Chapter 1

Introduction

India today has a population of over 140 crores. India, a growing country, is experiencing traffic congestion due to its population. More than 40 percent of patients going to the hospital cause this kind of traffic jam. Those who suffer from the traffic jams they cause are commuters, laborers, school-going children, etc. Apart from that, it is the patients like them who suffer from the traffic congestion caused by them. I have designed this app to control this.

In recent number of year's instant use of communication tools like mobile devices have been successfully introduced to support communication and collaboration processes in work environments. Research suggests that the use of mobile applications also increasing rapidly day by day due to social presence and awareness within a collaborative group. Now a day's all over the world people are more interested in various type of mobile application, and dynamical informative apps are one of the top most choice for its very significant matter. This kind of apps helps to learn easily and patiently. Android devices come in all kinds of sizes, with all sorts of features, and all sorts of prices. Each version of Android is named by dessert, and the most recent version of Android is Tiramisu with Android in control of mobile experience. That's why now a day's android based application development is also one of the top most choice of the developers in many areas. Here we use android to develop an android hospital management system app to take care of family health condition.

1.1 Overview of the Project :

"Hospital Management System" is an application that helps anyone/user to get information about their family. When a user selects a particular patient listed in this app, all information about that patient of hers will be retrieved from the app. This app contains information, photos, doctor list, diet list, emergency calls, emergency notes and more. Touch your Android device to get the current state and calculate the estimated time to reach the selected location. Android is designed to work on a wide variety of devices, from phones to tablets to TVs. As a developer, your device choices make the potential users of your apps huge. This app is successful on all these devices. It should allow for variations in the functionality and provide a flexible user interface that adapts to different screen configurations. Our app's user interface is everything a user can see and interact with. Android provides various pre-built he UI components such as structured layout objects and UI controls that you can use to create the graphical user interface of your app. Android also provides additional his UI modules for special interfaces such as menus.

1.2 Problem Description :

Our project **Hospital Management system** includes registration of patients, storing their details into the system, and also booking their appointments with doctors. Our software has the facility to give a unique id for every patient and stores the details of every patient and the staff automatically. User can search availability of a doctor and the details of a patient using the id. The Hospital Management System can be entered using a username and password. It is accessible either by an administrator or receptionist. Only they can add data into the database. The data can be retrieved easily. The interface is very user-friendly. The data are well protected for personal use and makes the data processing very fast.

Chapter 2

SYSTEM ANALYSIS

Existing System :

Our project Hospital Management system includes registration of patients, storing their details into the system, and also booking their appointments with doctors. Our software has the facility to give a unique id for every patient and stores the details of every patient and the staff automatically. User can search availability of a doctor and the details of a patient using the id. The Hospital Management System can be entered using a username and password. It is accessible either by an administrator or receptionist. Only they can add data into the database. The data can be retrieved easily. The interface is very user-friendly. The data are well protected for personal use and makes the data processing very fast.

Algorithm :

Chepeastlink Algorithm :

Instead of starting at a reference vertex and moving to the nearest neighbor at each step, we “start in the middle.” That is, if there is a cheap edge that you know you will want to use eventually make sure you use it

DISADVANTAGE :

- Many problems with meeting doctors
- Doctors cannot be ordered
- Separately for each hospital
- It is difficult to listen to the advice of doctors
- Medicines cannot be purchased without a paper prescription
- Private Hospital management manage the application

PROPOSED SYSYTEM :

With the large amounts of data, people involved and innumerable processes, a hospital is definitely an ideal candidate for data management software. If hospitals are to run efficiently, provide top line care, ensure patient and other data confidentiality, and work seamlessly – they cannot hope to do so without an effective Hospital Management System. Reduced human intervention for paperwork, less paperwork, reduced staff headcount for jobs that can be easily managed within the HMS, speedier processes, reduction of errors, and data privacy and safety – are just some of the benefits of a Hospital Management System.

Advantages :

- These applications are easy to use by the user
- Can order in Doctors
- All hospital used this application
- Easy to communicate Doctors
- Medicines can be purchased without a paper prescription
- Government or Service Trust manage this application

Algorithm :

Knuth-Plass Algorithm :

Knuth's main purpose in describing Algorithm X was to demonstrate the utility of dancing links. Knuth showed that Algorithm X can be implemented efficiently on a computer using dancing links in a process Knuth calls "DLX". DLX uses the matrix representation of the exact cover problem, implemented as doubly linked list of the 1s of the matrix: each 1 element has a link to the next 1 above, below, to the left, and to the right of itself. (Technically, because the lists are circular, this forms a torus). Because exact cover problems tend to be sparse, this representation is usually much more efficient in both size and processing time required. DLX then uses dancing links to quickly select permutations of rows as possible solutions and to efficiently backtrack (undo) mistaken guesses

Chapter 3

System Specification

3.1 Hardware Requirements :

Processor : core i3 or above
RAM : 4 GB
Hard Disk Terminal Machine : 20 GB
Server Machine : 1 TB
Android version : 10 or more

Software Requirements :

Operating System : Windows 7 or above
Server : MySQL Server
IDE * : Android Studio
Front – End : XML and UI
Back - End : Java , Kotlin
Framework : Spring Boot

CHAPTER 4
Software Description

4.1 Front-End:

4.1.1 Basic Android Overview

Android is a comprehensive open-source platform designed for mobile devices. It is championed by Google and Open Handset Alliance. The goal of the alliance is to “accelerate innovation in offer consumers a richer, less expensive, and better mobile experience.” Android is the vehicle to do so. As such, Android isrevolutionizing the mobile space. For the first time, it is a truly open platform that separates the hardware from the software that runs on it [2]. This allows for a muchlarger number of devices to run the same applications and creates a much richer ecosystem for developers and consumers. Fig 4.1 shows the relation between companies of open handset alliance.

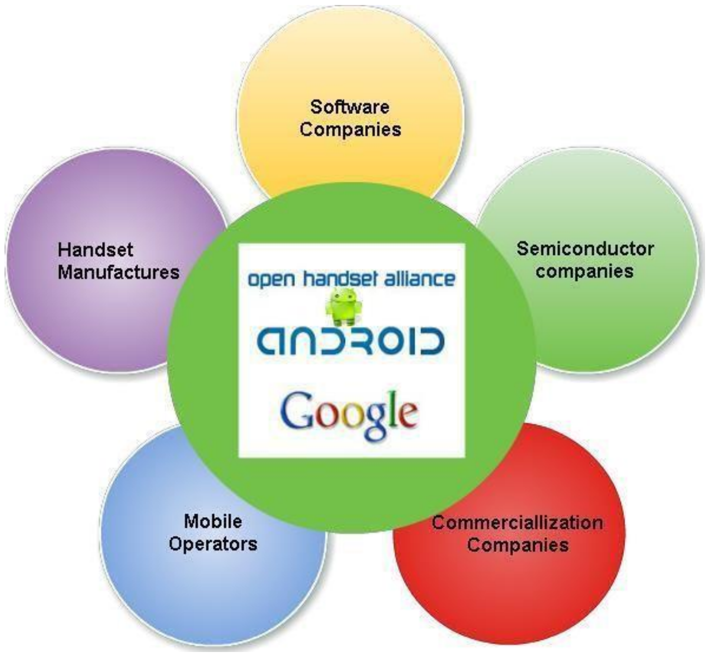


Fig 4.1: Open Handset Alliance (OHA) [2].

4.1.2Android Versions

Since April 2009, Android versions have been developed under a confectionery-themed code name and released in alphabetical order; the exceptions are versions 1.0 and 1.1 as they were not released under specific code names:

- Alpha (1.0)
- Beta (1.1)
- Cupcake (1.5)
- Donut (1.6)
- Eclair (2.0–2.1)
- Froyo (2.2–2.2.3)
- Gingerbread (2.3–2.3.7)
- Honeycomb (3.0–3.2.6)
- Ice Cream Sandwich (4.0–4.0.4)
- Jelly Bean (4.1–4.3.1)
- KitKat (4.4–4.4.4)
- Lollipop (5.0–5.0.2)
- Marshmallow(6.0-6.0.1)
- Nougat(7.0-7.1.2)
- Oreo(8.0-8.1)
- Pie(9.0)
- Android Q (10)
- Red Velvet Cake(11)
- Snow Cone(12)
- Tiramisu(13)

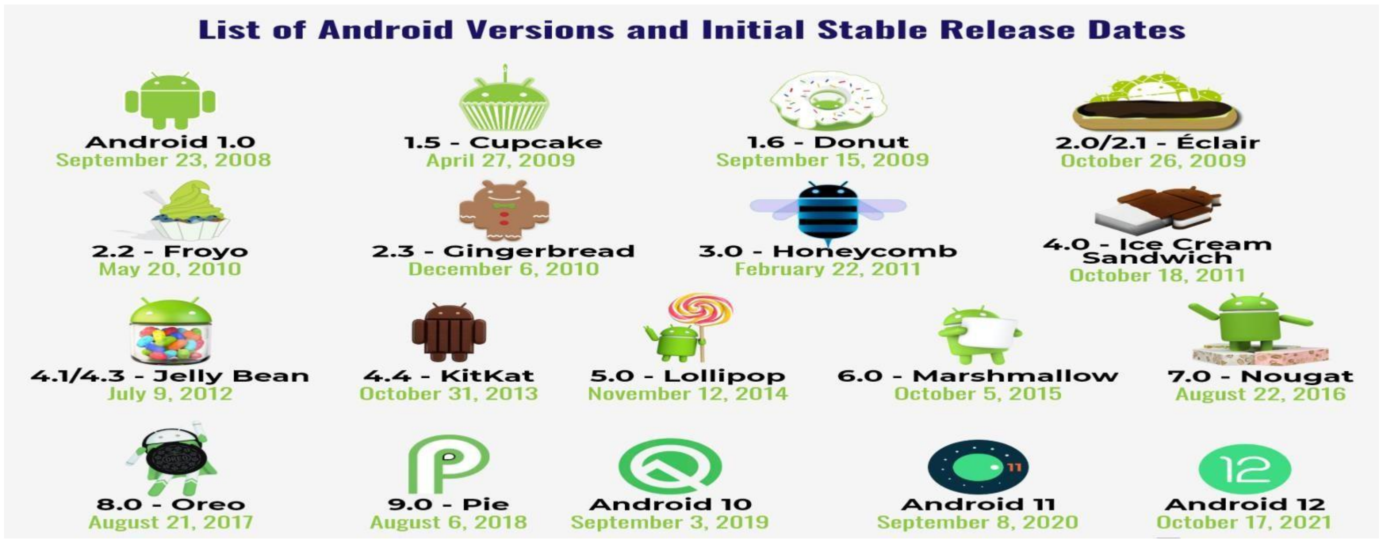


Fig 4.2: Android Versions [3].

Fig 4.2 is the Android versions with codenames. The Android version number itself partly tells the story of the software platform's major and minor releases. What is most important is the AIP level. Version numbers change all the time, sometimes because the AIPs have changed, and other times because of minor bug fixes or performance improvements [3].

4.1.3 The Android Stack

The Android operating system is like a cake consisting of various layers. Each layer has its own characteristics and purpose. The layers are not cleanly separated but often seep into each other on. Fig 4.3 shows the part of the Android stack.

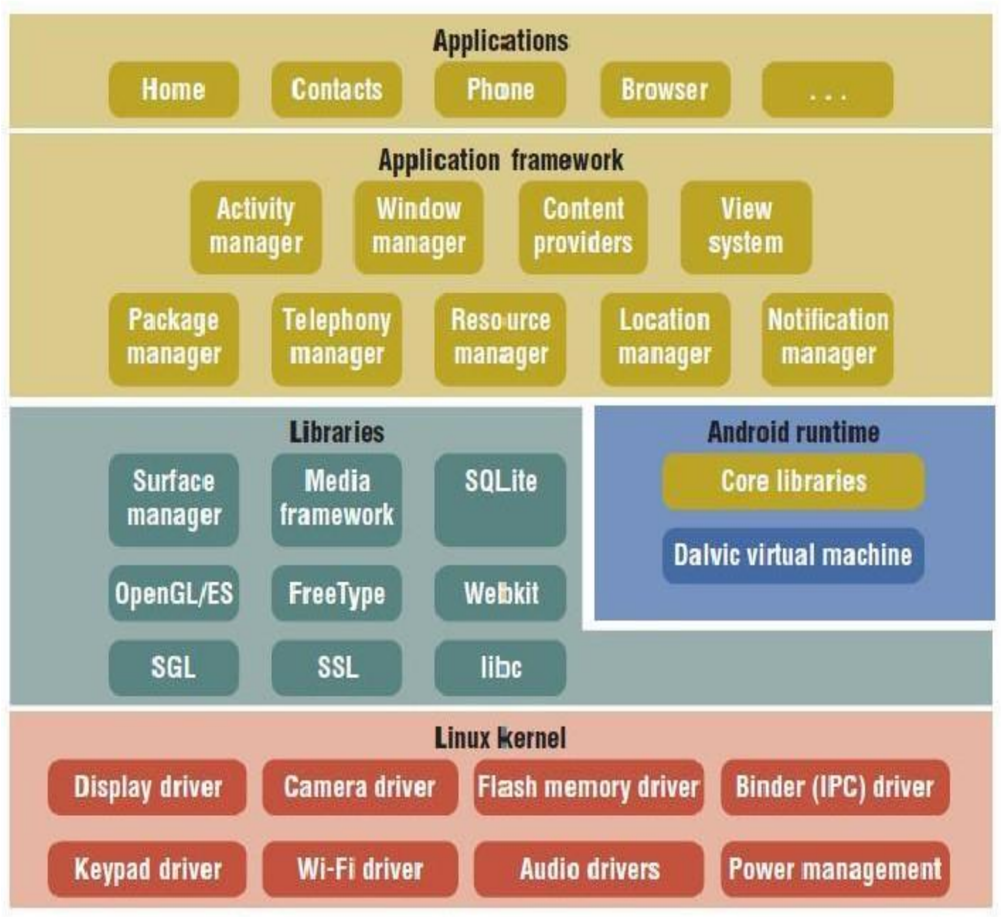


Fig 4.3: Android Stack [1].

4.1.4 Android and Java

In Java, we write our java source file, compile it into a java byte code using the Java compiler, and then run this byte code on the Java VM. In Android, things are different. We still write the Java source file, and we still compile it to Java bytecode using the same Java compiler. But at that point, we recompile it once again using the Dalvik compiler to Dalvik byte code [3]. It is this Dalvik byte code that is than executed on the Dalvik VM. Fig 4.4 illustrates this comparison between standard Java (on the list) in Android using Dalvik (on the right).

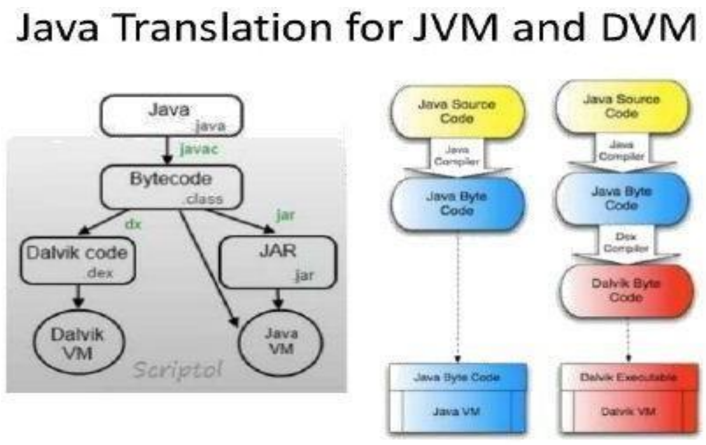


Fig 4.4: Java versus Dalvik [3].

A side effect of this is that in theory we could write Android applications in any other language that compiles down to Java byte code.

4.1.5Activities

An activity is usually a single screen that the user sees on the device at one time. An application typically has multiple activities, and the user flips back and forth among them. As such, activities are the most visible part of our application. We usually use an activity [4]. Just like a website has a "home page," an android app has a main (full screen) activity, usually the one that is shown first when we launce the application. And just like a website has to provide some sort of navigation among various pages, an Android app should do too same. On the web, we can jump from a page on one website to a page on another. Similarly, in Android, we could be looking at an activity of one application, but shortly after we could start another activity in a completely separate application.

4.1.6 Activity Life Cycle

Launching an activity can be quite expensive. It may involve creating a new Linux process, allocating memory for all the UI objects, inflating all the objects from XML layouts, and setting up the whole screen. Since we are doing a lot of work to launch an activity, it would be a west to just toss it out once the user leavesthat screen. To avoid this waste, the activity life cycle is managed via Activity Manager. Activity Manager is responsible for creating, destroying, and managing activities. For example, when the user starts an application for the first time, the Activity manager will create its activity and put it onto the screen. Later, when the user switches screens, the Activity Manager will move that previous activity to a holding place [4]. This way, if the user wants to go back to a previous activity, it can be started more quickly. Previous activities that the user hasn’t used in a while will be destroyed in order to free more space for the currently active one. This mechanism is designed to help improve the speed of the user interface

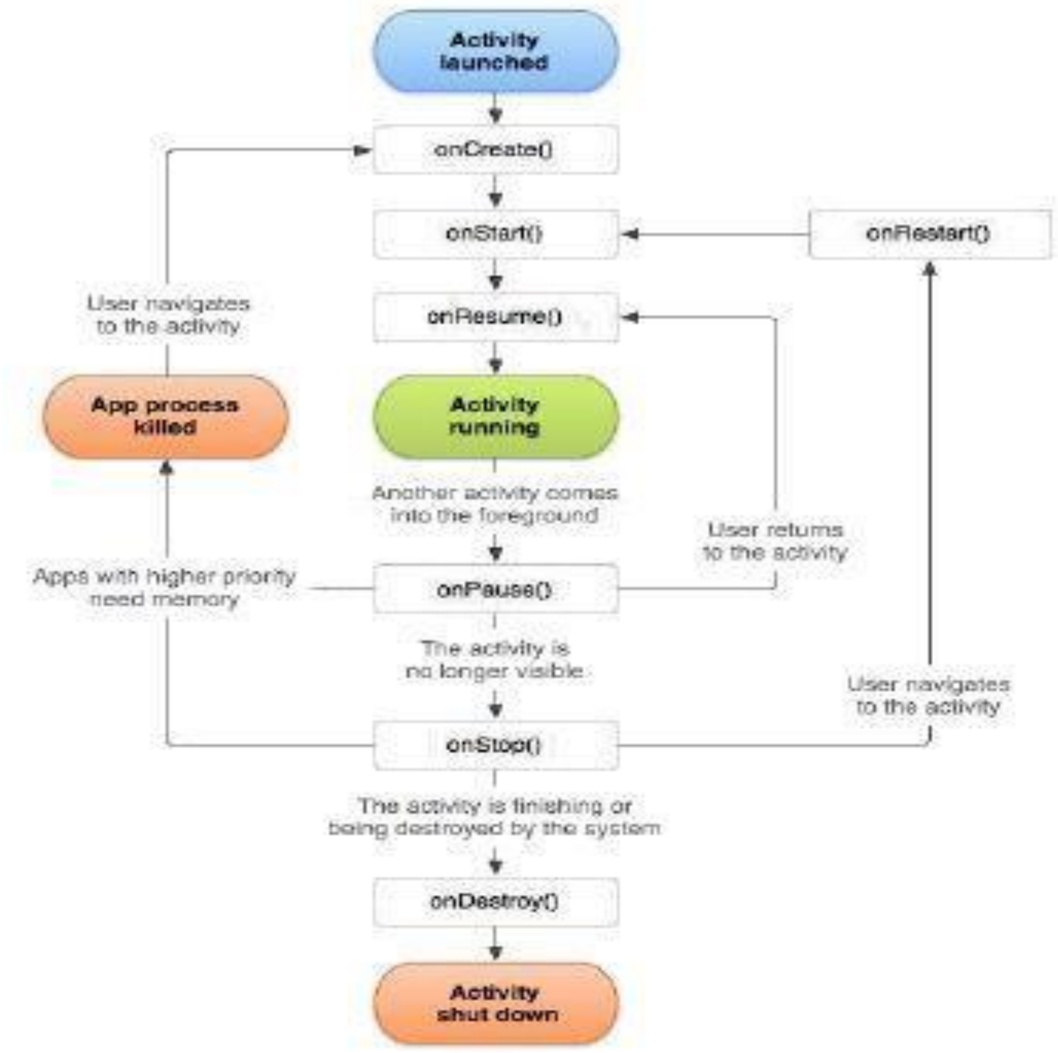

Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

Fig 2.6: Activity Life Cycle [7].

Improve the overall user experience. Programming for Android is conceptually different than programming for some other environments. Fig 4.6 shows the states that activity [7] can go through.

4.3.1 Starting state

When an activity doesn't exist in memory, it is in a starting state. It is while starting up, the activity will go through a whole set of callback methods that we asa developer have an opportunity to fill out. Eventually, the activity will be in a running state.

4.3.2 Running state

The activity in a running state is the one that is currently on the screen and interacting with the user. We also say this activity is in focus, meaning that all user interactions such as touching the screen and clicking buttons are handled by this one activity. As such, there is only one running activity at any given time.

4.3.3 Paused state

When an activity is not in focus (i.e., not interacting with the user) but still visible on the screen, we say it's in a paused state. This is not a typical scenario, because the device's screen is usually small, and an activity is either taking up the whole screen or none all. We often see this case with dialog boxes that comes up in front of an activity, causing it to become paused. All activities go through a paused state in route to being stopped.

4.3.4 Stopped state

When an activity is not visible, but still in memory, we say it's in a stopped state. Stooped activity could be brought back to the front to become a running activity again. Or, it could be destroyed and removed from memory. The system keeps activities around in a stopped state because it is likely that the user will still want to get back to those activities sometime soon.

4.3.5 Destroyed state

A destroyed activity is no longer in memory. The Activity Manager decided that this activity is no longer needed and has removed it. Before the activity is destroyed, it can perform certain actions, such as save any unsaved information.

4.1.7 Intents

Intents are messages that are sent among the major building blocks [5]. They trigger an activity to start up, tell a service to start or stop, or are simply broadcasts. Intent could be explicit or implicit. In an explicit intent, the sender clearly spells out which specific component should be on the receiving end. In an implicit intent,the sender specifies the type of receiver. Fig 4.7 shows how Intent work.

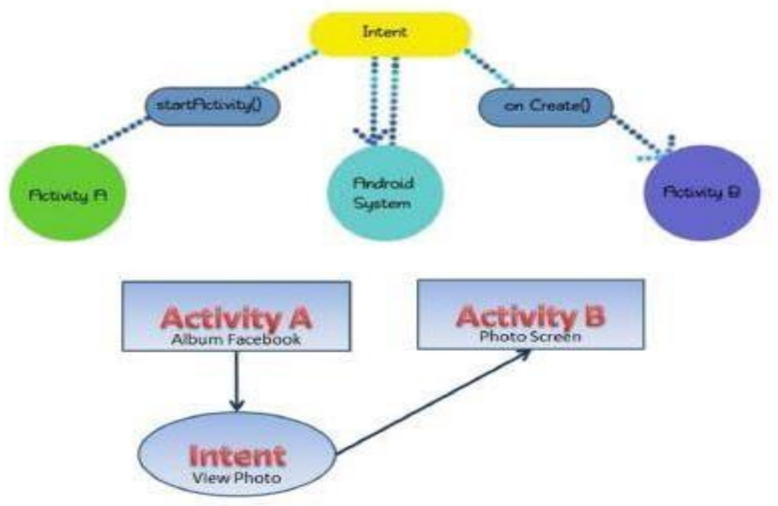


Fig 2.7: Intents

.

4.1.8 Services

Services run in the background and don’t have any user interface components. They can perform the same actions as activities, but without any user interface. Services are useful for actions that we want to perform for a while, regardless of what is on the screen. For example, it may music player to play musiceven as you are flipping between other applications. Services have a much simpler life cycle than activities. Either start a service or stop it. Also, the service life cycleis more or less controlled by the developer, and not so much by the system.

Consequently, we as developers have to be mindful to run our services so that they don’t consume shared resources unnecessarily, such as the CPU and battery. Fig 4.8 shows services life cycle and how it works.

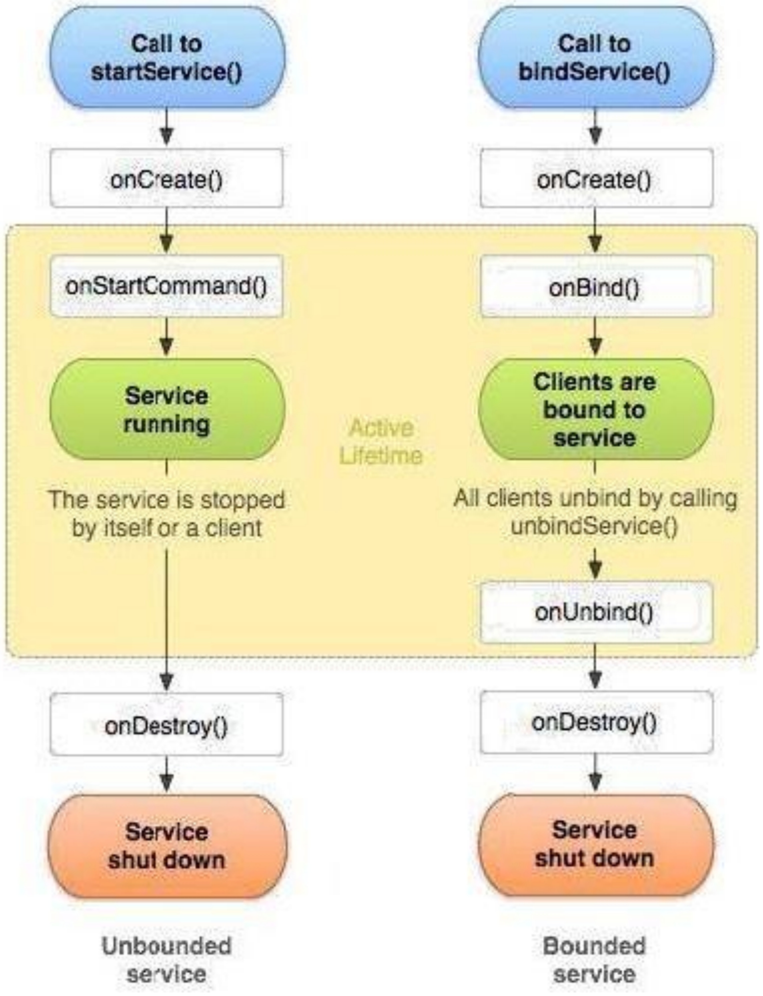


Fig 4.8: Services Life Cycle.

4.2 BACK END

4.2.1 MySQL

MySQL tutorial provides basic and advanced concepts of MySQL. Our MySQL tutorial is designed for beginners and professionals. MySQL is a relational database management system based on the Structured Query Language, which is the popular language for accessing and managing the records in the database. MySQL is open-source and free software under the GNU license. It is supported by Oracle Company. MySQL database that provides for how to manage database and to manipulate data with the help of various SQL queries. These queries are: insert records, update records, delete records, select records, create tables, drop tables, etc. There are also given MySQL interview questions to help you better understand the MySQL database.

MySQL is currently the most popular database management system software used for managing the relational database. It is open-source database software, which is supported by Oracle Company. It is fast, scalable, and easy to use database management system in comparison with Microsoft SQL Server and Oracle Database. It is commonly used in conjunction with PHP scripts for creating powerful and dynamic server-side or web-based enterprise applications. It is developed, marketed, and supported by MySQL AB, a Swedish company, and written in C programming language and C++ programming language. The official pronunciation of MySQL is not the My Sequel; it is My Ess Que Ell. However, you can pronounce it in your way. Many small and big companies use MySQL. MySQL supports many Operating Systems like Windows, Linux, MacOS, etc. with C, C++, and Java languages.

4.2.2 Framework :

4.2.2.1 Spring Boot :

- Spring Boot is an open source Java-based framework used to create a micro Service. It is developed by Pivotal Team and is used to build stand-alone and production ready spring applications. This chapter will give you an introduction to Spring Boot and familiarizes you with its basic concepts.

- Spring Boot provides a good platform for Java developers to develop a stand-alone and production-grade spring application that you can **just run**. You can get started with minimum configurations without the need for an entire Spring configuration setup.

- Spring Boot Auto Configuration automatically configures your Spring application based on the JAR dependencies you added in the project. For example, if MySQL database is on your class path, but you have not configured any database connection, then Spring Boot auto configures an in-memory database.

Spring Boot Application

The entry point of the Spring Boot Application is the class contains **@SpringBootApplication** annotation. This class should have the main method to run the Spring Boot application. **@SpringBootApplication** annotation includes Auto- Configuration, Component Scan, and Spring Boot Configuration.

If you added **@SpringBootApplication** annotation to the class, you do not need to add the **@EnableAutoConfiguration,**
@ComponentScan and **@SpringBootConfiguration** annotation.
The **@SpringBootApplication** annotation includes all other annotations.

Chapter 5

SYSTEM DESIGN

5.1 System Architecture:

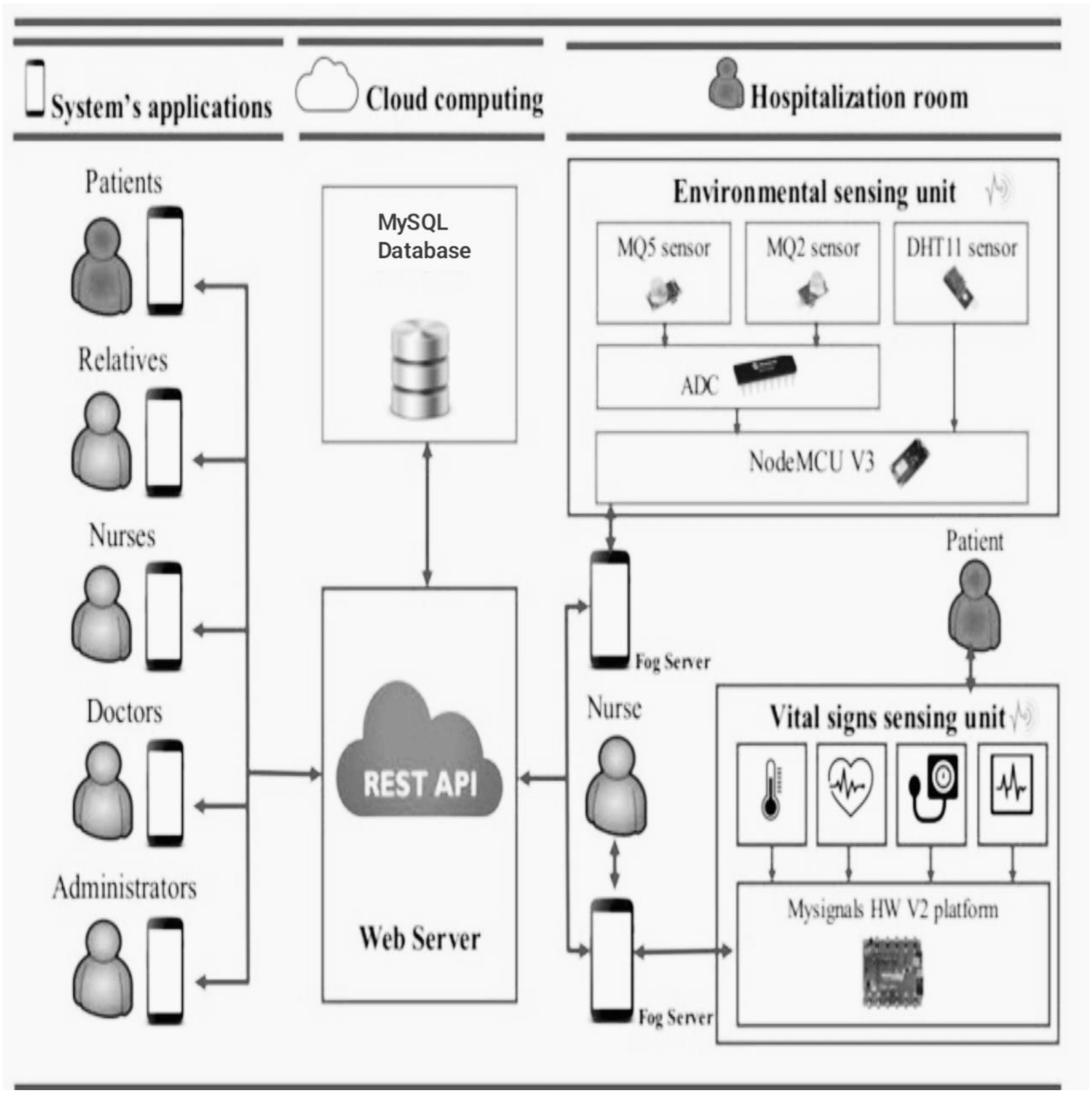


Fig 5.1 System Architecture

5.2 Data flow Diagram:

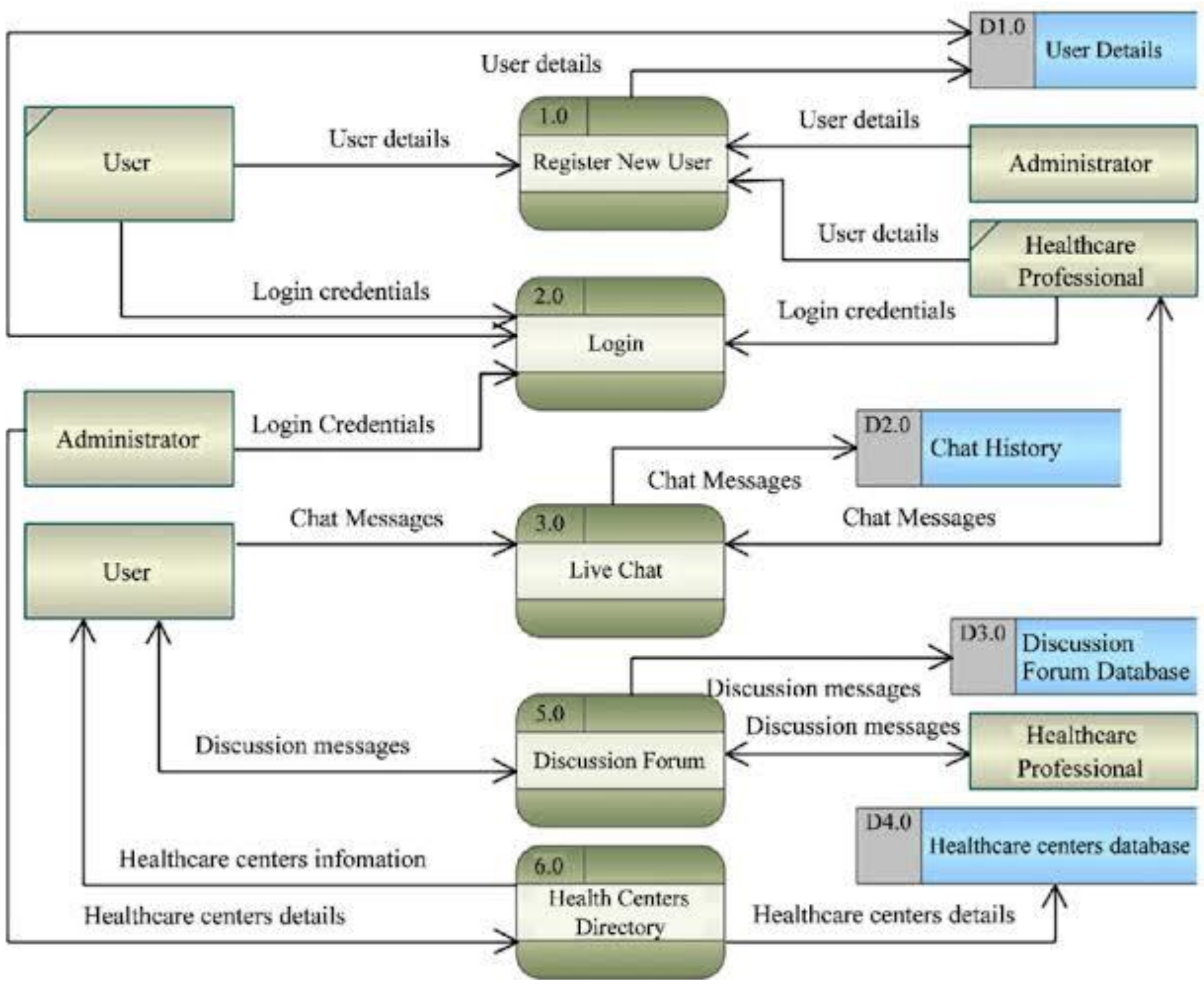


Fig 5.2 : Data flow diagram

5.3 Database Design:

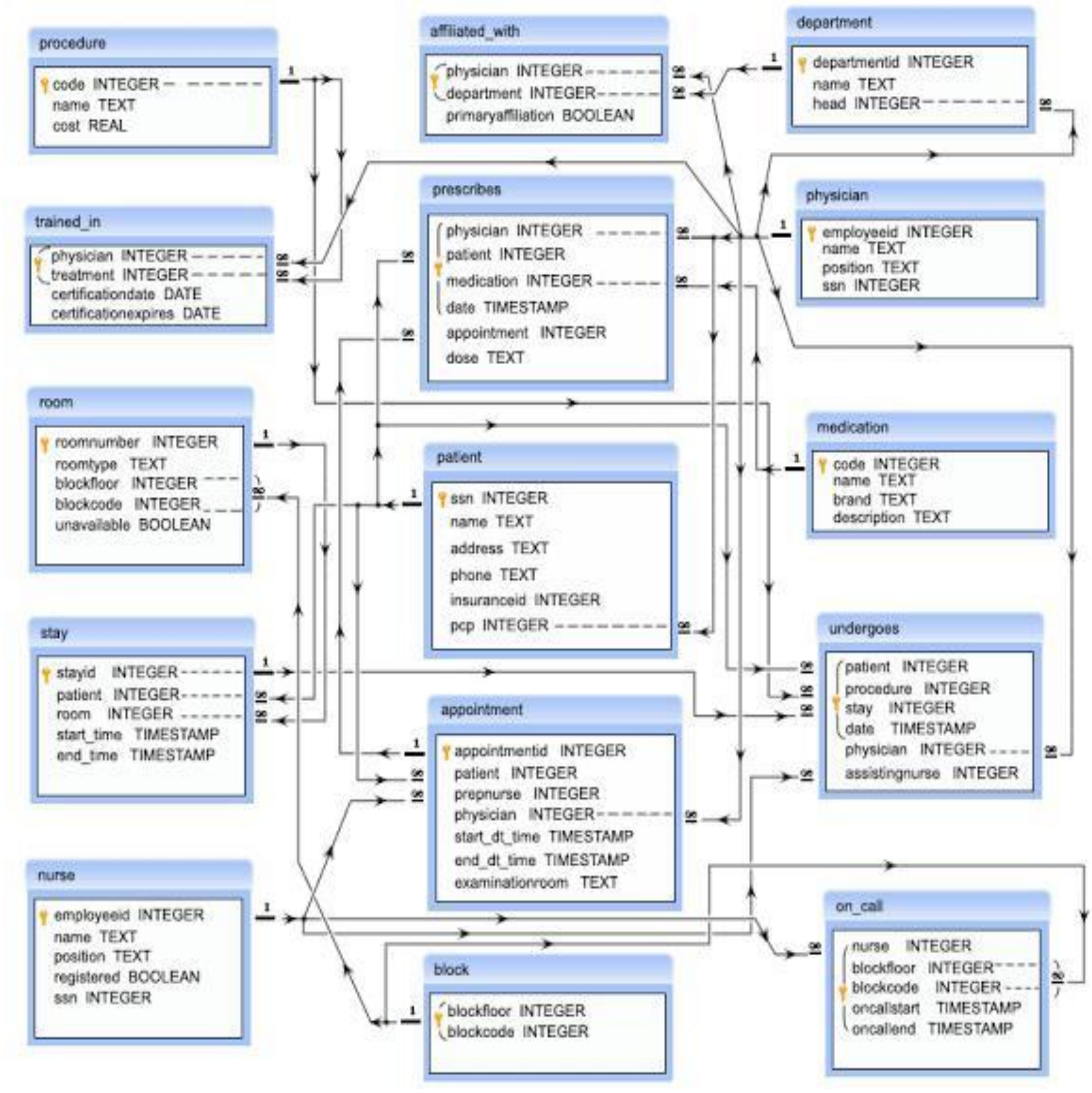


Fig 5.3: Database Design

5.4 UML Diagram:

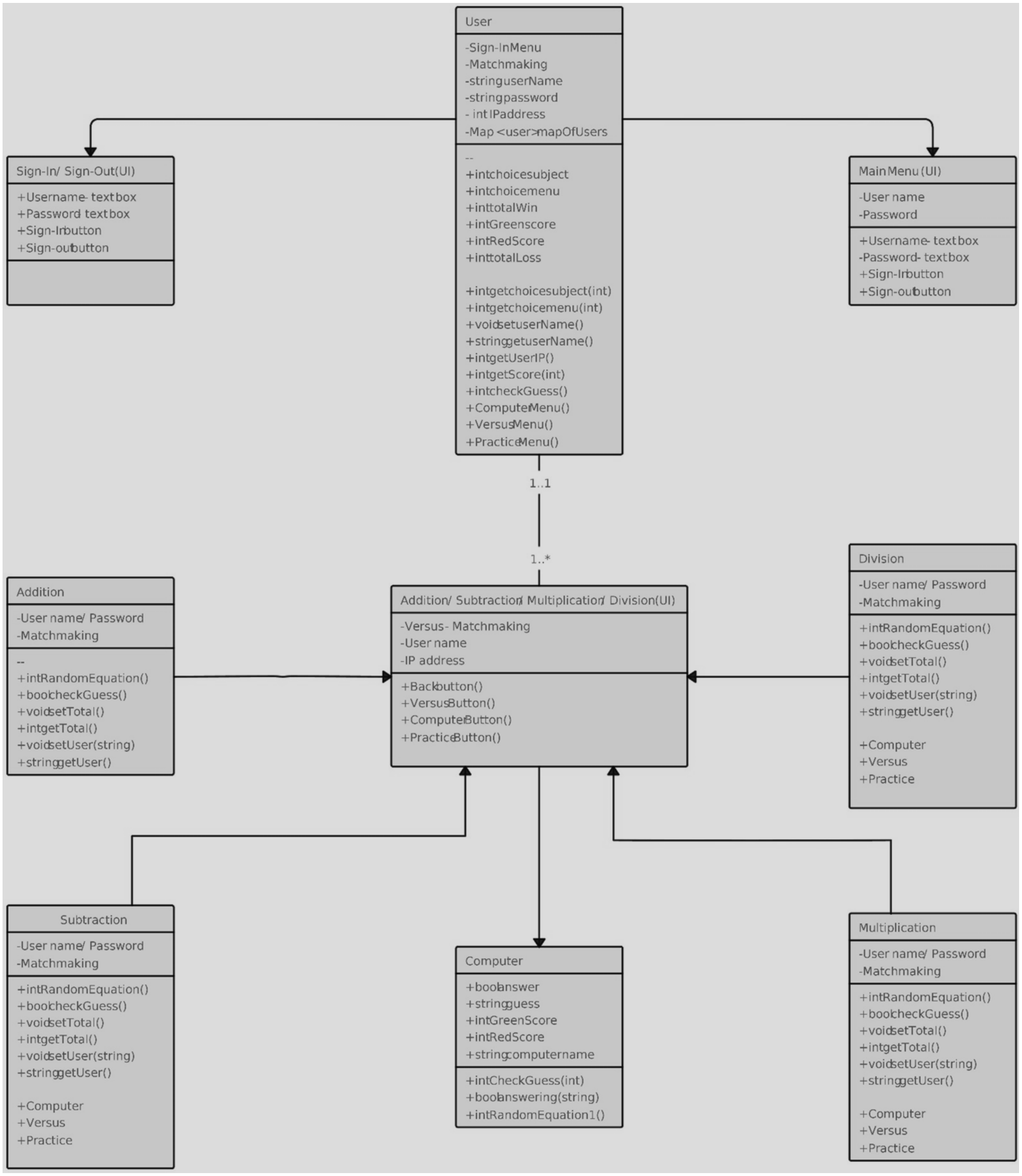


Fig 5.4 : UML Diagram

5.4.1 ER-Diagram:

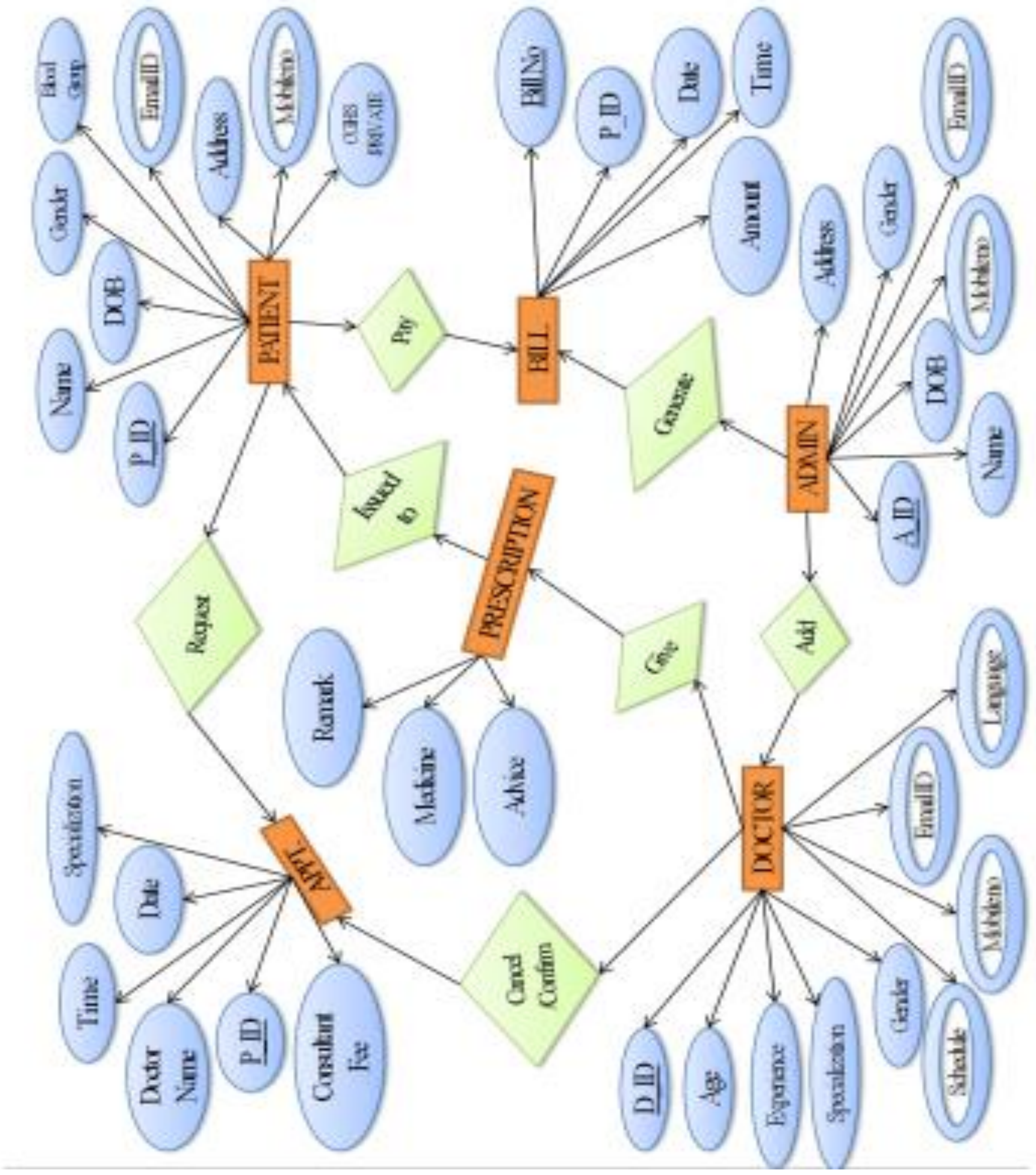


Fig 5.5 ER Diagram

Chapter 6

SYSTEM TESTING

Testing:

<u>BASIS PATH TESTING FOR BOOK APPOINTMENT MODULE</u>

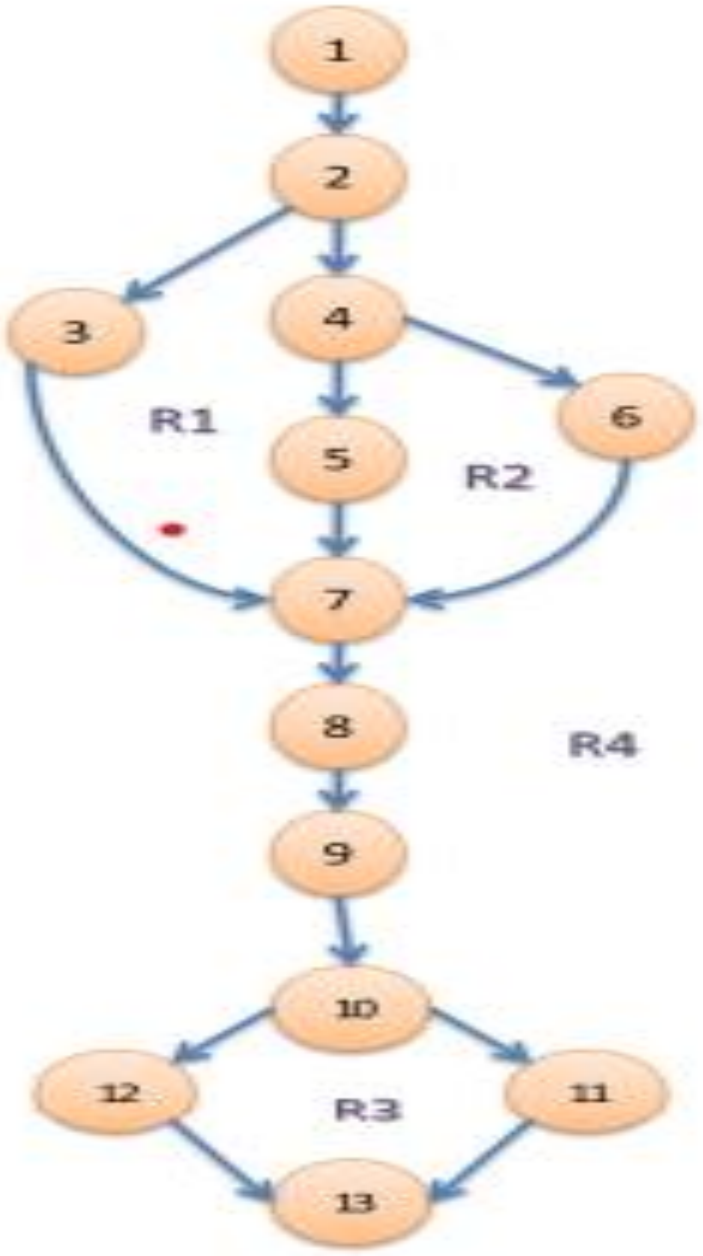
```
enum Status {confirm, cancel};
int Department, Date, Time, mode, ch;
char Dr_Name(50);
cout<< Enter The Information :
cin>> Department;
cin>>Dr_Name;
cin>> Date;
cin>> Time;
bool Appointment = cancel;                       1
cout<<Mode;
cout<<1.Cash;
cout<<2.Debit Card/Credit Card
cout<<3.Net Banking
cout<<Enter mode of payment;
cin>>mode;
```

```
if(mode==1). ———————————— 2
{
Generate a Receipt and send confirmation message;        3
}
else if(mode == 2) ——————— 4
{
Enter Card Details          ┐
Make Payment                │  5
Send confirmation message   ┘
}
else
{
Enter Account Details       ┐  6
 Make Payment               │
Send confirmation message   ┘
}
 //end if ———————————————— 7
 Send appointment Request to the doctor———— 8

Doctor will check the Appointment Requests;
Cout<<Mode;
```

```
Cout<<1.confirm                                              9
Cout<<2.cancel
Cout<<Enter your choice;
Cin>>ch;
if(ch==1) ___________________________________________________ 10
 {
Appointment = Confirm;                 ______________
Send a Confirm Message to the patient. ______________|        11
}
 else
{
 Send a Cancel Message to the patient. ______________         12
}
//end if ____________________________________________         13
```

FLOW GRAPH NOTATION:



2) CYCLOMATIC COMPLEXITY

V(G) 1. Cyclomatic complexity V(G) = Total number of Regions.

V(G) = 4.

2.Cyclomatic complexity V(G) = (E – N) + 2 where E = the number of flow graph edges. i.e. 15

N = the number of flow graph nodes. i.e. 13

V(G) = (15 – 13) + 2 = 4.

3. Cyclomatic complexity

V(G) = P + 1

where P = the number of predicate nodes contained in the flow graph G.

V(G) = 3 + 1 = 4.

There will be 4 independent Paths.

3) INDEPENDENT PATHS

Path A : 1 – 2 – 3 – 7 – 8 – 9 – 10 – 11 – 13

Path B : 1 – 2 – 4 – 5 – 7 – 8 – 9 – 10 – 12 – 13

Path C : 1 – 2 – 4 – 6 – 7 – 8 – 9 – 10 – 11 – 13

Path D : 1 – 2 – 3 – 7 – 8 – 9 – 10 – 12 – 13

CHAPTER 7

SYSTEM IMPLEMENTS

7.1 Name of The modules :

- Registration
- EMR
- Scheduler
- IP and OP Modules
- Vaccination module
- Patient Communication
- Billing
- Inventory
- Insurance
- Pharmacy

7.2 Module Description :

1. Registration

The registration desk at a hospital without an HMS suffers from major setbacks including the lack of digitized documents which has major impacts on the patient experience at these institutions.

Here are a few features of the registration module within an HMS:

- Pre-registration with patient information generates a unique patient identification ID or medical record number that patients use throughout their visit/stay at the hospital.

- Consultant wise token numbers are auto-assigned upon registration and registration bar-code stickers and cards are printed.

- Insurance cards or patient documents are scanned and uploaded at the registration itself.

- Patients are tagged as an emergency, unidentified or medico-legal case, VIP, or based on confidentiality for controlled access to a patient's medical records.

2. EMR

Electronic medical records (EMR) is a centrally stored database that provides hospital staff with access to patient information. They display medical records generated along the patient’s journey across various patient touchpoints in the hospital and are centrally stored in a digitized format that makes it accessible across all departments, thereby reducing the chance of error.

EMRs contain hospital information such as:

- patient’s demographics and insurance details.
- vital signs, past medical history, and allergies of the patient.
- current medications, ongoing care plan for patients in hospital or at home with telemetry devices.
- all radiology images (X Rays/MRI/CT scans)
- laboratory results and hospital notes.
- discharge summaries after hospitalization, which help reduce readmissions to the hospital by providing a summary of care for patients as they leave the hospital.

3. Scheduler

As the name suggests, the scheduler module schedules the tests, surgeries, resources, and calendars of doctors and other staff. It also tracks appointments by their status of 'booked', 'confirmed', 'no shows', 'cancelled', etc.

The interesting waitlist feature allows overbooking appointments so as to manage patient expectations, waiting time, and maximize resource utilization.

4. IP and OP modules

The OP and IP modules of an HMS help hospital staff keep track of patients more accurately and streamline the hospital's processes.

If a patient checks into a hospital for surgery or other procedures, the hospital management system will automatically place them as an inpatient and log them on. The modules allow hospital staff to efficiently manage appointments, surgical operations, and scheduling.

The OP module provides staff members access to the patient’s previous prescriptions and contains templates specifically designed for capturing clinical documentation, SOAP notes, medical history, review of systems, history of patient illness, physical examination, image charts, allergies, vitals, problems, diagnosis, and other progress notes.

The IP module has a ward-activity tracking feature that enables the nurse to convert a physician order to a billable order and review upcoming/completed/missed clinical activities.

5. Vaccination module

The vaccination module in hospital management systems can help a hospital stay up to date with their immunization records. It can also keep track of which vaccines are due soon and auto-schedule those for the right patient.

It can also be set to send automated notifications and alerts to patients about their vaccination dates and dosage.

6. Patient Communication

Effective communication between the patient and the staff is one of the most essential aspects of running a hospital. The patient communication module can send automated promotional and transactional messages and emails to patients regarding new treatment plans, and notifications on upcoming appointments.

A two-way SMS system allows patients to confirm or cancel their appointments without having to visit the hospital or make a call.

7. Billing

The billing module helps hospitals stay up to date with their invoicing and payments. The hospital can also keep track of other hospital-related tasks such as insurance receipts, bills to other departments, and other hospital-related charges.

Here are a few features of the hospital billing module:

- A new bill creation process initiates a new bill for every new patient visit.
- The reconciliation feature synchronizes hospital charges to accounts to streamline the payment process.
- The billing displays view outstanding bills for a department or upcoming bills for a department
- Allows creating one's own custom report based on needs such as bills paid or not paid, outstanding invoices, etc.

8. Inventory

Keeping track of hospital equipment, medical supplies, and stocks is a tremendous task that falls on the shoulders of the stock manager. This is where the inventory module comes in. It keeps track of hospital equipment and hospital supplies and provides an audit trail of who accessed hospital equipment, when it was used, and more.

The inventory module has the following features:

- Tags items with their name, size, color, and categorize them according to the batch number and expiration date.
- Automated first expiry-first out batch determination that makes sure the medication with the earliest expiration date is used first.
- Maintains a list of hazardous material
- Creates unique IDs for each new item to avoid duplications during the process

9. Insurance

This module plays an important role in the core administration, operational and financial processes of a hospital. It improves productivity with lower error rates, reduce claim rejection rates with accurate billing, and improved turnaround rates on claim submissions and resubmission cycles.

Here are a few features are generally seen in insurance modules:

- Manual and online pre-authorization request creation, approval tracking, cancellation, and resubmissions.
- Coder claim review features to ensure tracking and liaising of changes between the medical coder, clinician, and administrative staff.
- Claim history tracking.

10. Pharmacy

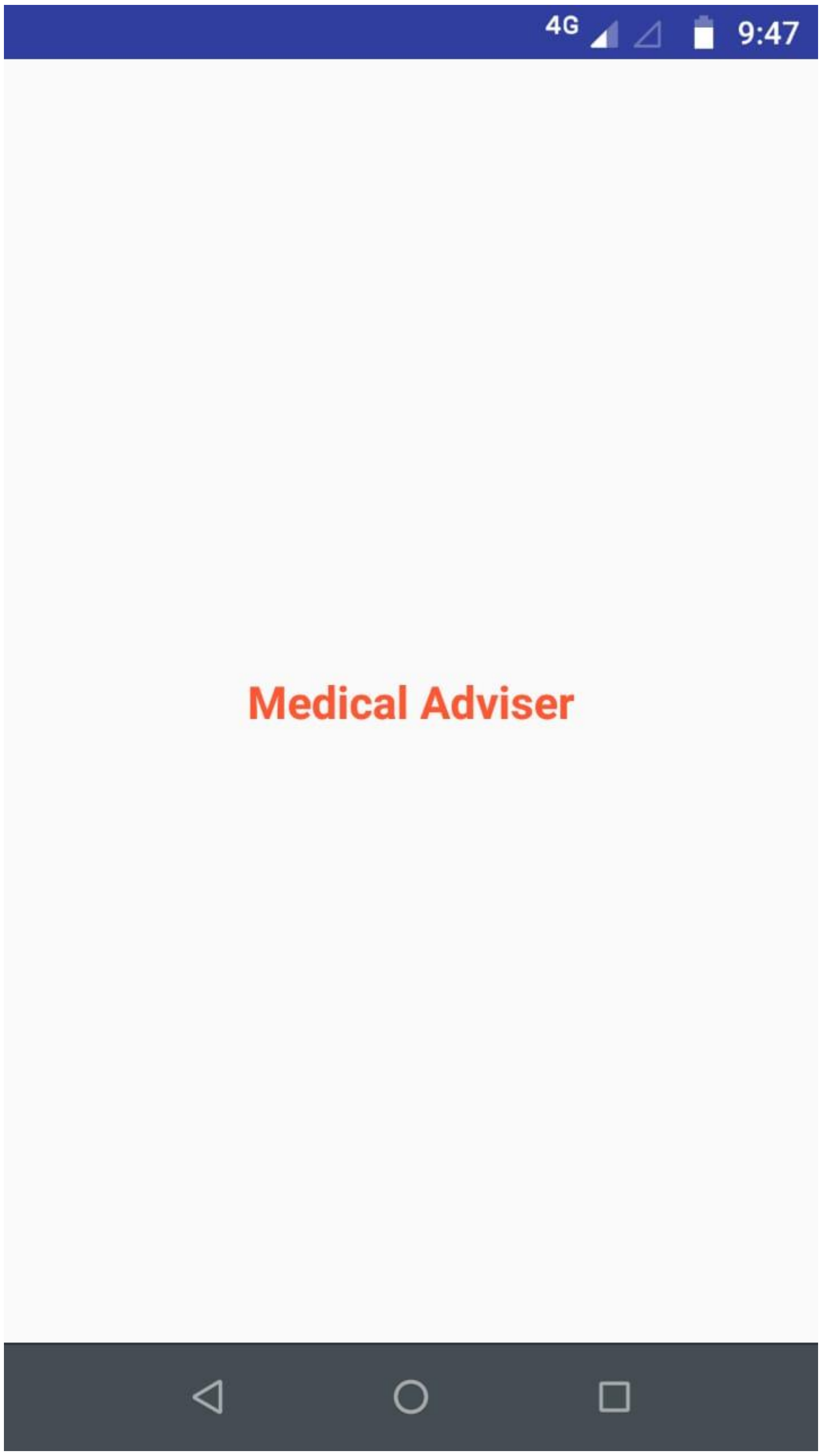
The main objective of a pharmacy module is the efficient management and delivery of pharmaceutical drugs in a hospital or medical center. The module is linked to the rest of the modules within a hospital management system. This allows doctors to prescribe drugs presently in stock and notifies the inventory module to order those drugs out of stock.

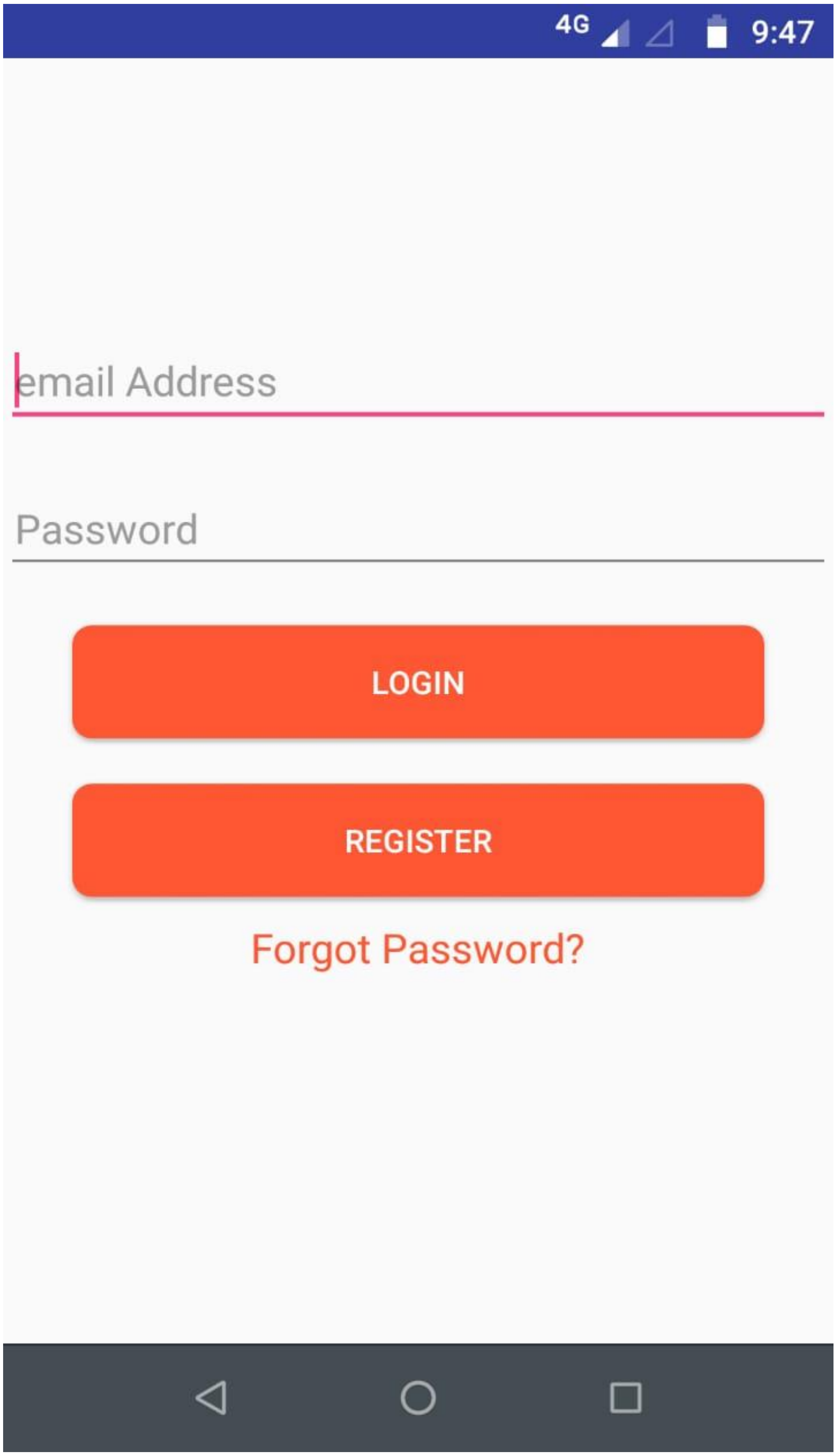
- Pharmacy modules should have features including
- Automated dispensing flow against OP prescriptions to sales by brand names.
- Item and bill-wise discounting and capturing discount authorizer.
- Cash and single or multi-sponsor-based billing.
- Separate pharmacy billing or option to add to the centralized hospital bill.

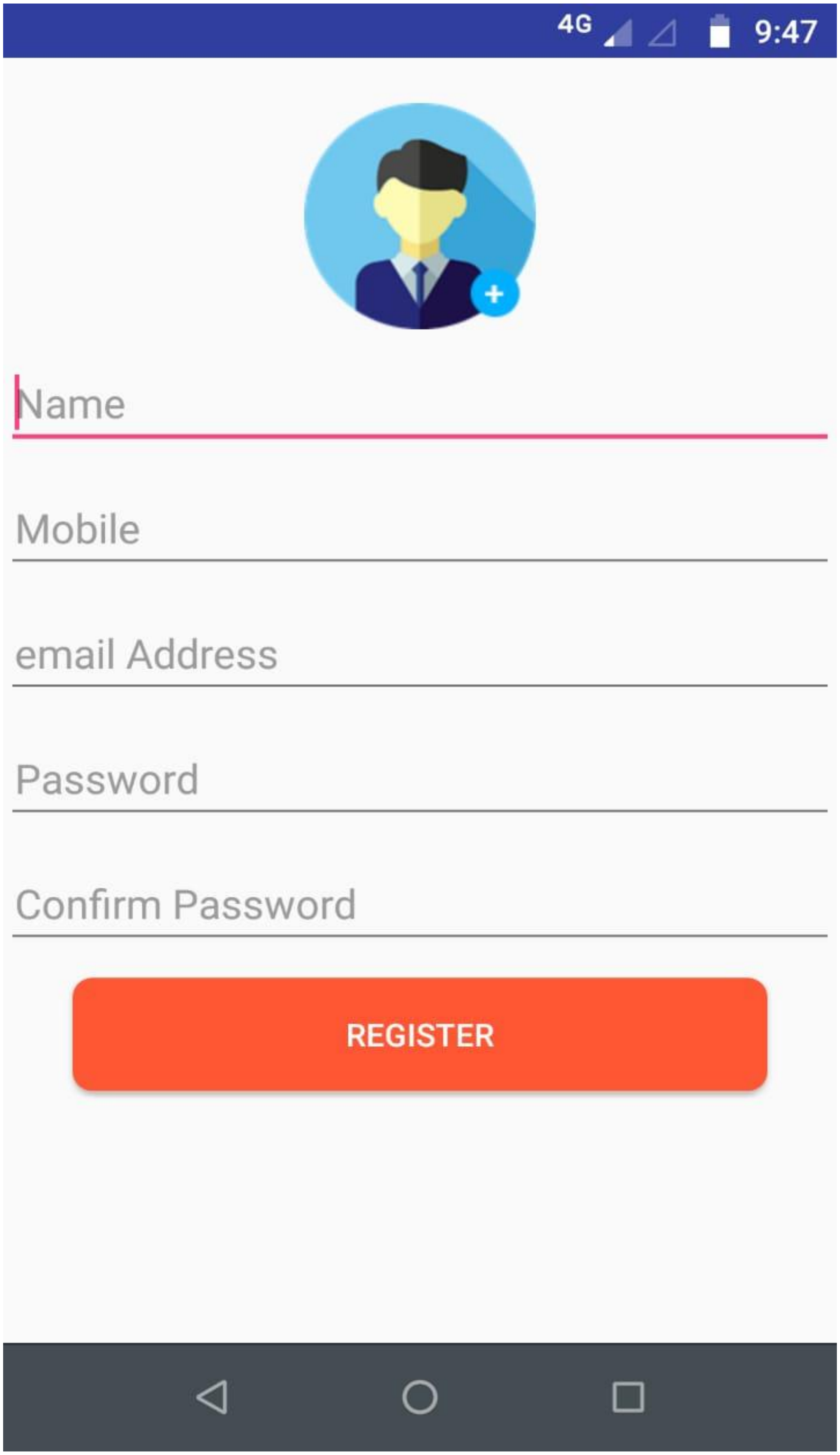
These 10 modules in a hospital management system are so important for the success of medical practices.

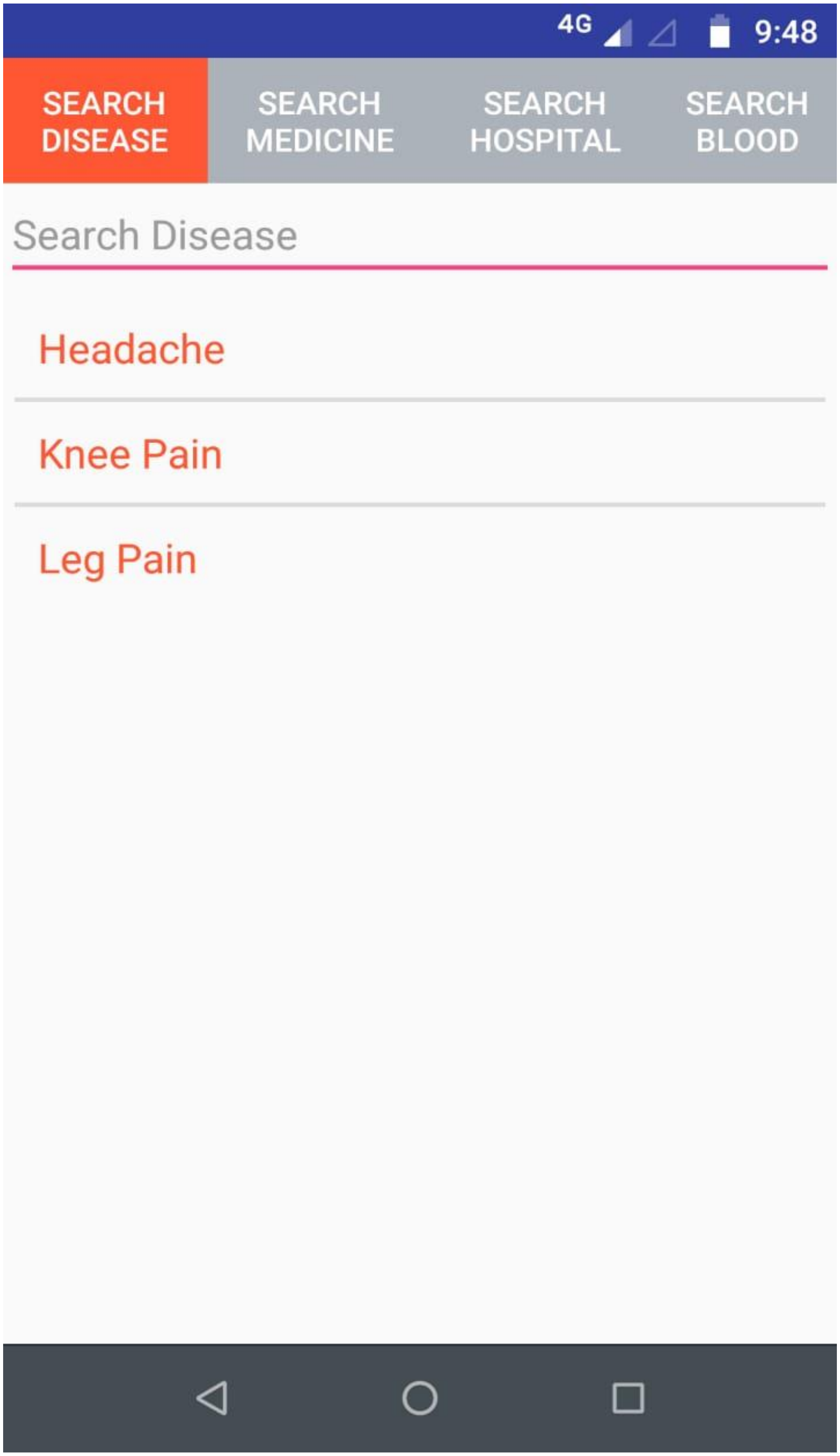
Chapter 8

8.1 Screen Shots :




4G
9:47
email Address
Password
LOGIN
REGISTER
Forgot Password?

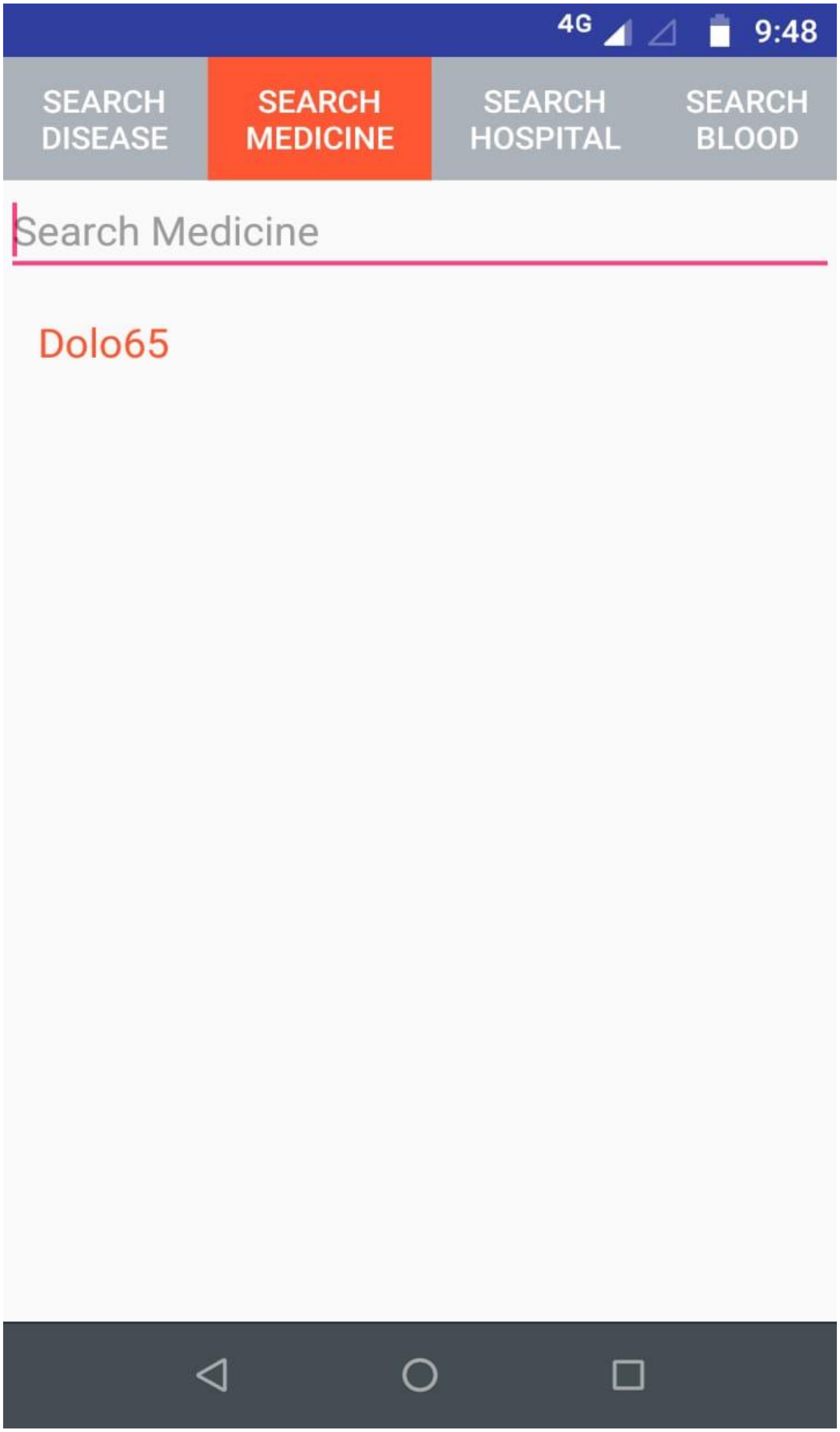

4G
9:47
Name
Mobile
email Address
Password
Confirm Password
REGISTER



Headache

Knee Pain

Leg Pain


4G
9:48
SEARCH DISEASE
SEARCH MEDICINE
SEARCH HOSPITAL
SEARCH BLOOD
Search Medicine
Dolo65

8.2 Source Code :

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.drawerlayout.widget.DrawerLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:openDrawer="start">

    <include
        android:id="@+id/app_bar_main"
        layout="@layout/app_bar_main"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <com.google.android.material.navigation.NavigationView
        android:id="@+id/nav_view"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:fitsSystemWindows="true"
        app:headerLayout="@layout/nav_header_main"
        app:menu="@menu/activity_main_drawer" />
    <include
        android:id="@+id/app_bar_main"
        layout="@layout/app_bar_main"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <com.google.android.material.navigation.NavigationView
        android:id="@+id/nav_view"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:fitsSystemWindows="true"
    app:headerLayout="@layout/nav_header_main"
    app:menu="@menu/activity_main_drawer" />
<include
    android:id="@+id/app_bar_main"
    layout="@layout/app_bar_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />

<com.google.android.material.navigation.NavigationView
    android:id="@+id/nav_view"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:layout_gravity="start"
    android:fitsSystemWindows="true"
    app:headerLayout="@layout/nav_header_main"
    app:menu="@menu/activity_main_drawer" />
<include
    android:id="@+id/app_bar_main"
    layout="@layout/app_bar_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />

<com.google.android.material.navigation.NavigationView
    android:id="@+id/nav_view"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:layout_gravity="start"
    android:fitsSystemWindows="true"
    app:headerLayout="@layout/nav_header_main"
    app:menu="@menu/activity_main_drawer" />
<include
    android:id="@+id/app_bar_main"
    layout="@layout/app_bar_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />

<com.google.android.material.navigation.NavigationView
    android:id="@+id/nav_view"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:layout_gravity="start"
    android:fitsSystemWindows="true"
    app:headerLayout="@layout/nav_header_main"
    app:menu="@menu/activity_main_drawer" />
<include
    android:id="@+id/app_bar_main"
    layout="@layout/app_bar_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />

<com.google.android.material.navigation.NavigationView
    android:id="@+id/nav_view"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:layout_gravity="start"
    android:fitsSystemWindows="true"
    app:headerLayout="@layout/nav_header_main"
    app:menu="@menu/activity_main_drawer" />
<include
    android:id="@+id/app_bar_main"
    layout="@layout/app_bar_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />

<com.google.android.material.navigation.NavigationView
    android:id="@+id/nav_view"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:layout_gravity="start"
    android:fitsSystemWindows="true"
    app:headerLayout="@layout/nav_header_main"
    app:menu="@menu/activity_main_drawer" />
<include
    android:id="@+id/app_bar_main"
    layout="@layout/app_bar_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />

<com.google.android.material.navigation.NavigationView
    android:id="@+id/nav_view"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:layout_gravity="start"
    android:fitsSystemWindows="true"
    app:headerLayout="@layout/nav_header_main"
    app:menu="@menu/activity_main_drawer" />
```

```
<include
   android:id="@+id/app_bar_main"
   layout="@layout/app_bar_main"
   android:layout_width="match_parent"
   android:layout_height="match_parent" />

<com.google.android.material.navigation.NavigationView
   android:id="@+id/nav_view"
   android:layout_width="wrap_content"
   android:layout_height="match_parent"
   android:layout_gravity="start"
   android:fitsSystemWindows="true"
   app:headerLayout="@layout/nav_header_main"
   app:menu="@menu/activity_main_drawer" />
```

```
<include
   android:id="@+id/app_bar_main"
   layout="@layout/app_bar_main"
   android:layout_width="match_parent"
   android:layout_height="match_parent" />

<com.google.android.material.navigation.NavigationView
   android:id="@+id/nav_view"
   android:layout_width="wrap_content"
   android:layout_height="match_parent"
   android:layout_gravity="start"
   android:fitsSystemWindows="true"
   app:headerLayout="@layout/nav_header_main"
   app:menu="@menu/activity_main_drawer" />
<include
   android:id="@+id/app_bar_main"
   layout="@layout/app_bar_main"
   android:layout_width="match_parent"
   android:layout_height="match_parent" />

<com.google.android.material.navigation.NavigationView
   android:id="@+id/nav_view"
   android:layout_width="wrap_content"
   android:layout_height="match_parent"
   android:layout_gravity="start"
   android:fitsSystemWindows="true"
   app:headerLayout="@layout/nav_header_main"
   app:menu="@menu/activity_main_drawer" />
```

```
</androidx.drawerlayout.widget.DrawerLayout>
```

app_bar.xml :

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <com.google.android.material.appbar.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/Theme.MyApplication.AppBarOverlay">

        <androidx.appcompat.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            app:popupTheme="@style/Theme.MyApplication.PopupOverlay" />

    </com.google.android.material.appbar.AppBarLayout>

    <include layout="@layout/content_main" />

    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:id="@+id/fab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|end"
        android:layout_marginEnd="@dimen/fab_margin"
        android:layout_marginBottom="16dp"
        app:srcCompat="@android:drawable/ic_dialog_email" />
```

```
</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

MainAvctivity.java :

```
package com.example.myapplication;

import android.os.Bundle;
import android.view.View;
import android.view.Menu;

import com.google.android.material.snackbar.Snackbar;
import com.google.android.material.navigation.NavigationView;

import androidx.navigation.NavController;
import androidx.navigation.Navigation;
import androidx.navigation.ui.AppBarConfiguration;
import androidx.navigation.ui.NavigationUI;
import androidx.drawerlayout.widget.DrawerLayout;
import androidx.appcompat.app.AppCompatActivity;

import com.example.myapplication.databinding.ActivityMainBinding;

public class MainActivity extends AppCompatActivity {

    private AppBarConfiguration mAppBarConfiguration;
    private ActivityMainBinding binding;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        setSupportActionBar(binding.appBarMain.toolbar);
        binding.appBarMain.fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, "Replace with your own action",
Snackbar.LENGTH_LONG)
                        .setAction("Action", null).show();
            }
        });
        DrawerLayout drawer = binding.drawerLayout;
        NavigationView navigationView = binding.navView;
        // Passing each menu ID as a set of Ids because each
        // menu should be considered as top level destinations.
        mAppBarConfiguration = new AppBarConfiguration.Builder(
                R.id.nav_home, R.id.nav_gallery, R.id.nav_slideshow)
                .setOpenableLayout(drawer)
                .build();
        NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment_content_main);
        NavigationUI.setupActionBarWithNavController(this, navController,
mAppBarConfiguration);
        NavigationUI.setupWithNavController(navigationView, navController);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public boolean onSupportNavigateUp() {
        NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment_content_main);
        return NavigationUI.navigateUp(navController, mAppBarConfiguration)
                || super.onSupportNavigateUp();
    }
}


public class MainActivity extends AppCompatActivity {

    private AppBarConfiguration mAppBarConfiguration;
    private ActivityMainBinding binding;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        setSupportActionBar(binding.appBarMain.toolbar);
        binding.appBarMain.fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, "Replace with your own action",
Snackbar.LENGTH_LONG)
                        .setAction("Action", null).show();
            }
        });
        DrawerLayout drawer = binding.drawerLayout;
        NavigationView navigationView = binding.navView;
        // Passing each menu ID as a set of Ids because each
        // menu should be considered as top level destinations.
        mAppBarConfiguration = new AppBarConfiguration.Builder(
                R.id.nav_home, R.id.nav_gallery, R.id.nav_slideshow)
                .setOpenableLayout(drawer)
                .build();
        NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment_content_main);
        NavigationUI.setupActionBarWithNavController(this, navController,
mAppBarConfiguration);
        NavigationUI.setupWithNavController(navigationView, navController);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public boolean onSupportNavigateUp() {
        NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment_content_main);
        return NavigationUI.navigateUp(navController, mAppBarConfiguration)
                || super.onSupportNavigateUp();
    }
}
```

```
public class MainActivity extends AppCompatActivity {

    private AppBarConfiguration mAppBarConfiguration;
    private ActivityMainBinding binding;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        setSupportActionBar(binding.appBarMain.toolbar);
        binding.appBarMain.fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, "Replace with your own action",
Snackbar.LENGTH_LONG)
                        .setAction("Action", null).show();
            }
        });
        DrawerLayout drawer = binding.drawerLayout;
        NavigationView navigationView = binding.navView;
        // Passing each menu ID as a set of Ids because each
        // menu should be considered as top level destinations.
        mAppBarConfiguration = new AppBarConfiguration.Builder(
                R.id.nav_home, R.id.nav_gallery, R.id.nav_slideshow)
                .setOpenableLayout(drawer)
                .build();
        NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment_content_main);
        NavigationUI.setupActionBarWithNavController(this, navController,
mAppBarConfiguration);
        NavigationUI.setupWithNavController(navigationView, navController);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
    @Override
    public boolean onSupportNavigateUp() {
        NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment_content_main);
        return NavigationUI.navigateUp(navController, mAppBarConfiguration)
                || super.onSupportNavigateUp();
    }
}

public class MainActivity extends AppCompatActivity {

    private AppBarConfiguration mAppBarConfiguration;
    private ActivityMainBinding binding;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        setSupportActionBar(binding.appBarMain.toolbar);
        binding.appBarMain.fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, "Replace with your own action",
Snackbar.LENGTH_LONG)
                        .setAction("Action", null).show();
            }
        });
        DrawerLayout drawer = binding.drawerLayout;
        NavigationView navigationView = binding.navView;
        // Passing each menu ID as a set of Ids because each
        // menu should be considered as top level destinations.
        mAppBarConfiguration = new AppBarConfiguration.Builder(
                R.id.nav_home, R.id.nav_gallery, R.id.nav_slideshow)
                .setOpenableLayout(drawer)
                .build();
        NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment_content_main);
        NavigationUI.setupActionBarWithNavController(this, navController,
mAppBarConfiguration);
        NavigationUI.setupWithNavController(navigationView, navController);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public boolean onSupportNavigateUp() {
        NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment_content_main);
        return NavigationUI.navigateUp(navController, mAppBarConfiguration)
                || super.onSupportNavigateUp();
    }
}


public class MainActivity extends AppCompatActivity {

    private AppBarConfiguration mAppBarConfiguration;
    private ActivityMainBinding binding;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        setSupportActionBar(binding.appBarMain.toolbar);
        binding.appBarMain.fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, "Replace with your own action",
Snackbar.LENGTH_LONG)
                        .setAction("Action", null).show();
            }
        });
        DrawerLayout drawer = binding.drawerLayout;
        NavigationView navigationView = binding.navView;
        // Passing each menu ID as a set of Ids because each
        // menu should be considered as top level destinations.
        mAppBarConfiguration = new AppBarConfiguration.Builder(
                R.id.nav_home, R.id.nav_gallery, R.id.nav_slideshow)
                .setOpenableLayout(drawer)
                .build();
        NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment_content_main);
        NavigationUI.setupActionBarWithNavController(this, navController,
mAppBarConfiguration);
        NavigationUI.setupWithNavController(navigationView, navController);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public boolean onSupportNavigateUp() {
        NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment_content_main);
        return NavigationUI.navigateUp(navController, mAppBarConfiguration)
                || super.onSupportNavigateUp();
    }
}
public class MainActivity extends AppCompatActivity {

    private AppBarConfiguration mAppBarConfiguration;
    private ActivityMainBinding binding;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        setSupportActionBar(binding.appBarMain.toolbar);
        binding.appBarMain.fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, "Replace with your own action",
Snackbar.LENGTH_LONG)
                        .setAction("Action", null).show();
            }
        });
        DrawerLayout drawer = binding.drawerLayout;
        NavigationView navigationView = binding.navView;
        // Passing each menu ID as a set of Ids because each
        // menu should be considered as top level destinations.
        mAppBarConfiguration = new AppBarConfiguration.Builder(
                R.id.nav_home, R.id.nav_gallery, R.id.nav_slideshow)
                .setOpenableLayout(drawer)
                .build();
        NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment_content_main);
        NavigationUI.setupActionBarWithNavController(this, navController,
mAppBarConfiguration);
        NavigationUI.setupWithNavController(navigationView, navController);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public boolean onSupportNavigateUp() {
        NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment_content_main);
        return NavigationUI.navigateUp(navController, mAppBarConfiguration)
                || super.onSupportNavigateUp();
    }
}
public class MainActivity extends AppCompatActivity {

    private AppBarConfiguration mAppBarConfiguration;
    private ActivityMainBinding binding;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        setSupportActionBar(binding.appBarMain.toolbar);
        binding.appBarMain.fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, "Replace with your own action",
Snackbar.LENGTH_LONG)
                        .setAction("Action", null).show();
            }
        });
        DrawerLayout drawer = binding.drawerLayout;
        NavigationView navigationView = binding.navView;
        // Passing each menu ID as a set of Ids because each
        // menu should be considered as top level destinations.
        mAppBarConfiguration = new AppBarConfiguration.Builder(
                R.id.nav_home, R.id.nav_gallery, R.id.nav_slideshow)
                .setOpenableLayout(drawer)
                .build();
        NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment_content_main);
        NavigationUI.setupActionBarWithNavController(this, navController,
mAppBarConfiguration);
        NavigationUI.setupWithNavController(navigationView, navController);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public boolean onSupportNavigateUp() {
        NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment_content_main);
        return NavigationUI.navigateUp(navController, mAppBarConfiguration)
                || super.onSupportNavigateUp();
    }
}
```

```
public class MainActivity extends AppCompatActivity {

    private AppBarConfiguration mAppBarConfiguration;
    private ActivityMainBinding binding;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        setSupportActionBar(binding.appBarMain.toolbar);
        binding.appBarMain.fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, "Replace with your own action",
Snackbar.LENGTH_LONG)
                        .setAction("Action", null).show();
            }
        });
        DrawerLayout drawer = binding.drawerLayout;
        NavigationView navigationView = binding.navView;
        // Passing each menu ID as a set of Ids because each
        // menu should be considered as top level destinations.
        mAppBarConfiguration = new AppBarConfiguration.Builder(
                R.id.nav_home, R.id.nav_gallery, R.id.nav_slideshow)
                .setOpenableLayout(drawer)
                .build();
        NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment_content_main);
        NavigationUI.setupActionBarWithNavController(this, navController,
mAppBarConfiguration);
        NavigationUI.setupWithNavController(navigationView, navController);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
    @Override
    public boolean onSupportNavigateUp() {
        NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment_content_main);
        return NavigationUI.navigateUp(navController, mAppBarConfiguration)
                || super.onSupportNavigateUp();
    }
}
public class MainActivity extends AppCompatActivity {

    private AppBarConfiguration mAppBarConfiguration;
    private ActivityMainBinding binding;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        setSupportActionBar(binding.appBarMain.toolbar);
        binding.appBarMain.fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, "Replace with your own action",
Snackbar.LENGTH_LONG)
                        .setAction("Action", null).show();
            }
        });
        DrawerLayout drawer = binding.drawerLayout;
        NavigationView navigationView = binding.navView;
        // Passing each menu ID as a set of Ids because each
        // menu should be considered as top level destinations.
        mAppBarConfiguration = new AppBarConfiguration.Builder(
                R.id.nav_home, R.id.nav_gallery, R.id.nav_slideshow)
                .setOpenableLayout(drawer)
                .build();
        NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment_content_main);
        NavigationUI.setupActionBarWithNavController(this, navController,
mAppBarConfiguration);
        NavigationUI.setupWithNavController(navigationView, navController);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public boolean onSupportNavigateUp() {
        NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment_content_main);
        return NavigationUI.navigateUp(navController, mAppBarConfiguration)
                || super.onSupportNavigateUp();
    }
}
public class MainActivity extends AppCompatActivity {

    private AppBarConfiguration mAppBarConfiguration;
    private ActivityMainBinding binding;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        setSupportActionBar(binding.appBarMain.toolbar);
        binding.appBarMain.fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, "Replace with your own action",
Snackbar.LENGTH_LONG)
                        .setAction("Action", null).show();
            }
        });
        DrawerLayout drawer = binding.drawerLayout;
        NavigationView navigationView = binding.navView;
        // Passing each menu ID as a set of Ids because each
        // menu should be considered as top level destinations.
        mAppBarConfiguration = new AppBarConfiguration.Builder(
                R.id.nav_home, R.id.nav_gallery, R.id.nav_slideshow)
                .setOpenableLayout(drawer)
                .build();
        NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment_content_main);
        NavigationUI.setupActionBarWithNavController(this, navController,
mAppBarConfiguration);
        NavigationUI.setupWithNavController(navigationView, navController);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public boolean onSupportNavigateUp() {
        NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment_content_main);
        return NavigationUI.navigateUp(navController, mAppBarConfiguration)
                || super.onSupportNavigateUp();
    }
}
public class MainActivity extends AppCompatActivity {

    private AppBarConfiguration mAppBarConfiguration;
    private ActivityMainBinding binding;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        setSupportActionBar(binding.appBarMain.toolbar);
        binding.appBarMain.fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, "Replace with your own action",
Snackbar.LENGTH_LONG)
                        .setAction("Action", null).show();
            }
        });
        DrawerLayout drawer = binding.drawerLayout;
        NavigationView navigationView = binding.navView;
        // Passing each menu ID as a set of Ids because each
        // menu should be considered as top level destinations.
        mAppBarConfiguration = new AppBarConfiguration.Builder(
                R.id.nav_home, R.id.nav_gallery, R.id.nav_slideshow)
                .setOpenableLayout(drawer)
                .build();
        NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment_content_main);
        NavigationUI.setupActionBarWithNavController(this, navController,
mAppBarConfiguration);
        NavigationUI.setupWithNavController(navigationView, navController);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public boolean onSupportNavigateUp() {
        NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment_content_main);
        return NavigationUI.navigateUp(navController, mAppBarConfiguration)
                || super.onSupportNavigateUp();
    }
}
public class MainActivity extends AppCompatActivity {

    private AppBarConfiguration mAppBarConfiguration;
    private ActivityMainBinding binding;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());
        setSupportActionBar(binding.appBarMain.toolbar);
        binding.appBarMain.fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, "Replace with your own action",
Snackbar.LENGTH_LONG)
                        .setAction("Action", null).show();
            }
        });
        DrawerLayout drawer = binding.drawerLayout;
        NavigationView navigationView = binding.navView;
        // Passing each menu ID as a set of Ids because each
        // menu should be considered as top level destinations.
        mAppBarConfiguration = new AppBarConfiguration.Builder(
                R.id.nav_home, R.id.nav_gallery, R.id.nav_slideshow)
                .setOpenableLayout(drawer)
                .build();
        NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment_content_main);
        NavigationUI.setupActionBarWithNavController(this, navController,
mAppBarConfiguration);
        NavigationUI.setupWithNavController(navigationView, navController);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public boolean onSupportNavigateUp() {
        NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment_content_main);
        return NavigationUI.navigateUp(navController, mAppBarConfiguration)
                || super.onSupportNavigateUp();
    }
}
public class MainActivity extends AppCompatActivity {

    private AppBarConfiguration mAppBarConfiguration;
    private ActivityMainBinding binding;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        setSupportActionBar(binding.appBarMain.toolbar);
        binding.appBarMain.fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, "Replace with your own action",
Snackbar.LENGTH_LONG)
                        .setAction("Action", null).show();
            }
        });
        DrawerLayout drawer = binding.drawerLayout;
        NavigationView navigationView = binding.navView;
        // Passing each menu ID as a set of Ids because each
        // menu should be considered as top level destinations.
        mAppBarConfiguration = new AppBarConfiguration.Builder(
                R.id.nav_home, R.id.nav_gallery, R.id.nav_slideshow)
                .setOpenableLayout(drawer)
                .build();
        NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment_content_main);
        NavigationUI.setupActionBarWithNavController(this, navController,
mAppBarConfiguration);
        NavigationUI.setupWithNavController(navigationView, navController);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public boolean onSupportNavigateUp() {
        NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment_content_main);
        return NavigationUI.navigateUp(navController, mAppBarConfiguration)
                || super.onSupportNavigateUp();
    }
}
public class MainActivity extends AppCompatActivity {

    private AppBarConfiguration mAppBarConfiguration;
    private ActivityMainBinding binding;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        setSupportActionBar(binding.appBarMain.toolbar);
        binding.appBarMain.fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, "Replace with your own action",
Snackbar.LENGTH_LONG)
                        .setAction("Action", null).show();
            }
        });
        DrawerLayout drawer = binding.drawerLayout;
        NavigationView navigationView = binding.navView;
        // Passing each menu ID as a set of Ids because each
        // menu should be considered as top level destinations.
        mAppBarConfiguration = new AppBarConfiguration.Builder(
                R.id.nav_home, R.id.nav_gallery, R.id.nav_slideshow)
                .setOpenableLayout(drawer)
                .build();
        NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment_content_main);
        NavigationUI.setupActionBarWithNavController(this, navController,
mAppBarConfiguration);
        NavigationUI.setupWithNavController(navigationView, navController);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public boolean onSupportNavigateUp() {
        NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment_content_main);
        return NavigationUI.navigateUp(navController, mAppBarConfiguration)
                || super.onSupportNavigateUp();
    }
}
public class MainActivity extends AppCompatActivity {

    private AppBarConfiguration mAppBarConfiguration;
    private ActivityMainBinding binding;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        setSupportActionBar(binding.appBarMain.toolbar);
        binding.appBarMain.fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, "Replace with your own action",
Snackbar.LENGTH_LONG)
                        .setAction("Action", null).show();
            }
        });
        DrawerLayout drawer = binding.drawerLayout;
        NavigationView navigationView = binding.navView;
        // Passing each menu ID as a set of Ids because each
        // menu should be considered as top level destinations.
        mAppBarConfiguration = new AppBarConfiguration.Builder(
                R.id.nav_home, R.id.nav_gallery, R.id.nav_slideshow)
                .setOpenableLayout(drawer)
                .build();
        NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment_content_main);
        NavigationUI.setupActionBarWithNavController(this, navController,
mAppBarConfiguration);
        NavigationUI.setupWithNavController(navigationView, navController);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public boolean onSupportNavigateUp() {
        NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment_content_main);
        return NavigationUI.navigateUp(navController, mAppBarConfiguration)
                || super.onSupportNavigateUp();
    }
}
public class MainActivity extends AppCompatActivity {

    private AppBarConfiguration mAppBarConfiguration;
    private ActivityMainBinding binding;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        setSupportActionBar(binding.appBarMain.toolbar);
        binding.appBarMain.fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, "Replace with your own action",
Snackbar.LENGTH_LONG)
                        .setAction("Action", null).show();
            }
        });
        DrawerLayout drawer = binding.drawerLayout;
        NavigationView navigationView = binding.navView;
        // Passing each menu ID as a set of Ids because each
        // menu should be considered as top level destinations.
        mAppBarConfiguration = new AppBarConfiguration.Builder(
                R.id.nav_home, R.id.nav_gallery, R.id.nav_slideshow)
                .setOpenableLayout(drawer)
                .build();
        NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment_content_main);
        NavigationUI.setupActionBarWithNavController(this, navController,
mAppBarConfiguration);
        NavigationUI.setupWithNavController(navigationView, navController);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public boolean onSupportNavigateUp() {
        NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment_content_main);
        return NavigationUI.navigateUp(navController, mAppBarConfiguration)
                || super.onSupportNavigateUp();
    }
}
public class MainActivity extends AppCompatActivity {

    private AppBarConfiguration mAppBarConfiguration;
    private ActivityMainBinding binding;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        setSupportActionBar(binding.appBarMain.toolbar);
        binding.appBarMain.fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, "Replace with your own action",
Snackbar.LENGTH_LONG)
                        .setAction("Action", null).show();
            }
        });
        DrawerLayout drawer = binding.drawerLayout;
        NavigationView navigationView = binding.navView;
        // Passing each menu ID as a set of Ids because each
        // menu should be considered as top level destinations.
        mAppBarConfiguration = new AppBarConfiguration.Builder(
                R.id.nav_home, R.id.nav_gallery, R.id.nav_slideshow)
                .setOpenableLayout(drawer)
                .build();
        NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment_content_main);
        NavigationUI.setupActionBarWithNavController(this, navController,
mAppBarConfiguration);
        NavigationUI.setupWithNavController(navigationView, navController);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public boolean onSupportNavigateUp() {
        NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment_content_main);
        return NavigationUI.navigateUp(navController, mAppBarConfiguration)
                || super.onSupportNavigateUp();
    }
}
public class MainActivity extends AppCompatActivity {

    private AppBarConfiguration mAppBarConfiguration;
    private ActivityMainBinding binding;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        setSupportActionBar(binding.appBarMain.toolbar);
        binding.appBarMain.fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, "Replace with your own action",
Snackbar.LENGTH_LONG)
                        .setAction("Action", null).show();
            }
        });
        DrawerLayout drawer = binding.drawerLayout;
        NavigationView navigationView = binding.navView;
        // Passing each menu ID as a set of Ids because each
        // menu should be considered as top level destinations.
        mAppBarConfiguration = new AppBarConfiguration.Builder(
                R.id.nav_home, R.id.nav_gallery, R.id.nav_slideshow)
                .setOpenableLayout(drawer)
                .build();
        NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment_content_main);
        NavigationUI.setupActionBarWithNavController(this, navController,
mAppBarConfiguration);
        NavigationUI.setupWithNavController(navigationView, navController);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
    @Override
    public boolean onSupportNavigateUp() {
        NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment_content_main);
        return NavigationUI.navigateUp(navController, mAppBarConfiguration)
                || super.onSupportNavigateUp();
    }
}
```

Chapter 9

Conclusion and Future Enhancement

9.1 Conclusion :

This application is a utility for hospital executives and helps them to review, monitor and analyze the hospital performance metrics on a daily basis such as daily census, average length of patient stay in the hospital, number of patients admitted and discharged for a given day, patient insurance details like insurance company name and number of patients using the corresponding insurance, diagnosis details like the least and the most common diagnosed diseases and patient demographic distribution like percent of patients coming from each county, without using paper based generated reports and desktop reports. This system can improve the efforts of hospital executives by providing hospital operational data in a portable manner with ease of access and helps them in making effective decisions required for improving the quality of patient care services.

9.2 FUTURE ENHANCEMENT:

This application can be enhanced using Interface Engines which store decoded HL7 messages. The Interface Engines can be mapped and synchronized with the database. Soon the facility of communicating by message will be established. This application is early available in other Indian regional languages. The application can be further extended to display new metrics like surgery schedule, staffing requirement, etc. on the Android User Interface.

Chapter 10

REFERENCES

Online Reference :

https://www.youtube.com/playlist?list=PL4uWLKFsZfcGBja19mrwodNm6AyzZk2B
https://github.com/sbmvirdi/Hospital-Management-System-Android-App#redme
https://youtu.be/U1iwe3L5pzc
https://youtu.be/B-WRwKVUKPk
https://github.com/sreenivasankv/hospital_android Developer Tools,
http://developer.android.com/tools/index.html, Get the Android SDk,
http://developer.android.com/sdk/index.html, Resources Overview,
http://developer.android.com/guide/topics/resources/overview.html, Activities,
http://developer.android.com/guide/components/activities.html, Intents and Intent Filters,
http://developer.android.com/guide/components/intents-filters.html, Media Playback
http://developer.android.com/guide/topics/media/mediaplayer.html, Activity Codes,
http://developer.android.com/guide/topics/manifest/activity-element.html,

Book Reference :

Reto Meier, Professional Android 2 Application Development, 2010 by Wiley Publishing, Inc., Indianapolis, Indiana ISBN: 978-0-470-56552-0 Wei Meng Lee,

Beginning Android™ 4 Application Development, 2012 by John Wiley & Sons, Inc., Indianapolis, Indiana ISBN: 978-1-118-19954-1