors to make their schedules available online so that patients can find doctors and make appointments from the convenience and privacy of **MEDROID** application.

3.2.2 Product Functions

- **Find Doctor** – A patient uses this interface to search for a doctor. Ultimately, the patient might be able to search for doctors based

on appointment availability time (e.g. I need to find a doctor that takes appointments early in the morning).

- **Patient Schedule** – This is a limited view of the schedule in which patients request appointments. It does not show what other patients have made appointments for. Approved and requested appointments are represented as unavailable appointment times.

- **Doctor Schedule** – This is a master view of the schedule that shows detailed appointment information. Some of this information should be hidden from the patient view. Doctors can use this view to approve requested appointments.

- **View Appointments** – A registered patient uses this interface to see what appointments he/she has scheduled.

3.2.3 User characteristics

The users of this system are assumed to have a basic understanding of the Internet and android.

3.2.4 Assumptions and dependencies

- Our System shall be assuming and depending upon the following facts:
- Users have sufficient knowledge of Computers and Smart Phone's Usage.
- Users know the English language because user interface will be provided in English.
- The users are clear about the objectives.
- User authentication procedure shall be used to protect data from unauthorized access.
- The project scope is fixed but we may enhance the functionalities of the application according to the available time and the required resources.

3.2.5 Operating Environment

Under this heading, will be covering about software, hardware platform.

3.2.5.1 Software Platform

- Operating System (Server Side): Windows
- Operating System (Client End): Android
- All browsers like IE, Firefox, Safari and Google Chrome.

3.2.5.2 Hardware Platform

- PCs and laptops.
- Tablet computers.
- Smart Phones
- Internet connection.

3.3 Specific Requirements

3.3.1 Functional Requirements

The functional requirements were derived from the following use case diagram. Use case diagrams represent the functional interactions of a system. The stick figures represent the actors, which are external to the system and interact with the system through interfaces. The actors of the online appointment system are the Patient and the Doctor. The Patient uses the system to schedule appointments to obtain services from Doctors. The ovals are individual use cases that represent the functions the system performs to provide the services that the actors desire. The primary use cases of the online appointment system are:

- Find Doctor
- Manage Schedule
- Manage Appointment
- Register

The Patient interacts with the following use cases:

- Register
- Manage Appointment
- Find Doctor
- The Doctor interacts with the following use cases:
- Register
- Manage Appointment
- Manage Schedule

The online appointment system was also modeled as an analysis-level class diagram. Class diagrams represent the static associations between objects in the system. Analysis-level class diagrams avoid design, and only concentrate on the concepts related to the domain. Furthermore, distinguishing between classes and attributes is deferred until detailed design. Although, the leaves of the analysis-level class diagram are typically folded up into the parent object to become an attribute (e.g. Username would be an attribute of the Registered User class). Also, depending on the architecture chosen, this domain model would become the basis for designing the business layer of an enterprise architecture.

Patient and doctor are external actors of the system which would also be represented as objects in the system, derived from the Registered User class. Anonymous patients that are using the system to find Doctors, will not be represented as objects in the system. Patient, doctor, Schedule, and Appointment would be the primary objects in the system, and would be the most complex shown from the multiple relationships that they share. Another interesting item is how many attributes Appointment and Non Work Hours share. Design might modify these items such that Appointment is a specialization of Non Work Hours as to avoid code duplication during the Implementation

phase. UML design artifacts, such as collaboration diagrams, would provide the exercise to flesh out the complete set of attributes and methods necessary to support the system. This analysis artifact helps the transition from analysis to design.

Finally, the dynamic aspects of the system are modeled as sequence diagrams. Each use case has its own corresponding sequence diagram. The sequence diagram shows the flow of data between the system and external actors through interfaces, as well as for the flow of data and message calls between objects internal to the system. Data cannot flow between actors and objects unless they have the appropriate association represented in the class diagram. Only some of the objects of the class diagram are required to support the function of each individual use case. The scenarios represented are the most common path through the system and do not detail alternate scenarios. The scenarios are also time dependent and happen in sequence. The transitions internal and external to scenarios might be either synchronous or asynchronous.

3.3.1.1 Find Doctor

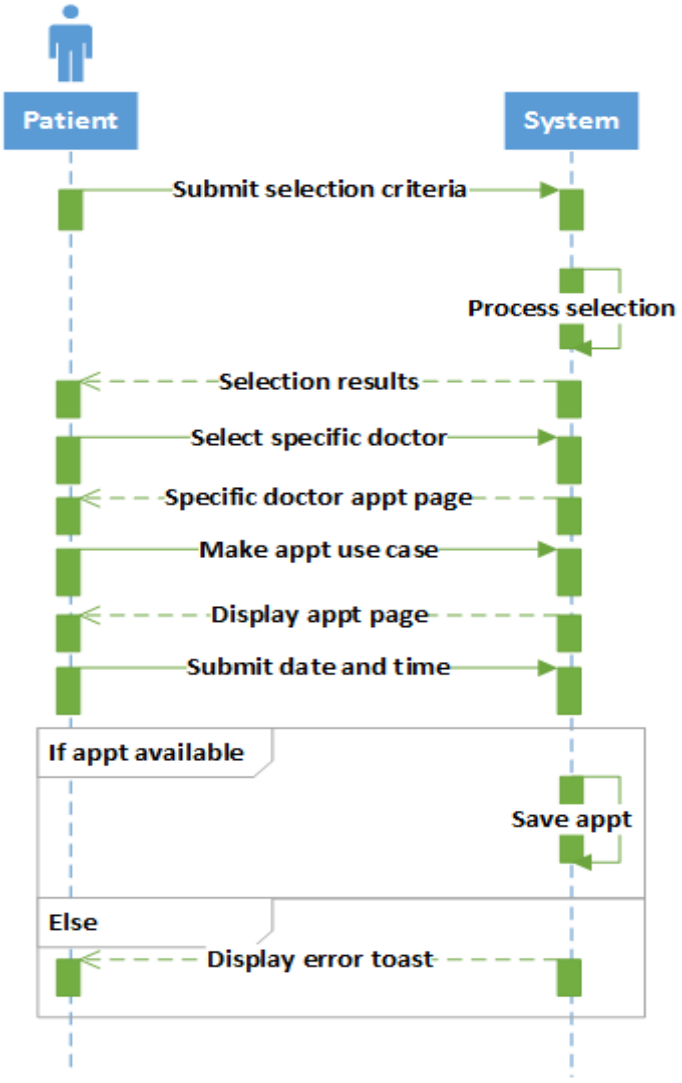


Figure 2-Find Doctor Sequence Diagram

3.3.1.1.1 Introduction

Patients shall be allowed to search for Doctors by a variety of criteria. This criterion shall be by location, name and specialty but must be extensible so that it is possible to sort or search on any data field.

3.3.1.1.2 Inputs

The user would input the searching criteria/values and the sorting criteria.

3.3.1.1.3 Processing

The system performs a search on the database of Doctor.

3.3.1.1.4 Outputs

The user is presented with a list of Doctors that match the searching criteria sorted as defined by the sorting criteria. By clicking on any doctor listed the user is taken to that doctor's main appointment page.

3.3.1.2 Create Schedule

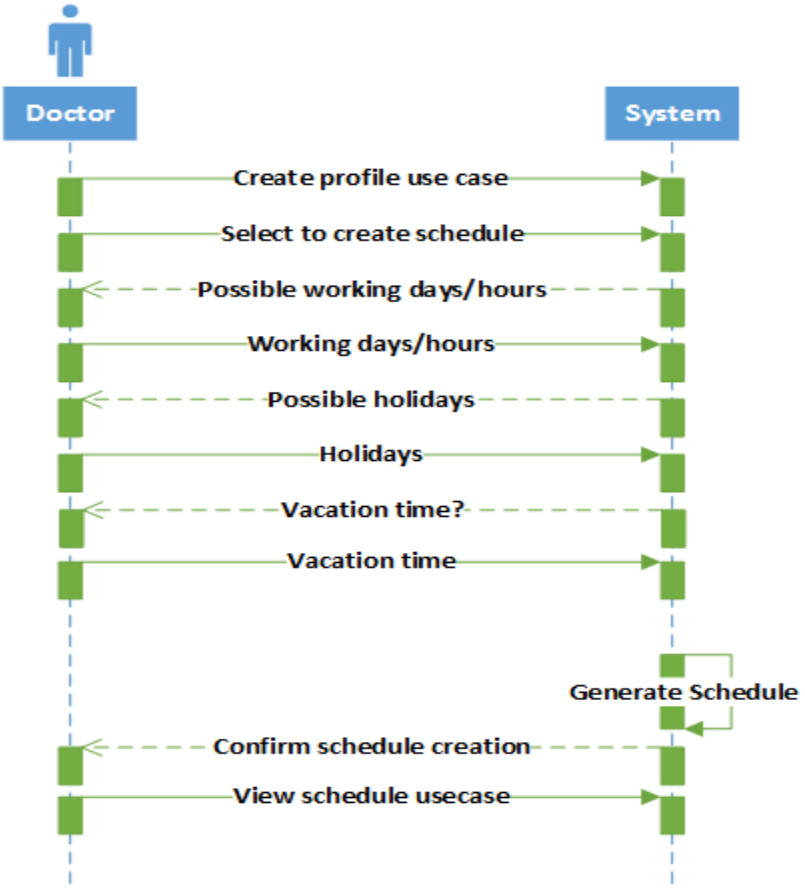


Figure 3-Create Schedule Sequence Diagram

3.3.1.2.1 Introduction

Doctors shall be able to define their schedule. This schedule shall determine when patients can make appointments to see that Doctor. The schedule shall be exception based. The schedule consists of the normal weekly schedule and then will contain exceptions to this normal schedule.

3.3.1.2.2 Inputs

The Doctor shall initially enter his/her normal weekly schedule. Then the Doctor will enter exceptions to this normal weekly schedule. These exceptions will take the form of a day/time and duration. Exceptions may reoccur. Reoccurrences may be weekly, monthly, or annually.

3.3.1.2.3 Processing

The system will create an internal representation of the specified schedule.

3.3.1.2.4 Outputs

The system will confirm the schedule addition.

3.3.1.3 View Schedule

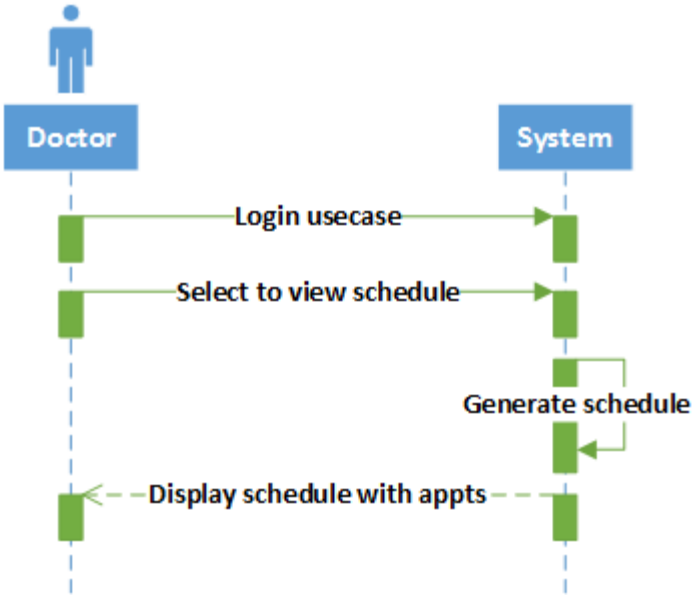


Figure 4-View Schedule Sequence Diagram

3.3.1.3.1 Introduction

The Doctor shall be able to view his/her schedule. This schedule shall be in the format of a calendar. The default view will be a weekly view, and the Doctor shall be able to view the schedule for the current week or any week in the past/future. The Doctor shall have the ability to navigate weeks by using a month view on the same page.

3.3.1.3.2 Inputs

The Doctor will choose to view his/her schedule.

3.3.1.3.3 Processing

System will retrieve schedule from the server.

3.3.1.3.4 Outputs

The schedule view as described above.

3.3.1.4 Change Schedule

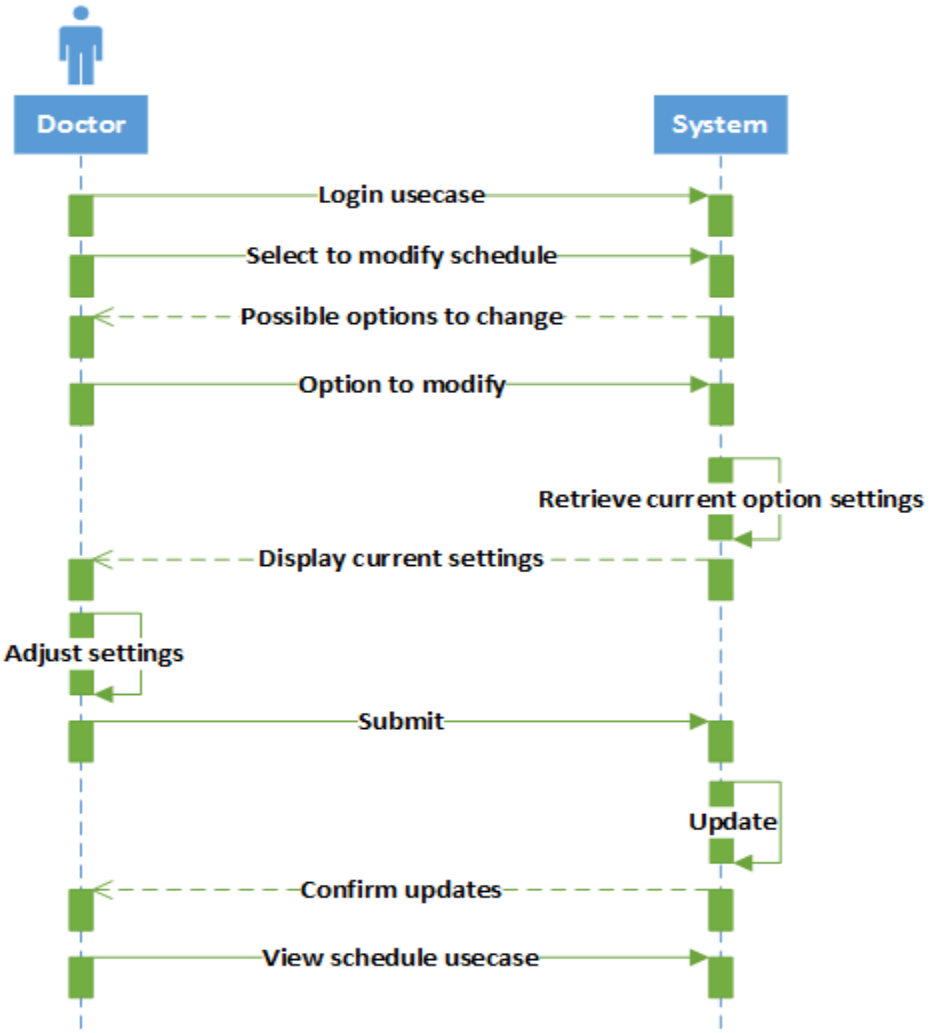


Figure 5-Change Schedule Sequence Diagram

3.3.1.4.1 Introduction

Doctors shall have the ability to change their schedule much in the way they originally created it. They can either change their standard weekly schedule or add/change/delete exceptions.

3.3.1.4.2 Inputs

The Doctor chooses to either modify standard weekly settings or to modify exceptions. The Doctor will then provide the modification values.

3.3.1.4.3 Processing

System updates if the changes occur.

3.3.1.4.4 Outputs

The system displays confirmation of the modification.

3.3.1.5 Request Appointment

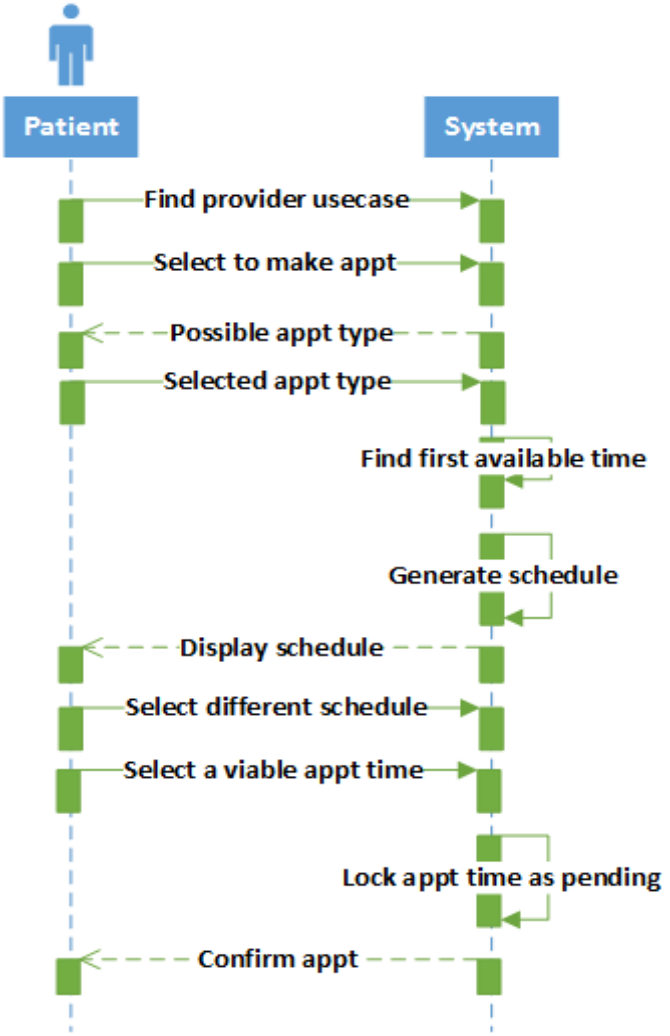


Figure 6-Request Appointment Sequence Diagram

3.3.1.5.1 Introduction

The patient uses a MEDROID application to schedule an appointment pending the Doctor's approval.

3.3.1.5.2 Inputs

The patient interacts with the Doctors schedule to select the appointment type and time for appointment.

3.3.1.5.3 Processing

The system recommends the 1st available appointment time to the patient, and locks the appointment time when the patient submits it.

3.3.1.5.4 Outputs

A confirmation of the appointment request is displayed to the user, and a notification of an appointment request is sent to the Doctor. Another patient viewing the schedule, would now see that block of time on the schedule represented as being a pending appointment.

3.3.1.6 Change Appointment

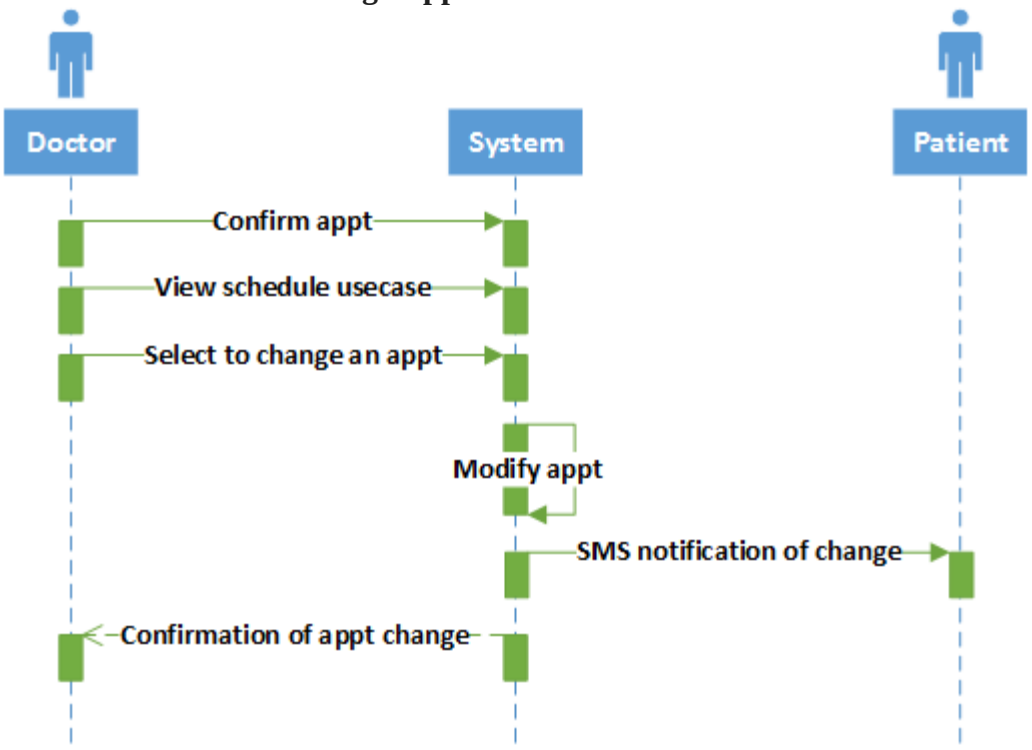


Figure 7-Change Appointment Sequence Diagram

3.3.1.6.1 Introduction

This allows patients or Doctors to change an appointment. The following sequence diagram represents the Doctor initiating the change.

3.3.1.6.2 Inputs

The Doctor selects the appointment and submits the associated information to change.

3.3.1.6.3 Processing

The system modifies the appointment and changes its state.

3.3.1.6.4 Outputs

The patient receives SMS notification of the change, and the Doctor receives confirmation that the change was submitted.

3.3.1.7 Cancel Appointment

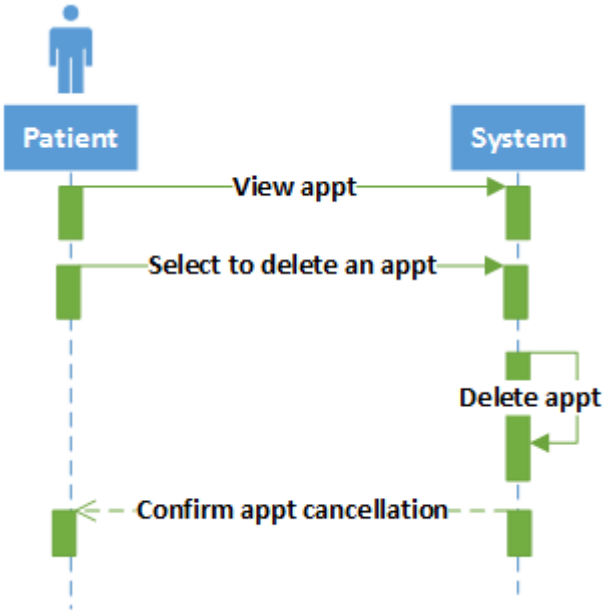


Figure 8-Cancel Appointment Sequence Diagram

3.3.1.7.1 Introduction

This allows patients or Doctors to cancel an appointment. The following sequence diagram represents the patient initiating to cancel.

3.3.1.7.2 Inputs

The patient selects to delete his or her appointment.

3.3.1.7.3 Processing

The system marks the appointment as cancelled.

3.3.1.7.4 Outputs

The Patient gets notification of the cancellation.

3.3.1.9 View Appointment

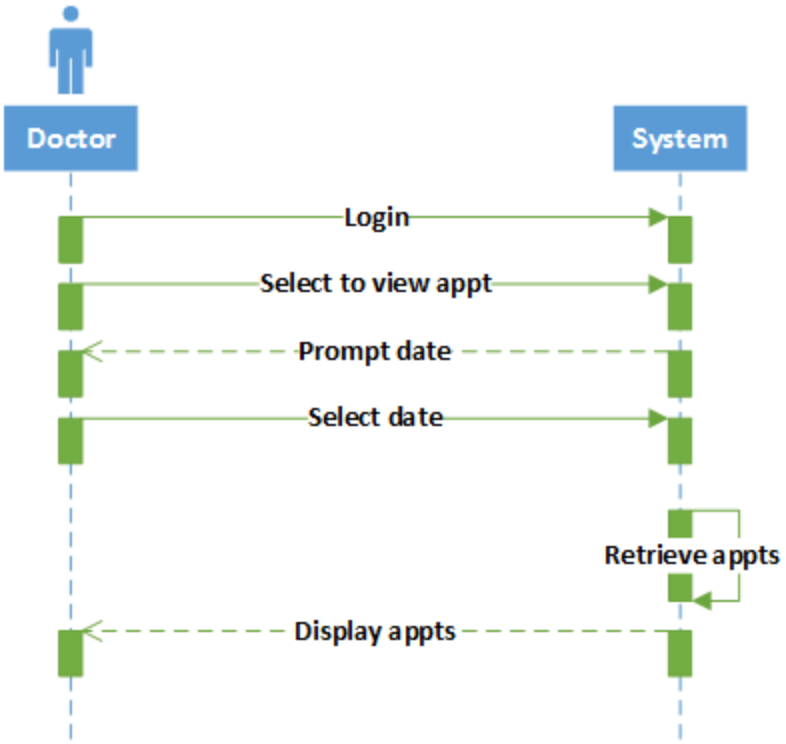


Figure 9-View Appointment Sequence Diagram

3.3.1.9.1 Introduction

Doctor does this to view his appointment(s).

3.3.1.9.2 Inputs

A registered doctor selects to view his or her appointments.

3.3.1.9.3 Processing

The system receives the doctor's appointment info.

3.3.1.9.4 Outputs

A dynamically generated page is displayed to the doctor showing his or her past and upcoming appointments.

3.3.1.10 Create Profile

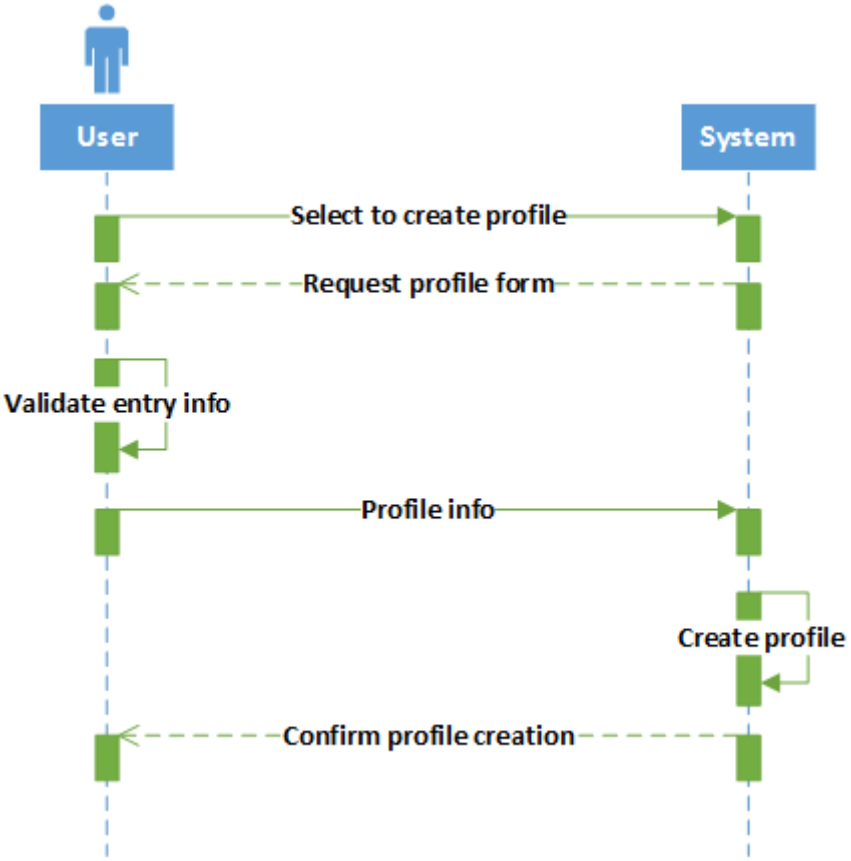


Figure 10-Create Profile Sequence Diagram

3.3.1.10.1 Introduction

The user shall be able to create a profile in the system. The profile shall consist of the user's name, physical address, phone number, email address, username and password.

3.3.1.10.2 Inputs

User's name, physical address, phone number, email address, username and password.

3.3.1.10.3 Processing

The system validates the username to make sure it is unique.

3.3.1.10.4 Outputs

Confirmation of profile creation to the screen and via email.

3.3.1.11 Login

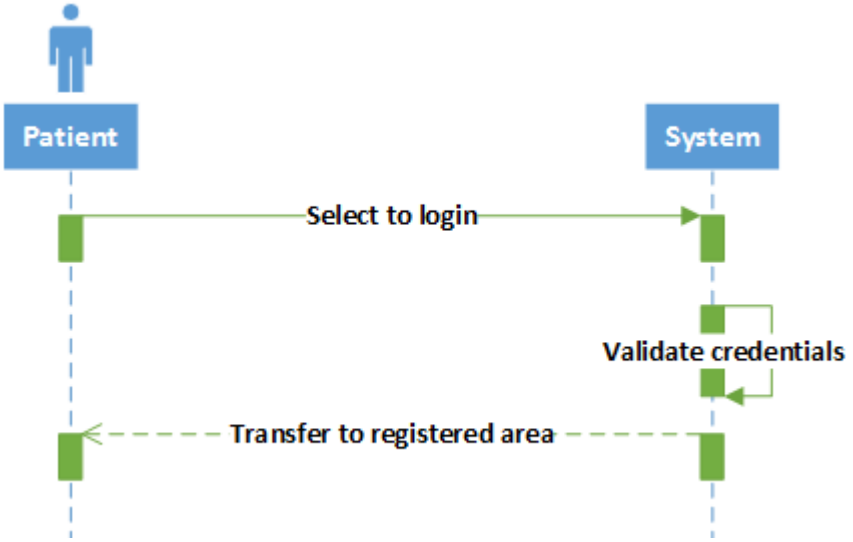


Figure 11-Login Sequence Diagram

3.3.1.11.1 Introduction

The users shall be able to login to the system. Patients and Doctors will log in using the same way, but will be taken to different screens once logged in.

3.3.1.11.2 Inputs

The user's username and password.

3.3.1.11.3 Processing

Check username and password against stored values.

3.3.1.11.4 Outputs

Login confirmation if username and password match. An error message if login fails.

3.3.1.12 Edit Profile

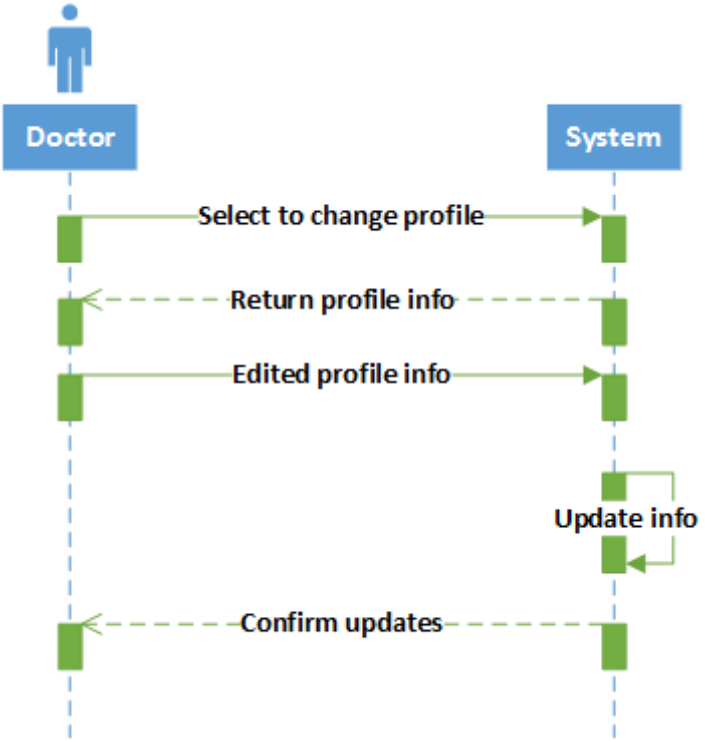


Figure 12-Edit Profile Sequence Diagram

3.3.1.12.1 Introduction

Changing the profile is very similar to creating a profile.

3.3.1.12.2 Inputs

The attributes to change and the values to change.

3.3.1.12.3 Processing

Validate username to make sure it is unique if the user changed the username.

3.3.1.12.4 Outputs

Confirmation of changed profile.

3.3.2 Non-Functional/ External Requirements

3.3.2.1 User interfaces

All user (Patient, Doctor) interfaces are android based application pages which can be accessed through any android based device. Application pages which ask for user input will include forms. These forms need to include code to verify that the appropriate information has been input before allowing the information to be submitted to the server. Pages which provide confirmation information will be dynamically generated.

3.3.2.2 Hardware interfaces

There are not direct hardware interfaces in the system. The operating systems on both the client and server side will be configured to handle the hardware interfaces.

3.3.2.3 Software interfaces

The client side software interface will be through any android based device. The server side interface will be through a Java web server.

3.3.2.4 Communication interfaces

Communication between the client and servers will occur between the standard internet protocols HTTP.

3.3.2.5 Performance Requirements

Patients will be using this system to make important appointments. If the system is unresponsive patients will be lost. The response time for any operation shall be no longer than two (2) seconds not including internet latency. To test the response time, the device will be placed on a LAN with the server and perform operations.

3.3.2.6 Standards compliance

The system complies with the internet standards of HTTP. These standards support the largest number of online customers.

3.3.2.7 Hardware limitations

There are no hardware limitations to be considered.

3.3.2.8 Software limitations

At this point in time it is not viable for MEDROID to purchase professional web hosting services. Because of this, files must be used for persistence of data to be used by the web server,

instead of persisting the data to a DBMS. Another limitation is the use of a Java web server instead of an application server. Because of this, the system will have to incorporate its own online transaction processing services as well as handle concurrent access by multiple clients.

3.3.2.9 Environmental limitations

The client side will run on any environment which supports android. The server side environment will not be a concern to the clients, and can be upgraded/changed as necessary, without affecting the clients.

3.3.2.10 Availability

The system has to be available at all times.

3.3.2.11 Security

MEDROID will possibly be handling sensitive medical/personal data so security is the utmost concern. Being able to prove to clients that the system is very secure will be essential for the future success of MEDROID. The system has to provide secure internet connections (SSL) for the transfer of private data. The web hosting service should provide security services so that third parties cannot modify the information hosted on the web site. Users will have to login to the system. Logging in will determine what level of access the user has with the system and the views with which they are provided. Security also means that user's personal information shall be guarded very closely and all passwords shall be encrypted in the database.

3.3.2.12 Maintainability

The system should be completely remote administrable. This means that system statistics (average transaction time, current active sessions, number of sessions processed in last hour/day/month) and system functions (start/stop system, reboot system, restart services etc...).

3.3.2.13 Transferability/conversion

There are no transferability concerns for the client side. The server side should be designed in such a manner so that the system can be transferred to a different web hosting service without effecting day to day operations.

3.3.2.14 Scalability

The system shall be designed so that is can easily be scaled to increase performance (response time). This means that the various tiers can be running on different processors on the same or different physical machines. This also means that the

system shall be designed so that many instances of components can be running concurrently.

3.3.2.15 Reliability

The system shall be designed so that it can be completely reliable. Reliability comes in two forms: reliability in terms of the system always being up and reliability in perfect atomic transactions.

3.3.3 Other Requirements

3.3.3.1 Database

A database is required from the offset.

3.3.3.2 Operations

Backups will be performed by both the owners and the administrators of the web hosting service.

3.3.3.3 Site adaptation

The site will be adaptable to the different users of the system, according to the access associated with the login that they provide.

3.4 Other Nonfunctional Requirements

3.4.1 Performance Requirements

Server shall be able to handle minimum of 500 requests at one time.

3.4.2 Security Requirements

- Appointments made by a user shall only be accessible to that user for modification, view and result analysis.
- Logical Database should be secure and known attacks on databases must be catered for e-g SQL Injection Attacks.

3.4.3 Software Quality Attributes

Software Quality Attributes are listed as follows.

3.4.3.1 Availability

- The system shall meet or exceed 95% uptime.
- The MTBF shall not be less than 23 hours.
- Less than 10 minutes shall be needed to restart the system after a failure, 99.90% of the time.

3.4.3.2 Maintainability

- Not more than 15% of the existing functionality shall be affected by adding any new functionality.

- Installation of a new version shall leave all database contents and all personal settings unchanged.

3.4.3.3 Response Time

It would take less than 10 seconds to completely load a page given a 1mbps internet connection.

3.4.3.4 Reusability

System shall allow the support for developers to make native applications for different platforms.

3.4.3.5 Portability

- System shall be accessible from different platforms on different Android tablets, Android Smart Phones (different sizes).
- System shall give the same look and feel being used at different platforms. Screen resolution and size adjustments should be catered for different screen sizes.
- System must use the display settings of the native device which it is being used.

3.4.3.6 Reliability

- System defect rate shall be less than 2 failures per 1000 requests to servers.
- The system shall rollback to a previous saved state in case of a failure.

3.4.3.7 Usability

- For most of the links or buttons, help would be provided to the user for knowing the main functionality of that particular link or button for his better understanding.
- Video Tutorials will be made available (constraint – sufficient time Available).

3.4.4 Business Rules

3.4.4.1 Confidentiality

Appointments shall only be visible to authorized users i-e users who have created the appointment and the doctor for which it is created for. Other users shall not be able to access appointment details of other users.

3.4.4.2 Sign-Up information usage

Users shall be made aware of the fact that the information about their age, education, profession and location will be used by authorized users i.e. the doctor whom he/she has made appointment for, to contact them for seeking any feedback.

Chapter 4: Design and Development

4.1 Introduction

4.1.1 Purpose

Design is most important part of the project as requirement analysis change into design phase. This part of design can give the overview of entire system. The design phase can produce an application specification that has addressed each feature tells in the requirements phase and should indicate how that feature will be implemented to check maximum ease of use and effectiveness. We can be able to read the system specification and know what will receive when application development can be delivered.

4.1.2 Scope

The system initially provides for a Doctor/patient online appointment system. The schedule is a dynamically generated. It is created from information submitted by the doctor through online forms. The patient uses the system to find a doctor and view their schedule. The schedules show information such as the doctor's working hours, holidays etc. The patient then uses the system to request an appointment at an available time. The doctor looks at a master view of the schedule and approves the requested appointments. A message is sent to the patient to confirm the appointment that they requested.

4.1.3 Definitions, acronyms, and abbreviations

Appendix A

4.1.4 References

- Following document has already been submitted and approved.
  - The project proposal document
- IEEE Std 1058-1998, IEEE Standard for Software Project Management Plans
- IEEE STD 830-1998, IEEE Recommended Practice for Software Requirements Specifications
- IEEE STD 1016-1998, Recommended Practice for Software Design Descriptions
- Pressman, Roger S., Software Engineering "A Practitioner's Approach", Fifth Edition, McGraw-Hill, 2000.
- *http://developer.android.com/sdk/installing/adding-packages.html*
- *http://stackoverflow.com/questions/19465049/changing-api-level-android-studio*
- *http://stackoverflow.com/questions/16782298/how-to-import-google-api-in-android-studio*
- *http://www.tutorialspoint.com/android/android_json_parser.html*
- *http://developer.android.com/reference/org/json/JSONObject.html*

- *http://developer.android.com/reference/android/util/JsonReader.html*
- *http://developer.android.com/reference/android/util/JsonWriter.html*
- *http://www.w3schools.com/json/json_http.asp*

4.1.5 Overview of document

This document shows the design and working of Medroid application. It starts from higher level details for a non-technical reader to understand just by seeing the diagrams to the lower level details that aid the developer to code and understand other technical details of the application.

In Section 2, *System Architecture Description* gives a detailed overview of the application. It shows the main component of the application and their inter-relationships. *Structure and Relationships* shows the higher level details system working by the means of System Block, Architecture and Activity. Lower level details are described using the Class, Chen's Entity Relationship, DFD, Sequence diagrams and Structure Chart.

In Section 3, *Detailed Description of Component* is given to show the working of modules with low level details. It shows the purpose, function, subordinates, dependencies, interfaces, resources, processing and data of the components and their relationships with each other.

In Section 4, *Reuse and Relationship to other Products* i.e.; information about work done in the same project before and any reuse of the same work. The section also provides a key to reuse this system for further upgrades.
Section 5 ***Pseudo Code*** of the components is given in Section 6 for human reading rather than machine reading.

4.2 System Architecture Description

4.2.1 Overview of modules / components

The project is implemented using the Java programming language with the database used at the backend. As we have selected the agile methodology (incremental model) we use a modular approach. We started with a single module and moved on to the next after completing and testing the first. Every module was broken down into several module and deal one at a time.

4.2.2 Structure and Relationships

4.2.2.1 Entity Relationship Diagram

Our physical data base diagram is almost structurally identical to our class diagram. For the most part we just took each of our

classes and made it a table, adding attributes as necessary. This was possible because we do not have any many-to- many relationships in our class diagram. This precludes the need for intermediate tables. We also do not have any classes that don't require data storage (temporary classes, etc.). The only structural change required is to create tables for the super- and sub-classes. Attributes were fairly easy to determine. Most are pulled directly from the class diagram. A few attributes in the class diagram are composite attributes and needed to be broken

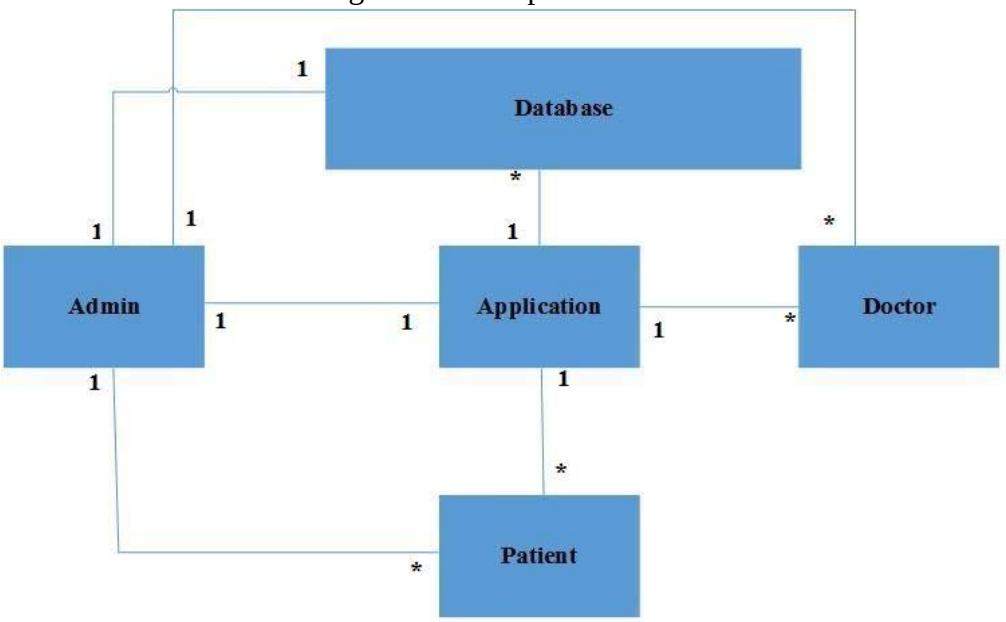


Figure 13-ER Diagram

down into multiple database columns.

4.2.2.2 Data Flow Diagram (DFD)

DFD is a simple graphical notation or representation that can be used to represent a system in terms of the input data to the system various processing carried out on that data and the output of data generated by the system.

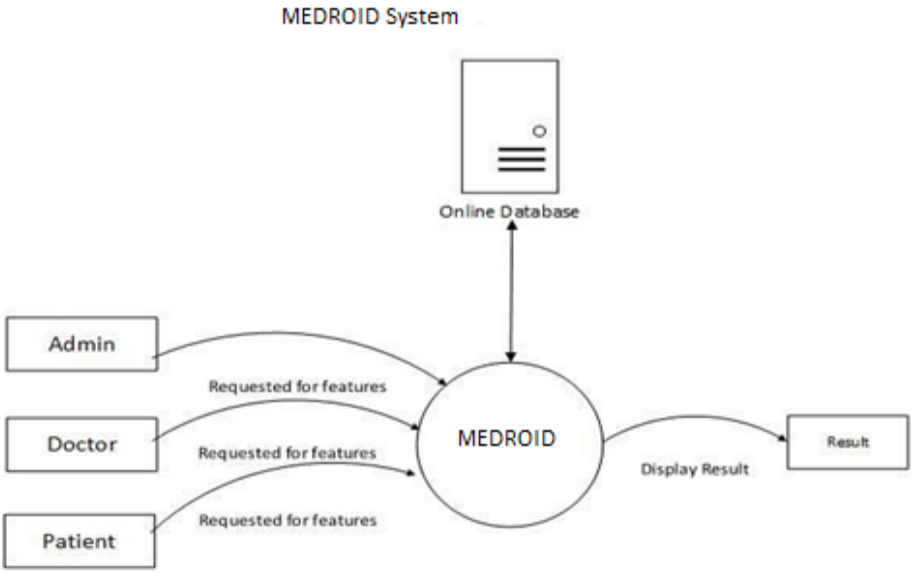


Figure 14-Medroid System

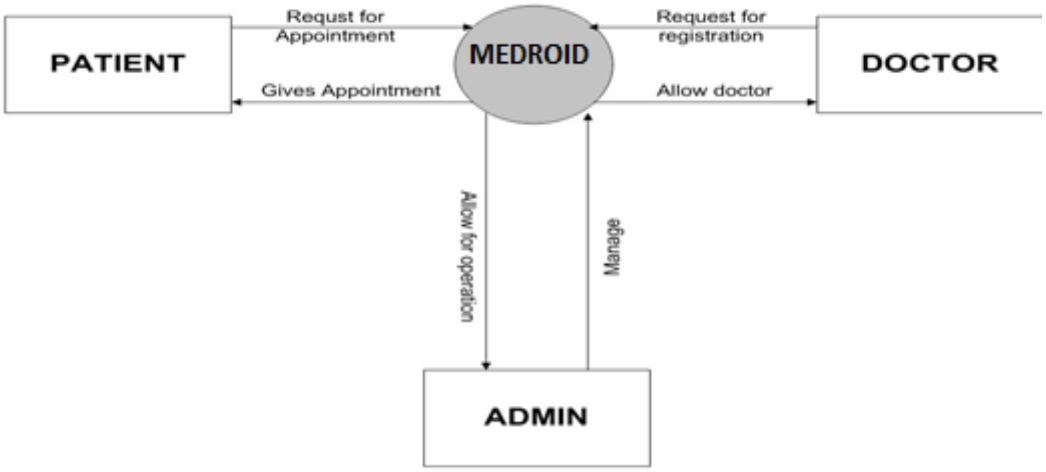


Figure 15-Context Level DFD

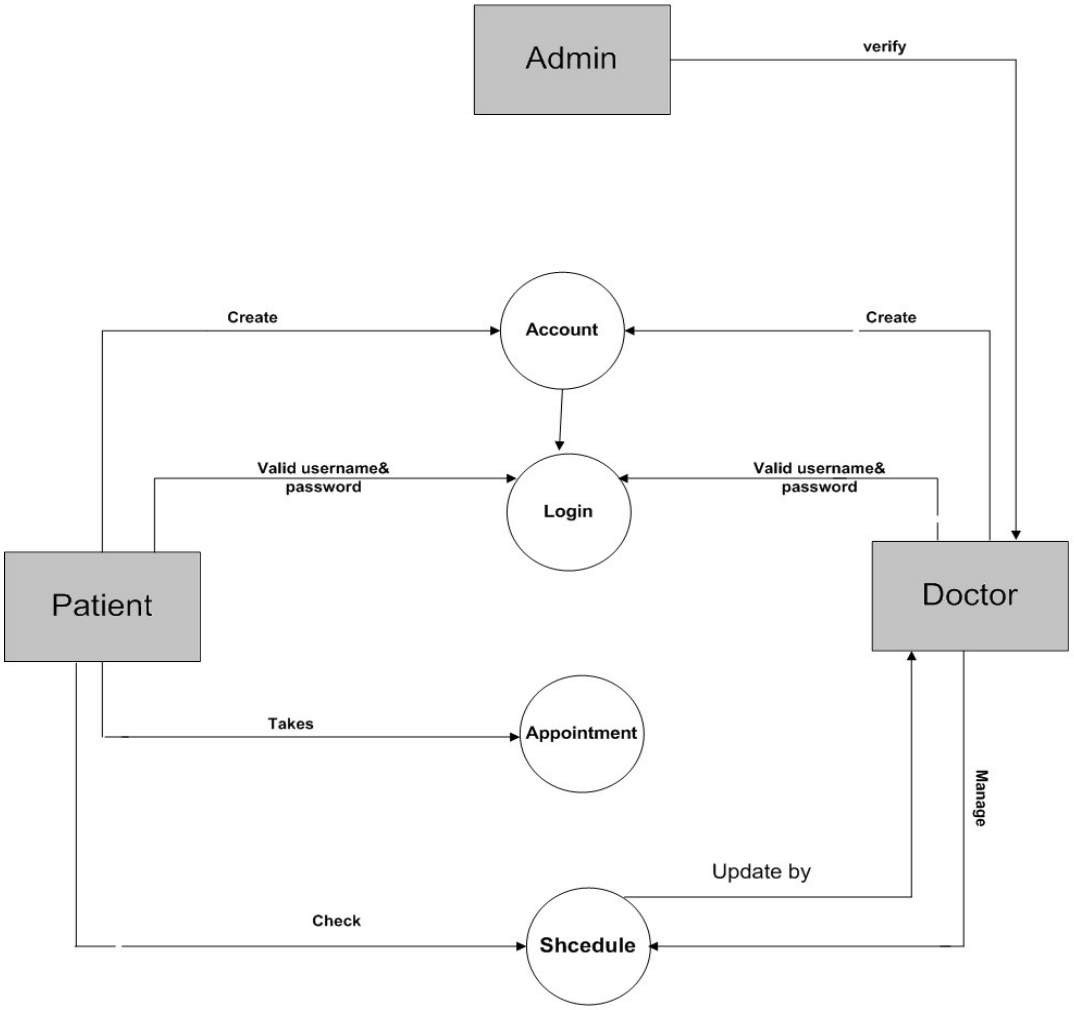


Figure 16-Level 0 DFD

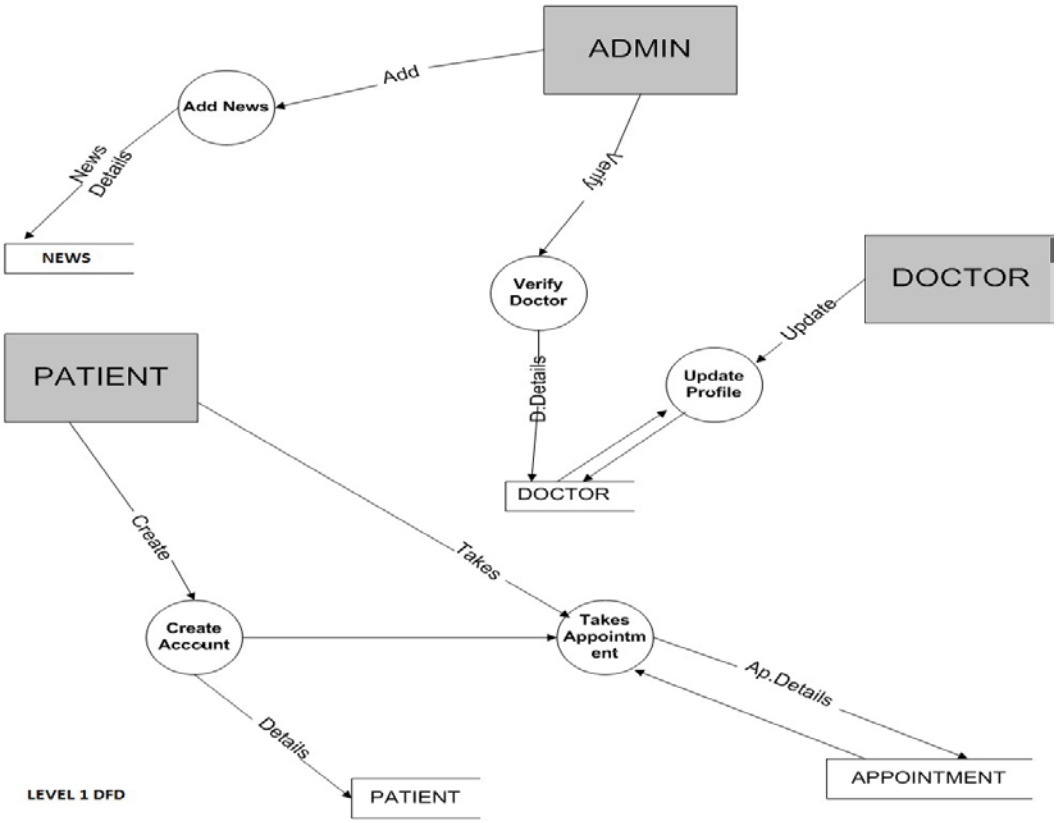


Figure 17-Level 1 DFD

4.2.2.3 Class diagram

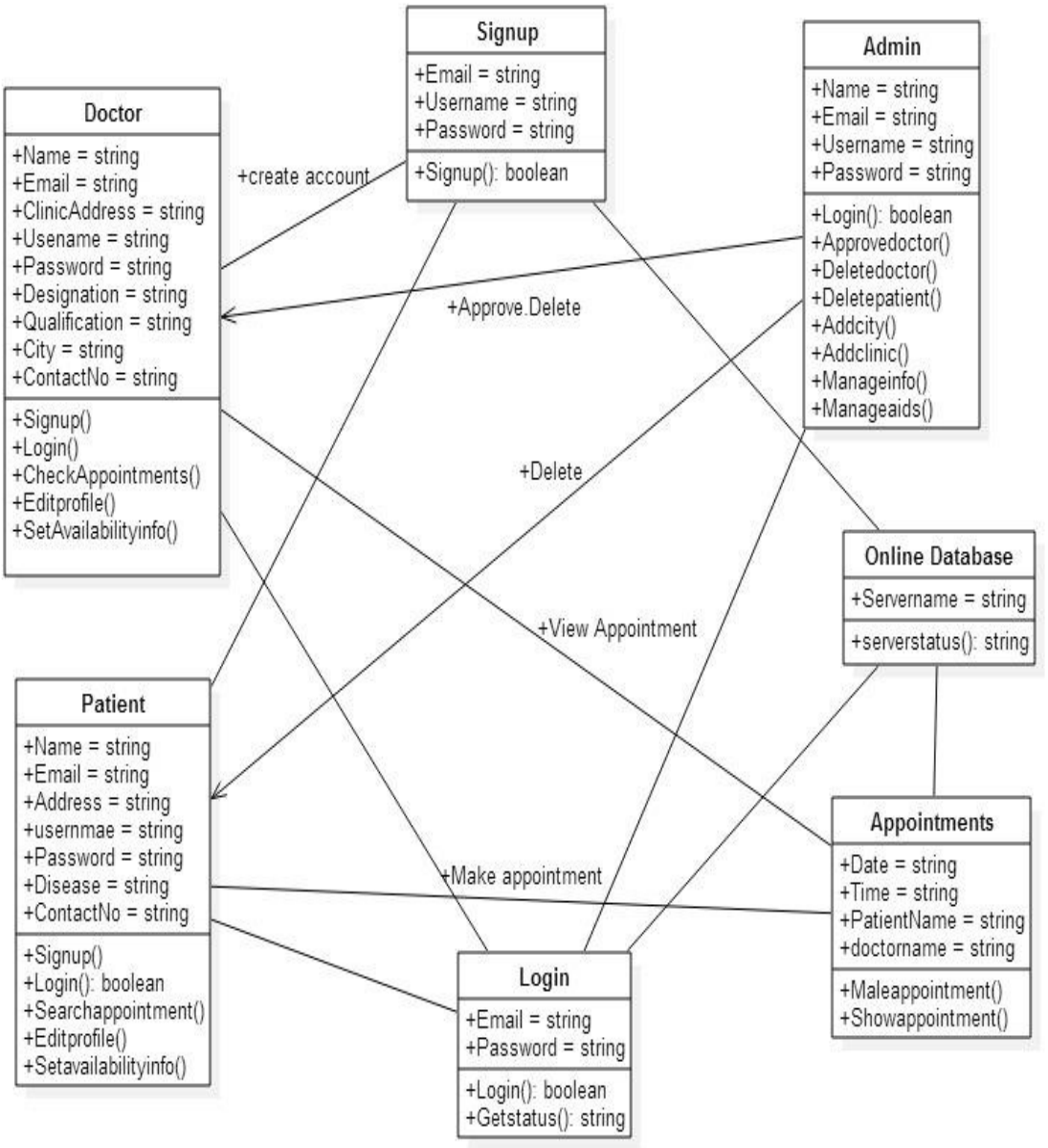


Figure 18-Class Diagram

4.2.2.4 Activity Diagram

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. Activity diagram is basically a flow chart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.

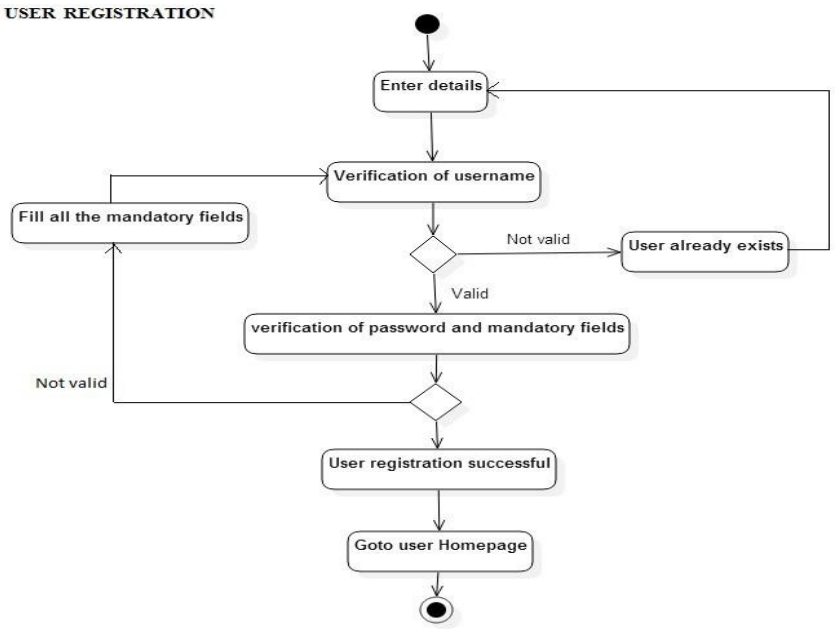


Figure 19-Activity Diagram (User Registration)

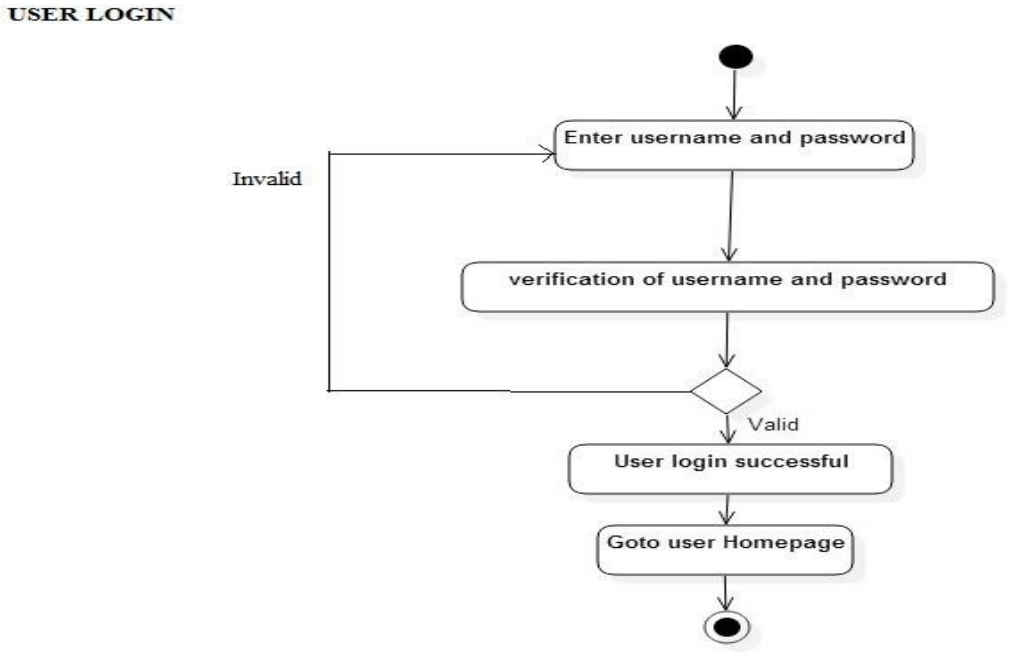


Figure 20-Activity Diagram (User Login)

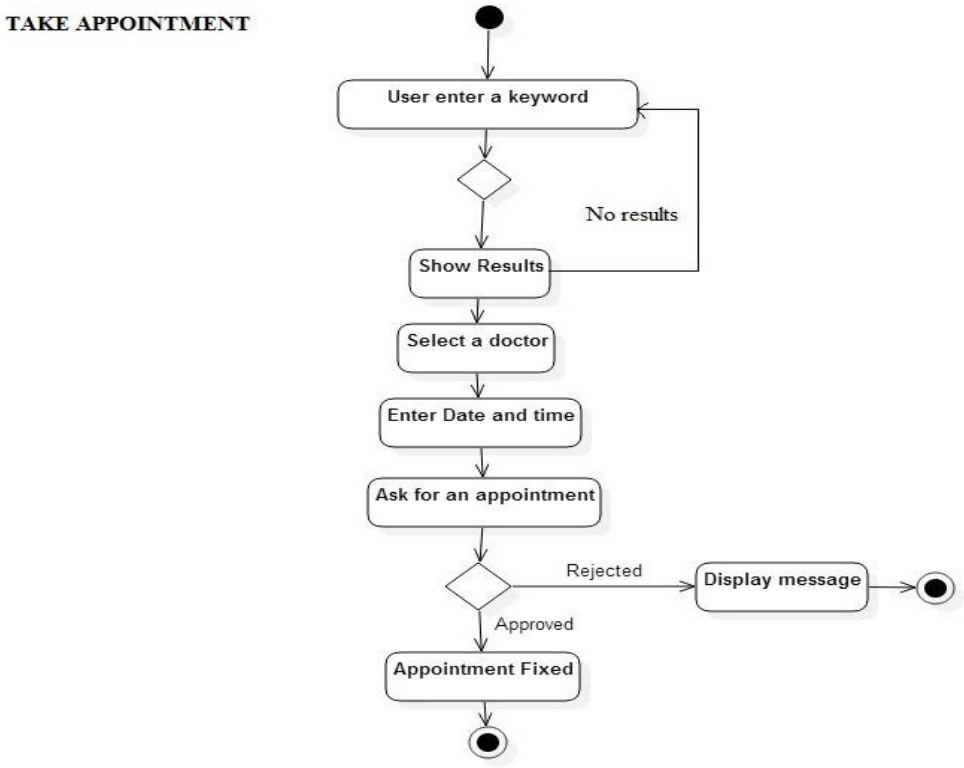


Figure 21-Activity Diagram (Take Appointment)

4.2.2.5 Sequence Diagram

USER LOGIN

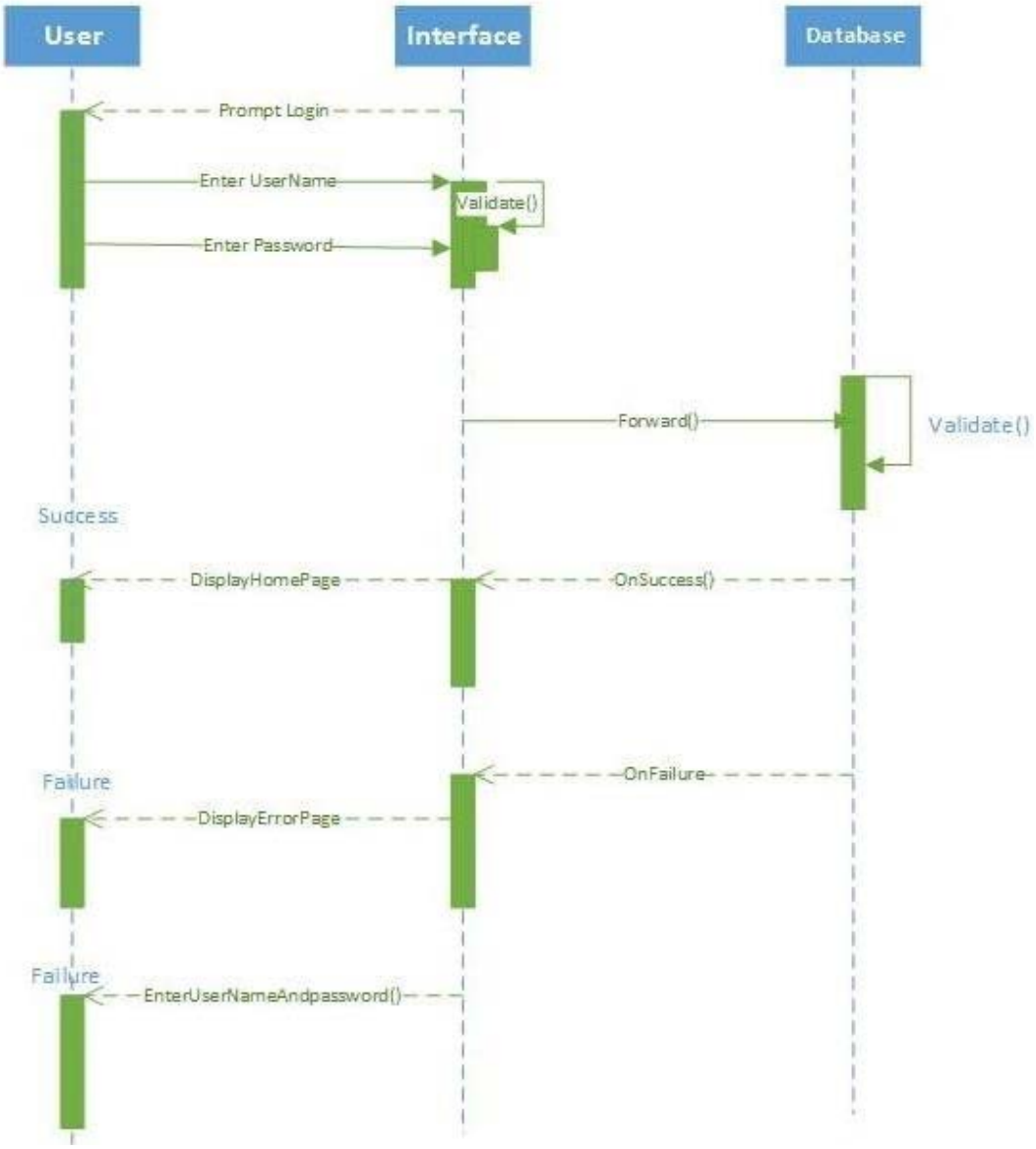


Figure 22-Sequence Diagram (User Login)

TAKE APPOINTMENT

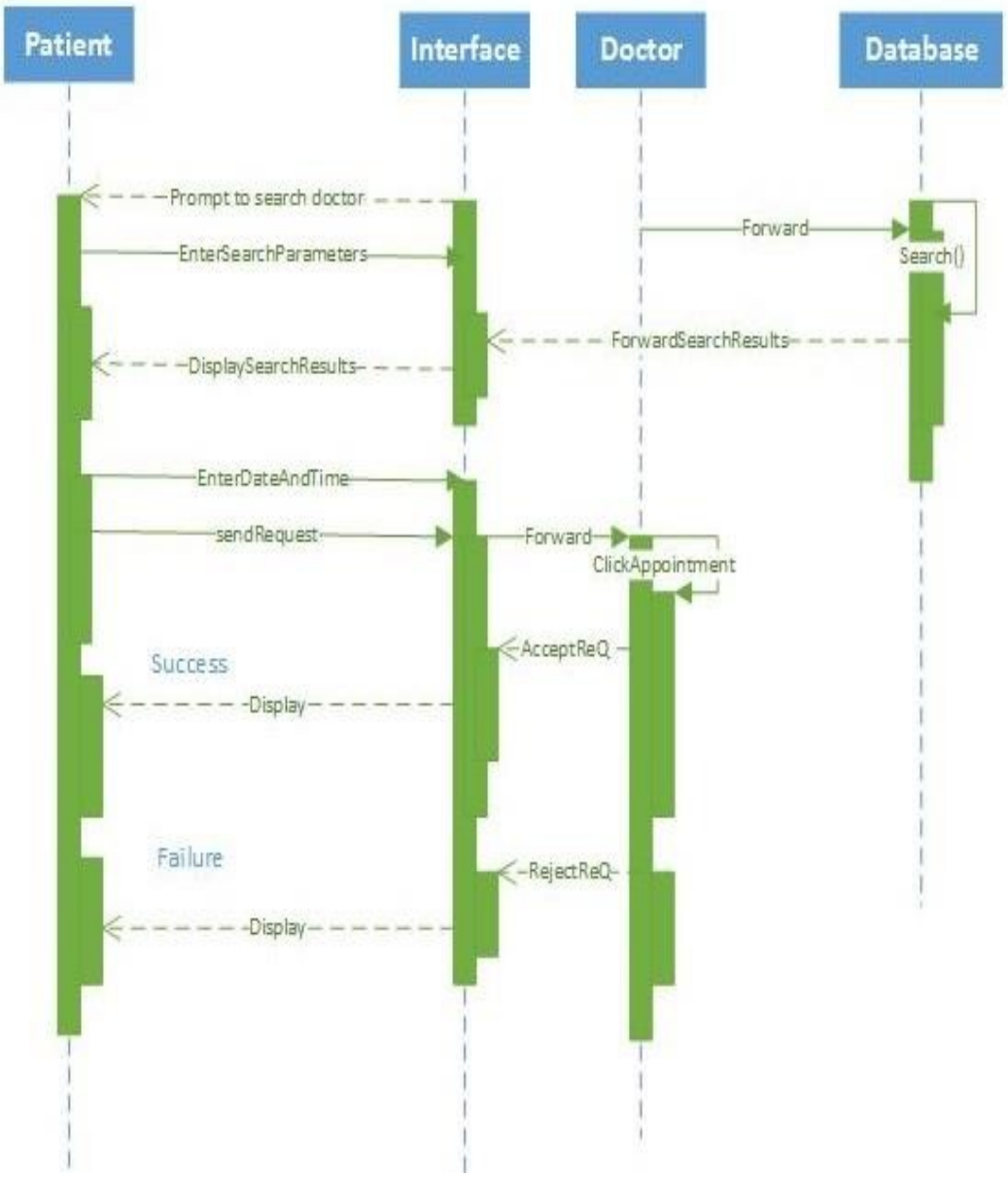
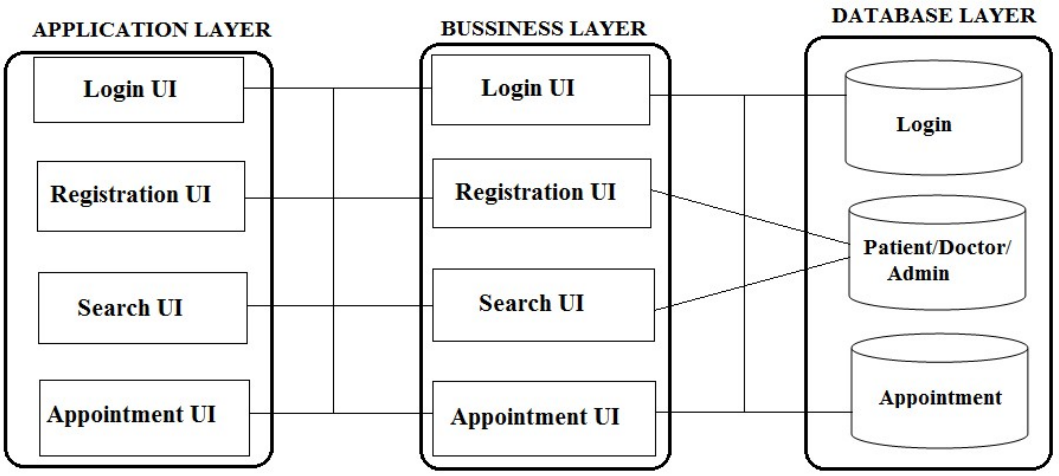


Figure 23-Sequence Diagram (Take Appointment)

4.2.2.6 Architecture Diagram



4.2.3 User Interface

This is the main screen.

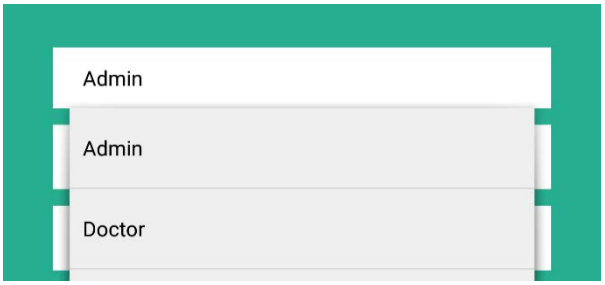


Figure 24-Architecture Diagram

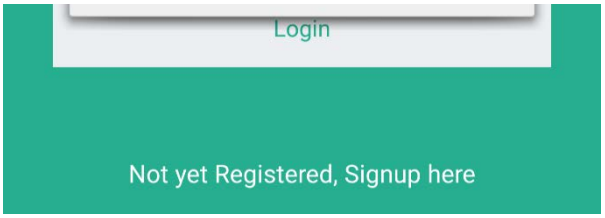


Figure 25-Main Screen

Login as Admin, Doctor or patient by entering username and password.

If you are not registered, then you need to register first as doctor or patient.

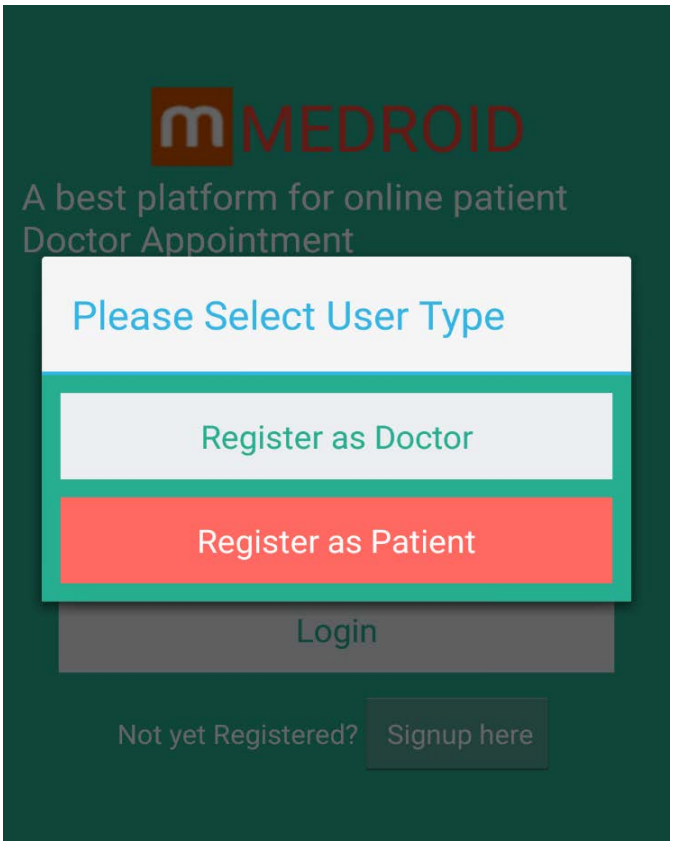


Figure 26-User Type

Fill the info appearing on the next page and then submit it for approval of admin if you are creating profile of a doctor and if you have created it as a patient then your profile is created without approval of admin.

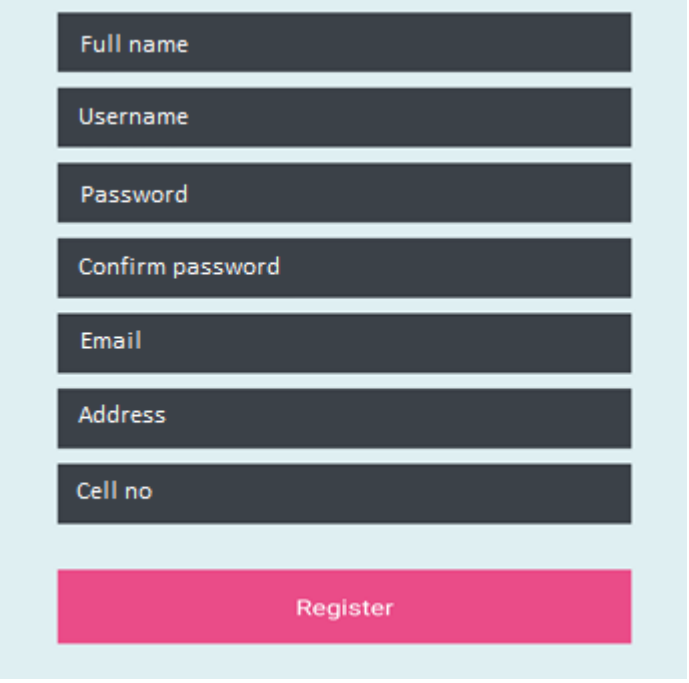


Figure 27-Information Filling Form

Once your profile as a doctor is created then you can use different functions of it.

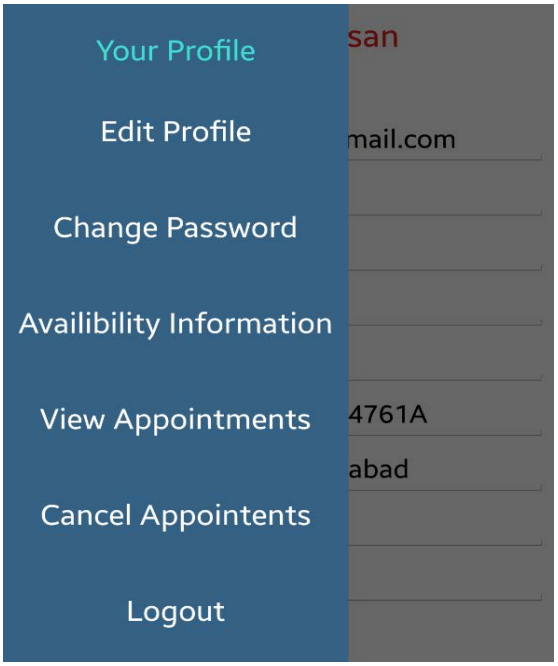


Figure 28-Options for Doctor

When your profile is created as a patient you can see your booked appointments, you can search doctor by name and disease and then book appointments.

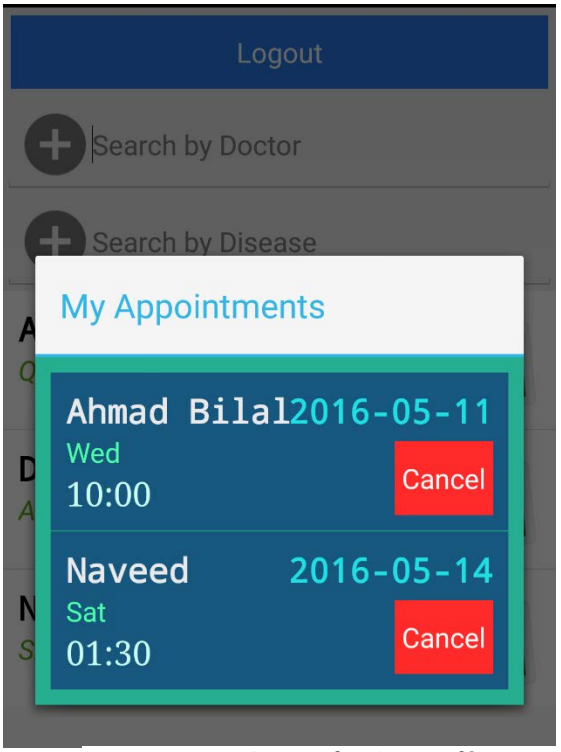


Figure 29-Front Screen of Patient Profile

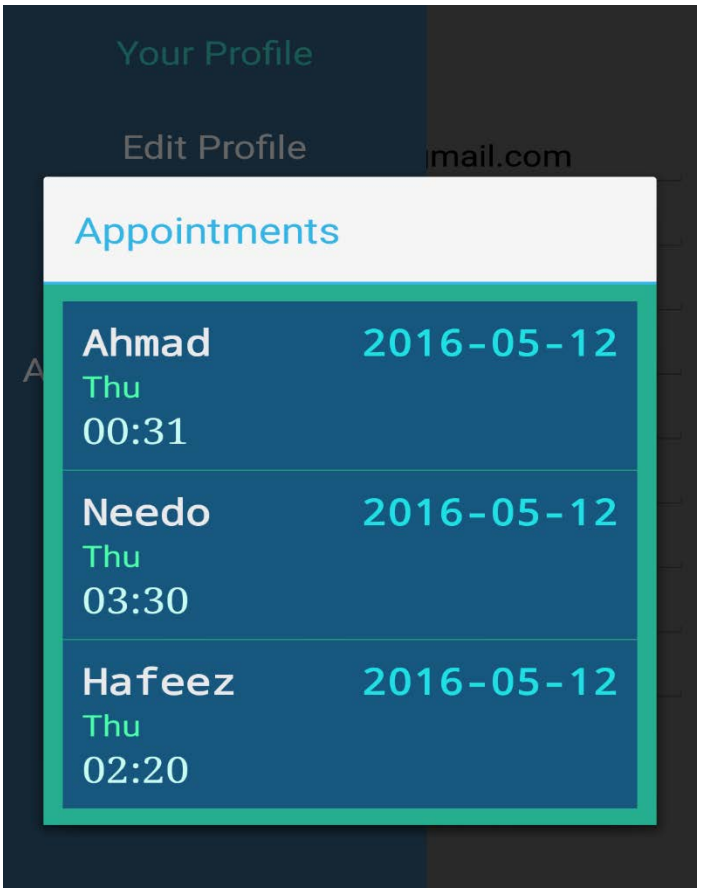


Figure 30-Doctor's Appointments

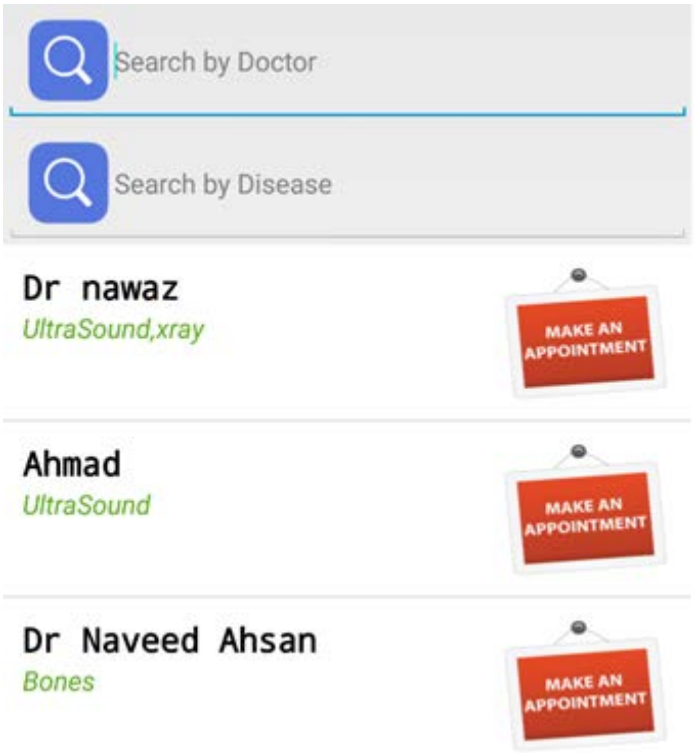


Figure 31-Search Doctor for Patient

4.3 Detailed Description of Components

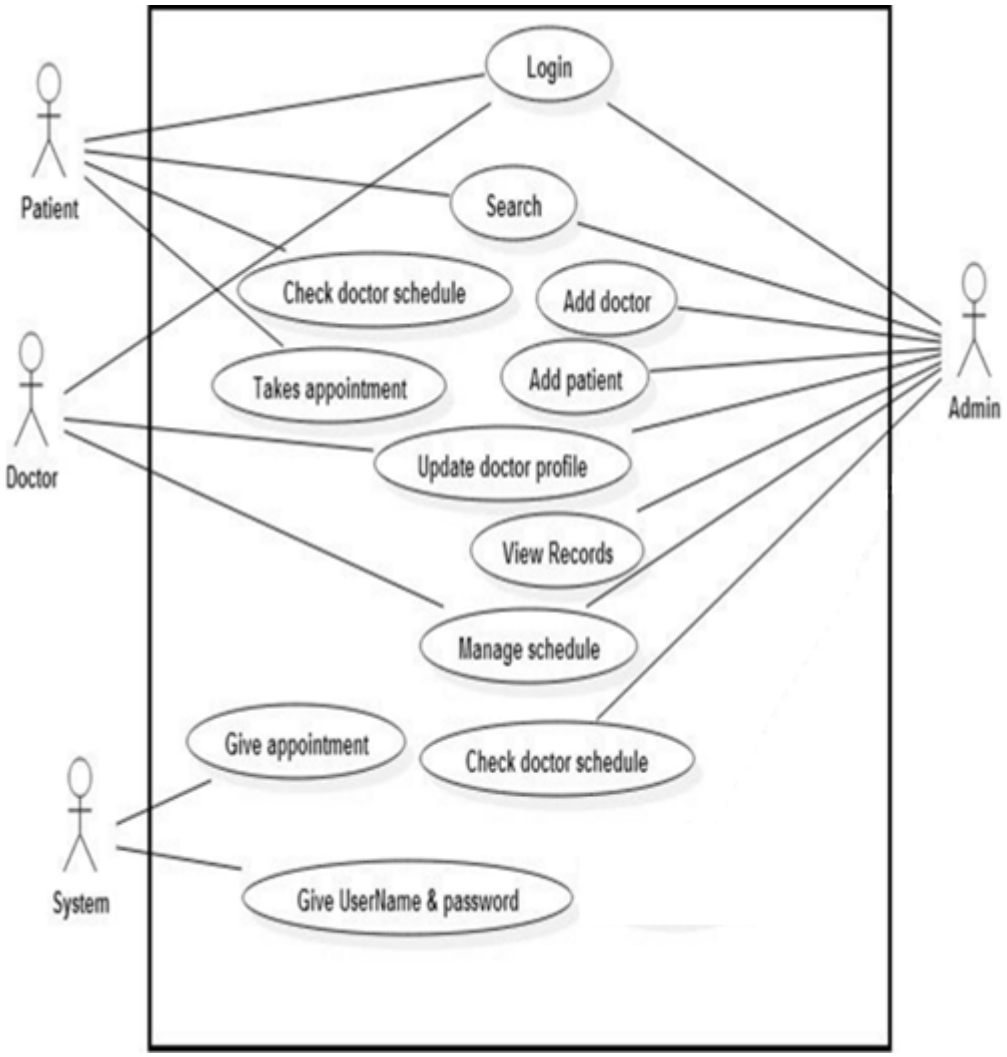


Figure 32-Usecase Diagram

Use case	**1**
Use case title	Login
Actors:	Patients, Doctors and Admin
Description:	Login allows the admin to control all the records of all the patients and doctors. Login allows the doctor to see his/her profile. His/her profile will contain all his academic information including his/her qualification, experience and specialty.
Preconditions:	The username and password of patients/doctors cannot change by the admin and a patient/doctor cannot access the admin section.

Post conditions:	When the admin logins he/she can change the patient record. When the patient/doctor logins he/she can visit his/her profile.
Uses:	Visit and manage
Normal course of events:	The use case starts when the admin or patient or doctor enters his own user name and password.
Alternative path:	If patient/doctor does not enter username and password and enter wrong password, he/she cannot see his/her profile. If the Admin does not enter username and password or enter wrong password he/she cannot access to control panel.
Exceptions	The system will not allow the patient/doctor to see his/her profile if he/she has no account. The system will not allow the Admin to change or insert record if he is not authorized user.

Table 1-Usecase (Login)

Use case	**2**
Use case title	Search
Actors:	Patient and Admin
Description:	Search allows a patient to browse all the information about doctor such as Specialty, Qualifications and Experience. Search allows the admin to view all the records of all patients and doctors.
Preconditions:	The patient will have to register by username and password.
Post conditions:	The patient's records are visited by the Admin and by the patient.
Uses:	Searching
Normal Course of events:	When the Admin or the patient wants to search something, the use Case start i.e. the system should allow the users to visit different records.
Exceptions	The system will not allow the patient to search any other patient Information. The system will not allow the Admin to change patient's Username and password.

Table 2-Usecase (Search)

Use case	**3**
Use case title	Approve/ Remove Doctor
Actors:	Admin
Description:	The Admin will approve the new doctor to APP catalog and will remove already added Doctor.
Preconditions:	The doctor record has not been maintained (saved) in database.
Post conditions:	The patient record will be saved in APP catalog.
Alternative path:	The Admin does not add a new doctor. If the Admin does not add a new doctor, then the app Catalog does not contain the doctor record.
Exceptions:	None

Table 3-Usecase (Approve/ Remove Doctor)

Use case	**4**
Use case title	View/ Remove patient
Actors:	Admin
Description:	The Admin Will See the required information of each patient to catalog and will remove it.
Preconditions:	The system will not allow the already registered user to catalog.
Post conditions:	The patient information is temporarily stored in the system.
Normal course of events:	The use case starts when the Admin wants to view/remove the patient. The system will add the username and password to system catalog.
Alternative path:	None
Exceptions:	The system will not allow the Admin to view/remove those patients who is already registered. System will not allow the Admin to give appointment to those patients who has already taken one appointment with one doctor.

Table 4-Usecase (View/ Delete Patient)

Use case	**5**
Use case title	Update Doctor Profile
Actors:	Doctor
Description:	Doctor will update his profile if the doctor improves his/her old record.
Preconditions:	The system contains all the doctor's records.
Post conditions:	The doctor's records are being changed (updated).
Normal course of events:	When a doctor improves qualification or experience the Admin will replace the old record by the new one. The system will maintain the new record.
Alternative path:	Doctor does not change or update doctor's Records. If the Doctor does not update the doctor record the system contain the doctor old result.
Exceptions:	The system will not allow the Admin to update the doctor record if the doctor already updated it.

Table 5-Usecase (Update Doctor Profile)

Use case	**6**
Use case title	View Profile
Actors:	Admin
Description:	When the admin wants to view profile, the system will allow to view profiles of Doctors and Patients.
Preconditions:	The system will have separate record for each doctor/patient.
Post conditions:	The profile of each doctor will be stored permanently and patient's profile will store temporarily.
Normal course of events:	The use case starts when the Admin wants to check or see the profile of doctor/patient.
Alternative Path:	None
Exceptions:	The system will not store any profile. If Admin does not want to save a doctor's record, then the system will not save prfile in catalog.

Table 6-Usecase (View Records)

Use case	**7**
Use case title	Manage schedule
Actors:	Doctor
Description:	When Doctor wants to manage schedule then the system will allow him to update schedule on weekly bases.
Preconditions:	The system will have the schedule of each doctor.
Post conditions:	The system will store the schedule of each doctor in app database.
Normal course of events:	The use case starts when the Doctor wants to change the schedule, then system will direct to generate the Schedule of each doctor.
Alternative Path:	None
Exceptions:	If the doctor does not want to change his/her schedule then the admin could not change his/her schedule.

Table 7-Usecase (Manage Schedule)

Use case	**8**
Use case title	Check Doctor's Schedule
Actors:	Patient
Description:	Patient can check doctor's schedule by clicking the doctor's schedule but the system will show the specific doctor's schedule.
Preconditions:	The system will have all the information required for doctor's schedule.
Post conditions:	Now system will allow patient to check the schedule.
Normal course of events:	The use case starts when a patient wants to take appointment and check the doctor's schedule.
Alternative Path:	If patient does not want to check doctor's schedule then the system will not show any schedule.
Exceptions:	The system will not allow the patient to check any other schedule concurrently. The system will not allow the patient to see any doctor's schedule if he/she has no account.

Table 8-Usecase (Check Doctor's Schedule)

Use case	9
Use case title	Take Appointment
Actors:	Patient
Description:	The use case allows Patient to take an appointment with specific doctor at a time.
Preconditions:	The system will allow that patient who has his/her own account.
Post conditions:	The system will allow doctor appointment to authorized patient.
Normal course of events:	The use case starts when authorized patient/user wants to take an appointment.
Alternative Path:	If patient does not need an appointment then the system will only allow to visit the site.
Exceptions:	The system will not allow the patient to take an appointment if he/she has no account.

Table 9-Usecase (Take Appointment)

Use case	10
Use case title	Give Appointment
Actors:	System
Description:	The system allows the authorized user to give an appointment.
Preconditions:	The system will prevent unauthorized user from any appointment.
Post conditions:	Now the system gave appointment to user.
Normal course of events	The use case starts when the patient/user create an account and wants to take an appointment with specific doctor, then the system will allow him/her.
Alternative Path:	The system does not allow to take an appointment.
Exceptions:	None

Table 10-Usecase (Give Appointment)

Use case	11
Use case title	Give username & password
Actors:	System
Description:	It allows the patient/doctor to create his/her own account and store in catalog.
Preconditions:	The system does not give user name and password who has not valid email address.

Post conditions:	The username and password will store in APP catalog.
Normal course of events:	The use case starts when a patient/doctor wants to create an account.
Alternative Path:	None
Exceptions:	The system will not allow username and password to already registered user.

Table 11-Usecase (Give Username & Password)

4.4 Reuse and relationships to other products

Medroid is not based on any previous systems neither it's an extension of any other applications at any level. But it can be evolved into a bigger and more complex system with more features and functionality. The application can also be enhanced to further include more activities such as online chatting, report sharing etc.

4.5 Design decisions and tradeoffs

Design decisions and tradeoffs which help users to understand about the product have been explained in detail in this document (consult section 2, 3 and 6 for details).

4.6 Pseudo code for Components

4.6.1 Classes

4.6.1.1 Doctor

4.6.1.1.1 Variables

Name, Email, Clinic Address, Username, Password, Designation, Qualification, City, Cell no.

4.6.1.1.2 Methods

Signup ( ), Login( ), Check_Appt( ), Edit_Profile( ), Aval_Info( ).

4.6.1.1.3 Pseudocode

```
Signup ( ) {
//Register as doctor
Fill the info and click register
Registration Status
Successful/Unsuccessful registered
}
Select category, enter username and password.
Login ( ) {
```

If (Username and password matches with the database entry)
then {login successful}
Else {error msg}
}
Check_Appt ( ) {
Approve/reject incoming appointments.
}
Edit_Profile ( ) {
Edit your info in the database.
}
Edit_Aval_Info ( ) {
Edit your aval info(time, days and no of patients)
}
View_appointments ( ) {
Select Date
View Appointments of doctor for selected date
}
Cancel_Appointments ( ) {
Select Date
Cancel Appointments of selected date and Send sms to all Patients
}

4.6.1.2 Patient

4.6.1.2.1 Variables

Name, Email,Address, Username, Password, Disease, Contact no.

4.6.1.2.2 Methods

Signup ( ), Login ( ), Search_Appt ( ).

4.6.1.2.3 Pseudo code

Signup ( ) {
//Register as patient
Fill the info and click register
Registration Status
Successful/Unsuccessful registered
}
Select category, enter username and password.
Login ( ) {

```
        If (Username and password matches with
        the database entry) then {login successful}
        Else {error msg}
}

Search_Appt ( ) {
        //Search doctors by name and specialty.
        //Fill in time, date and day and make
appointments.
}
```

4.6.1.3 Admin

4.6.1.3.1 Variables

Name, Email, Username, Password.

4.6.1.3.2 Method

Login ( ), Approve_Dr ( ), Del_Dr ( ), Del_Patient ( ), Add_city ( ), Add_clinic ( ), Manage_info ( ), Manage_aids ( ).

4.6.1.3.3 Pseudo code

```
Login ( ) {
If (Username and password matches with the
database entry)
T       hen {login successful}
Else {error msg}
}

Approve_Dr ( ){
        View doctor's profile and approve/reject.
}
Del_Dr()/Del_Patient ( ) {
        //Delete the profile
}
```

4.6.1.4 Signup

4.6.1.4.1 Variables

Email, Username, Password

4.6.1.4.2 Method

Signup ( )

4.6.1.4.3 Pseudocode

```
Signup ( ) {
        Signup as doctor or patient.
        Fill info.
        Submit it for approval from admin.
}
```

4.6.1.5 Login

4.6.1.5.1 Variables

Username, Password

4.6.1.5.2 Method

Login ( ), Get_status ( ).

4.6.1.5.3 Pseudocode

```
Select category, enter username and password.
Login ( ) {
If (Username and password matches with the
database entry)
        then {login successful}
Else {error msg}
}
```

4.6.1.6 Online Database

4.6.1.6.1 Variables

Server_name

4.6.1.6.2 Method

Server_status ( )

4.6.1.7 Appointments

4.6.1.7.1 Variables

Date, Time, Patient_name, Dr_name.

4.6.1.7.2 Method

Make_appt ( ), Show_appt ( ).

4.6.1.7.3 Pseudo code

```
Make_appt ( ) {
        //Make appointments for the given time,date
and day by the patient.
}
```

```
Show_appt ( ) {
        //Show all appointments made by the patient
and approved by the
        doctor.
}
```

Chapter 5: Project Analysis and Evaluation

5.1 Test Cases

Test Case Name	**Incorrect Admin Login**
Test Case Number	1
Description	Incorrect Credentials for Admin Login
Preconditions	Application should be open and connected to the Internet
Input values	Incorrect or Empty User Name / Password fields for Admin
Steps	Input Incorrect Credentials of username / Password Click on Login Check for the status on the Application
Expected output	Incorrect Credentials Error
Actual Output	Admin Login Failed

Table 12-Test Case (Incorrect Admin Login)

Test Case Name	**Correct Admin Login**
Test Case Number	2
Description	Correct Credentials for Admin Login
Preconditions	Application should be open and connected to the Internet Category/ Profile for admin is selected Category Should be selected as Admin
Input values	Correct User Name / Password Values for Admin ID
Steps	Open Application Input correct Credentials of username / Password Click on Login Check for the status on the Application
Expected output	Should login successfully
Actual Output	Successfully Logged in

Table 13-Test Case (Correct Admin Login)

Test Case Name	**Incorrect Patient Login**
Test Case Number	3
Description	Incorrect Credentials for Patient Login
Preconditions	Application should be open and connected to the Internet
Input values	Incorrect or Empty Username / Password fields
Steps	Input incorrect Credentials of username / Password for Patient Account Click on Login Check for the status on the Application
Expected output	Incorrect Credentials Error
Actual Output	Patient Login Failed

Table 14-Test Case (Incorrect Patient Login)

Test Case Name	**Correct Patient Login**
Test Case Number	4
Description	Correct Credentials for Patient Login
Preconditions	Application should be open and connected to the Internet Patient's Profile Should have been created Category Should be selected as Patient
Input values	Correct Username / Password values are entered
Steps	Input correct Credentials of username / Password for Patient Account Click on Login Check for the status on the Application
Expected output	Should login as Patient
Actual Output	Successfully logged in as patient

Table 15-Test Case (Correct Patient Login)

Test Case Name	**Incorrect Doctor Login**
Test Case Number	5
Description	Incorrect Credentials for Doctor Login
Preconditions	Application should be open and connected to the Internet
Input values	Incorrect or Empty Username / Password Fields
Steps	Input Credentials of username / Password for Doctor's Account Click on Login Check for the status on the Application
Expected output	Incorrect Credentials Error
Actual Output	Doctor Login Failed

Table 16-Test Case (Incorrect Doctor Login)

Test Case Name	**Correct Doctor Login**
Test Case Number	6
Description	Correct Credentials for Doctor Login
Preconditions	Application should be open and connected to the Internet Doctor's Profile Should have been created earlier Category/ Profile Should be selected as Doctor
Input values	Correct Username / Password Fields
Steps	Input correct Credentials of username / Password for Doctor's Account Click on Login Check for the status on the Application
Expected output	Should login successfully as Doctor
Actual Output	Successfully logged in as Doctor

Table 17-Test Case (Correct Doctor Login)

Test Case Name	**Incorrect Category/ Profile Selection**
Test Case Number	7
Description	Incorrect category selection for Admin/Doctor/Patient
Preconditions	Application should be open and connected to the Internet
Input values	Incorrect Category with Correct Username / Password Fields is selected
Steps	Select incorrect category/ profile Input correct Credentials of username/Password for Admin, Doctor or Patient Click on Login Check for the status on the Application
Expected output	Should not login successfully as Doctor
Actual Output	Invalid username/ Password Error

Table 18-Test Case (Incorrect Category/ Profile Selection)

Test Case Name	**Doctor's Profile Approval**
Test Case Number	8
Description	Admin Approves the Doctor's newly created Profile
Preconditions	Application should be open and connected to Internet Doctor's Profile Should have been created earlier User is Logged in as Admin
Input values	Click on Approve button of Doctor's Profile
Steps	Open Doctor's newly Created profile Click on approve button Wait for the changed status
Expected output	Doctor's profile should be approved
Actual Output	Doctor's profile is approved

Table 19-Test Case (Doctor's Profile Approval)

Test Case Name	**View/ Delete Patient's profile**
Test Case Number	9
Description	Admin can view and delete the Patient's Profile
Preconditions	Application should be open and connected to Internet Patient's Profile Should have been created earlier User is Logged in as Admin
Input values	Click on Patient's Profile
Steps	Open Patient's profile Click on delete button Wait for the changed status
Expected output	Patient's profile should be removed from data base
Actual Output	Patient's profile was removed successfully

Table 20-Test Case (View/ Delete Patient's profile)

Test Case Name	**Make Appointment**
Test Case Number	10
Description	Patient looks doctor's profile and selects Doctor and make appointment with him
Preconditions	Application should be open and connected to Internet Doctor's profiles have been created and approved by Admin User is Logged in as Patient
Input values	Select Doctor and click on Make Appointment
Steps	Select Doctor's profile Click on Make Appointment Select Date/Time/Day Click on Submit Button
Expected output	New Appointment for doctor should be submitted
Actual Output	Appointment submitted Successfully

Table 21-Test Case (Make Appointment)

Test Case Name	**Cancel Appointment by Patient**
Test Case Number	11
Description	Patient Can cancel appointment request made by him/ herself
Preconditions	Application should be open and connected to Internet User is Logged in as Patient Patient has made appointment request to Doctor
Input values	Click on cancel appointment
Steps	Click on cancel appointment Wait for the response
Expected output	Appointment should be cancelled and not visible next time
Actual Output	Appointment cancelled Successfully

Table 22-Test Case (Cancel Appointment by Patient)

Test Case Name	**Cancel Appointment by Doctor**
Test Case Number	12
Description	Doctor can cancel appointment requested by patient for selected date if he is busy at that day
Preconditions	Application should be open and connected to Internet User is Logged in as Doctor Patient has made appointment request to Doctor
Input values	Click on Cancel appointment
Steps	Click on Cancel Appointment Button Select Date Wait for the response
Expected output	Appointment should be Cancelled and SMS should be sent to all Patients
Actual Output	Appointment Cancelled and SMS sent to all Patients Successfully

Table 23-Test Case (Cancel Appointment by Doctor)

Test Case Name	**Edit Profile by Doctor**
Test Case Number	13
Description	Doctor Edits his/ her own Profile
Preconditions	Application should be open and connected to Internet User is Logged in as Doctor
Input values	Click on Edit Profile
Steps	Open setting menu Click on Edit profile Edit information of Doctor which you want to change Click on Update Wait for the response
Expected output	Doctor's profile should be Updated
Actual Output	Doctor's Profile updated successfully

Table 24-Test Case (Edit Profile by Doctor)

Test Case Name	**Change Availability Information of Doctor**
Test Case Number	14
Description	Doctor can edit availability information
Preconditions	Application should be open and connected to Internet Doctor's Profile has been created through signup User is Logged in as Doctor
Input values	Click on Availability Information
Steps	Open setting menu Click on Availability Information Set Availability Information of Doctor i.e. Time/ Date/ Day Click on Update Wait for the response
Expected output	Doctor's Availability Information should be Updated
Actual Output	Doctor's Profile updated successfully

Table 25-Test Case (Change Availability Information of Doctor)

Test Case Name	**View Appointment of Doctor**
Test Case Number	15
Description	Doctor can View Appointments of any Date
Preconditions	Application should be open and connected to Internet Doctor's Profile has been created through signup User is Logged in as Doctor
Input values	Click on View Appointments
Steps	Open setting menu Click on View Appointments Select Date Wait for the response
Expected output	Doctor's Appointments for selected Date should be Visible
Actual Output	Doctor's Appointments for selected Date were visible

Table 26-Test Case (View Appointment of Doctor)

Chapter 6: Future work

At present the database for the system is designed to use it, primarily as a data source for this system. But in future, by the addition of more attributes in features to database the same database can easily be used for bigger application.

- If doctors are changed then system can be updated easily.
- Patient's history will be available to Doctor.
- Patients will be able to make online Payment for Booking Appoints/ Dues.
- Patients will be able to chat online with Doctor.
- Doctor patient online transfer of health related document.
- Hardware devices will be added with android application.

Chapter 7: Conclusion

As Bachelors Students, developing a fully implemented system for Medroid was a new experience for us, especially since development in this field (i.e. the field of android mobile application). But the successful completion of this project has given us the confidence and knowledge to work the practical field as professional developers.
During the development of the following project we got a chance to practically implement what we have learned during our bachelors program. While doing our project we have achieved the several benefits. Some of them which will help us in future as a successful professional are as follows.

- Project management and scheduling.
- Analyzing a system and collecting data.
- Learned how to design software.
- In testing phase learned how to debug and new ways of implementing the test.
- The most important one, which is not bothered too much in our education environment, is how to document properly.
- Finally we choose Android application and select JAVA and got expertise in it.

This will help us to enter into industry and prove our self.

Bibliography

- M. Shaw and D. Garlan. Software Architecture: Perspectives on a Emerging Discipline. Prentice Hall, Englewood Cliffs, NJ, 1996

- F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal.Pattern-Oriented Software Architecture. A System of Patterns. John Wiley & Sons Ltd., Chichester, UK, 1996

- Lecture slides on Architecture by David Garlan, see http://www-2.cs.cmu.edu/afs/cs/academic/class/17655-s02/www/

- Lecture slides on Architecture by Marc Roper and Murray Wood, see https://www.cis.strath.ac.uk/teaching/ug/classes/52.440/

- Lecture slides by Ian Sommerville, Software Engineering, 6th edition

- http://michaelbach.de/fract/index.html

- http://www.colormax.org/colorblind-test.htm

- Dean Farnsworth, Journal Of The Optical Society Of America, volume 33, number 10, October 1943- The Farnsworth-Munsell 100-Hue and Dichotomous Tests for Color Vision. Department of Psychology, New York University, New York, New York

Appendix A: Glossary

Definitions:

Constraint	A limitation or restriction imposed on a function.
Android device	An Android device is a mobile phone built on a mobile computing platform, with more advanced computing ability and connectivity than a feature phone.
Web Link	URL of a web-page.
Scramble options	Permute answer options

Abbreviations and Acronyms

FAQ	Frequently Asked Question
GUI	Graphical User Interface
HTTP	Hyper Text Transfer Protocol
URL	Uniform Resource Locater
SQL	Structured Query Language
SRS	Software Requirements Specification
SDS	Software Design Specification
DBMS	Data Base Management System
IEEE	Institute of Electrical and Electronics Engineers
SSL	Secure Socket Layer
UML	Unified Modeling Language
LAN	Local Area Network