## HOSPITALITY MOBILE HEALTH-CARE USING CLOUD

***Project submitted in partial fulfillment of the requirements for the award of the degree***

## BACHELOR OF TECHNOLOGY

IN

## COMPUTER SCIENCE AND ENGINEERING

### BY

**P BHANUTRINADH (18C21A0562)**

### S SAI RAM (18C21A0564)

**D RAJU (18C21A0570)**

### Under the Esteemed guidance of

**Mr. RAVINDHER M.Tech**

Assistant Professor



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

## HOLY MARY INSTITUTE OF TECHNOLOGY & SCIENCE

**(COLLEGE OF ENGINEERING)**

***(Approved by AICTE New Delhi, Permanently Affiliated to JNTU Hyderabad, Accredited by NAAC with ‘A’ Grade)***

**Bogaram (V), Keesara (M), Medchal District -501 301.**

2021-2022

## HOLY MARY INSTITUTE OF TECHNOLOGY & SCIENCE

#### (COLLEGE OF ENGINEERING)

***(Approved by AICTE New Delhi, Permanently Affiliated to JNTU Hyderabad, Accredited by NAAC with ‘A’ Grade)***

**Bogaram (V), Keesara (M), Medchal Dist-501301.**

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



# CERTIFICATE

This is to certify that the mini project entitled “HOSPITAL MOBILE HEALTH-CARE USING CLOUD” is being submitted by P.BHANUTRINADH (18C21A0562), S.SAIRAM(18C21A0564),

D.RAJU(18C21A0570), in Partial fulfillement of the academia requirements for the award of the degree of Bachelor of Technology in “COMPUTER SCIENCE AND ENGINEERING” HOLY MARY INSTITUTE OF TECHNOLOGY & SCIENCE, JNTU Hyderabad during the year 2021-2022.

### INTERNAL GUIDE HEAD OF THE DEPARTMENT

Mr. **RAVINDHER** M.Tech DR .B.NARSIMHA M.Tech, Ph.D.

Assistant Professor Professor & HoD

Dept. of Computer Science & Engineering. Dept. of Computer Science & Engineering

### EXTERNAL EXAMINER

**ACKNOWLEDGEMENT**

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible, who’s constant guidance and encouragement crowns all effort with success.

I take this opportunity to express my profound gratitude and deep regards to My Guide **Mr.RAVINDHER, Assistant Professor**, Dept. of Computer Science & Engineering, Holy Mary Institute of Technology & Science for his / her exemplary guidance, monitoring and constant encouragement throughout the project work.

My special thanks to **Dr. B. Narsimha, Head of the Department**, Dept. of Computer Science & Engineering, Holy Mary Institute of Technology & Science who has given an immense support throughout the course of the project.

I also thank to **Dr. P. Bhaskara Reddy,** the **honorable Director** of my college Holy Mary Institute of Technology & Science for providing me the opportunity to carry out this work.

At the outset, I express my deep sense of gratitude to the beloved **Chairman A. Siddartha Reddy** of **Holy Mary Institute of Technology & Science**, for giving me the opportunity to complete my course of work

I am obliged to **staff members** of Holy Mary Institute of Technology & Science for the valuable information provided by them in their respective fields. I am grateful for their cooperation during the period of my assignment.

Last but not the least I thank **ALMIGHTY** and My **Parents**, and **Friends** for their constant encouragement without which this assignment would not be possible.

**P BHANUTRINADH (18C21A0562)**

**S SAIRAM (18C21A0564)**

**D RAJU (18C21A0570)**

# DECLARATION

This is to certify that the work reported in the present project titled **“HOSPITALITY MOBILE HEALTH-CARE USING CLOUD”** is a record of work done by me in the Department of Computer Science & Engineering, Holy Mary Institute of Technology and Science.

No part of the thesis is copied from books/journals/internet and wherever the portion is taken, the same has been duly referred in the text the reported are based on the project work done entirely by me not copied from any other source.

#### P BHANUTRINADH (18C21A0562)

**S SAIRAM (18C21A0564)**

**D RAJU (18C21A0570)**

**CONTENTS**

**CHAPTER NAME OF THE CHAPTER PAGE NO**

1. **INTRODUCTION**

PROBLEM STATEMENT…………………………… ……………3

OBJECTIVES ……………………………………………………….3

MOTIVATION ………………………………...................................3

EXISTING SYSTEM ……………………………………………….4

PROPOSED SYSTEM ………………………...................................5

2**. LITERATURE SURVEY**

EXISTING SYSTEM …………………………………………….......8

PROPOSAL SYSTEM ……………………………………………….8

APPLICATIONS ……………………………………………………..9

SUMMARY …………………………………………………………10

3.  **REQUIREMENTS SPECIFICATIONS**

SOFTWARE REQUIREMENTS …………………………………….11

HARDWARE REQUIREMENTS ………………...…………………11

4. **SYSTEM DESIGN**

SYSTEM ARCHITECTURE……. …………………………………15

5. **IMPLEMENTATION**

ENVIRONMENTAL SETUP………………………………………. 24

MODULE DESCRIPTION…………………………………………. 26

SOFTWARE DESCRIPTION………………………………………. 27

SAMPLE CODE……………………………………………………. 30

6.  **SYSTEM TESTING**

TESTING …………………………………………………………39

7. **RESULTSCREEN SHOTS** ………………………………44

8. **CONCLUSION** ………………………………………54

9. **BIBLIOGRAPHY**...…………………………………. 55

## ABSTRACT

Health-care industry is growing rapidly day by day. Rising hospital management costs, an aging population, a shortage of healthcare workers, challenges in accessing services, timely availability of information, issues of safety and quality, and rising consumerism are some of the facts of today’s healthcare system.

We have thus suggested the usage of an android mobile application which includes the use of cloud computing through which the doctor can easily use his/her mobile phone and access the data of patients using the modules. He/she can also use the mobile phone to suggest medicine and thus consume less time. The patient can also utilize the phone in order to view the medicine suggested by the doctor.

.

## CHAPTER 1

**INTRODUCTION**

### Overview

The Android SDK includes a comprehensive set of development tools. Requirements include Java Development Kit, the officially supported integrated development environment (IDE) is Eclipse (3.2 or later) using the Android Development Tools (ADT) Plug in, though developers may use any text editor to edit Java and XML files then use command line tools to create, build and debug Android applications.

New realities are placing pressure on the healthcare industry, and how patient care is delivered. Rising hospital management costs, an aging population, a shortage of healthcare workers, challenges in accessing services, timely availability of information, issues of safety and quality, and rising consumerism are some of the facts of today’s healthcare system.

Our project Hospitality in Android provides easiest way for all the doctors, patients and others to get all the information needed as quick as possible from anywhere in the world. Once the user registered in this he/she will get a unique username and password. After login the user has an ability to view his/her profile like which medicine is assigned to him and also his check up details within the mobile only. The doctors can search the patients and also view the patient profile and gives a medicine to the patient. After login the nurse has an ability to update the patient check up details.

Our project Hospitality in Android provides easiest way for all the doctors, patients and others to get all the information needed as quick as possible from anywhere in the world. Once the user registered in this he/she will get a unique username and password. After login the user has an ability to view his/her profile like which medicine is assigned to him and also his check up details within the mobile only. The doctors can search the patients and also view the patient profile and gives a medicine to the patient. After login the nurse has an ability to update the patient check up details.

## PROBLEM STATEMENT

USING JAVA PROGRAMMING LANGUAGE TO SOLVE THE HOSPITALITY MOBILE HEALTH\_CARE USING CLOUD COMPUTING SERVICES.

## OBJECTIVES

There are some system where we can use functionality related to hospital .In those system first user need to register and then he can get an appointment with the doctor, so in this manner he need to go through a lot of processes .After Consulting with the doctor only he can get clear picture of his health and then only he get some medicines.

New realities are placing pressure on the healthcare industry, and how patient care is delivered. Rising hospital management costs, an aging population, a shortage of healthcare workers, challenges in accessing services, timely availability of information, issues of safety and quality, and rising consumerism are some of the facts of today’s healthcare system.

One of the fastest growing industries now a days is mobile industry. There are many competitors in this area who are doing research and development on new platforms & user experience. One such technology is Android from Google which is supported for Google phones. These phones are described as next Generation mobiles [As described by Google].

## MOTIVATION

The problem arises when the patient needs to consult the doctor every now and then. This might be not feasible as the doctor has to be available every time and a prior appointment must be taken. This is done in order to meet the doctor to get a medicine based on the up gradation of the patients’ condition. Thus for every week or every few days the patient must keep meeting the doctor. The nurse must also keep checking the details of patient during the appointment. The patient might also not be in a position to visit the doctor every now and then. The desktops applications for hospital are also not that feasible. The use of this is only limited to certain extent. Thus we have to overcome the above problems.

To avoid all these, we have thus suggested the usage of an android mobile application which includes the use of cloud computing through which the doctor can easily use his/her mobile phone and access the data of patients using the modules. He/she can also use the mobile phone to suggest medicine and thus consume less time. The patient can also utilize the phone in order to view the medicine suggested by the doctor.

The use of this must be extended to other mediums too. The modules must have extensive operational facilities like a remainder service and alert messages to the patient to warn him/ her about the deadline for updating status. There must a way to access the details through a message sent to mobile phone instead of relying on the mails. These must be implemented in the project for future use.

## EXISTING SYSTEM

A common seamlessness problem is when an application's background process — for example, a service or broadcast receiver — pops up a dialog in response to some event. This may seem like harmless behavior, especially when you are building and testing your application in isolation, on the emulator. However, when your application is run on an actual device, your application may not have user focus at the time your background process displays the dialog. So it could end up that your application would display it's dialog behind the active application, or it could take focus from the current application and display the dialog in front of whatever the user was doing (such as dialing a phone call, for example). That behavior would not work for your application or for the user.

To avoid these problems, your application should use the proper system facility for notifying the user — the Notification classes. Using notifications, your application can signal the user that an event has taken place, by displaying an icon in the status bar rather than taking focus and interrupting the user.

The scope of this project is limited to only the android mobile users. The use of this must be extended to other mediums too. The modules must have extensive operational facilities like a remainder service and alert messages to the patient to warn him/ her about the deadline for updating status. There must a way to access the details through a message sent to mobile phone instead of relying on the mails. These must be implemented in the project for future use.

## PROPOSED SYSTEM

In our proposed system we are providing services so that a user(patient),nurse and doctor can access services from anywhere with the help of an android mobile .Generally in 70% of the cases there is no need to make a direct contact with the doctor by seeing symptoms only a doctor can suggest the medicine so in our proposed project a doctor can check the report of a patient which is sent by Nurse and then he can suggest medicines and ask for some more test if there is a necessity then doctor can check the patient directly.

It authenticates the user by matching the User Name and the Password (which in turn may be generated using some cryptographic algorithm) against the values stored in the database. It enables in creating new Users in the system i.e. it provides a registration form which makes the user to enter the new user’s details.

This over all Application was developed in android for the android mobile users.

Our project Hospitality in Android provides easiest way for all the doctors, patients and others to get all the information needed as quick as possible from anywhere in the world. Once the user registered in this he/she will get a unique username and password. After login the user has an ability to view his/her profile like which medicine is assigned to him and also his check up details within the mobile only. The doctors can search the patients and also view the patient profile and gives a medicine to the patient. After login the nurse has an ability to update the patient check up details.

### Operating Environment:

Software Requirements are Windows as Operating System, Linux operating system. Java 2 standard edition, Eclipse with AdtPlugin, android-sdk2.3. For the base SDK package, at least 600MB of available disk space. For each platform downloaded into the SDK, an additional 100MB is needed.

Hardware Requirements are Ram is 1GB Ram and above, hard disk is 50GB and above and Processor is Dual core or above.

### Java and its Features:

Java was conceived by James Gosling, Patrick Naughton, Chris Warth, Ed Frank and Mike Sheridan at SUN Microsystems Incorporation in the year 1991.It took 18 months to develop

the 1st working version. This language was initially called “OAK”, but was renamed “JAVA” in 1995, many more contributed to the design and evolution of the language.

#### Java Overview:

Java is a powerful but lean object-oriented programming language. It has generated a lot of excitement because it makes it possible to program for Internet by creating applets, programs that can be embedded in web page. The context of an applet can be an animation with sound, an interactive game or a ticker tape. With constantly updated stock prices. Applets can be just little decorations to liven up web page, or they can be serious applications like Word processor or Spreadsheet.

But Java is more than a programming language for writing Applets. It is being used more and more for writing standalone applications as well. It is becoming so popular that many people believe it will become standard language for both general purpose and Internet programming.

* + - * Java is simple, elegant, and powerful and easy-to-use.
      * Java is actually a platform consisting of 3 components:
      * Java Programming Language.
      * Java Library of Classes and Interfaces.
      * Java Virtual Machine

#### Java Development Environment

To code, edit, debug and test the java programs, one needs to have a java development environment. At the minimum this will consists of a java compiler interpreter and applet viewer where applets can be tested. Sun’s Java development kit (JDK) latest version is 2.2 can be freely downloaded from the Internet. Java compiler is available on DOS, Win95, WIN’NT, Solaris and MAC etc.

Data flow diagram is a graphical tool used to describe analyze the movement of data through a system manual or automated including the processes, stores of data, and delays in the system.

### Feasibility Study

The next step in analysis is to verify the feasibility of the proposed system. “All projects are feasible given unlimited resources and infinite time“. But in reality both resources and time are scarce. Project should confirm to time bounce and should be optimal in their consumption of resources.

* Technical feasibility
* Operational feasibility
* Economical feasibility

#### Technical Feasibility

As we are developing this Application on Java 2 platform edition which is an open source and free of cost. Once we started developing this application in Java 2 platform editions then there is no need of purchasing any special software or application software for support.

#### Operational Feasibility

To determine the operational feasibility of the system we should take into consideration the awareness level of the users. Users who are using this Application don’t require much knowledge of how to use. Everything will be understood by user once he sees the application.

#### Economical Feasibility

To decide whether a project is economically feasible, or not we have to consider various factors as:

* + - * Cost benefit analysis
      * Long-term returns
      * Maintenance costs

# Chapter 2

LITERATURE SURVEY

# Existing System

In the existing system if a patient wants to take an appointment of a doctor he has to go to hospital first. Even the people can’t get the correct information about doctors, their details and different hospitals available in a particular city. The only way to get all these are through directly contacting particular persons personally. A person suffering with some problem cannot get correct and immediate prescription or treatment until he meets the right doctor.Our Application provides interface to the users/patients who are busy and far away from city can access this application to know the prescribed medicines.

#### Limitations of Existing System

* Lack of security of data.
* Time consuming.
* Consumes large volume of paper work.
* Manual work

# Proposal System

This application is intended for patients and all the people who are facing minute problems in their health. In the proposed system the patient is able to view his/her check up details and prescribed medicines within his/her mobile only no need to go to hospital again and again.

Which makes people aware of what kind of medicine should be taken for type of health problem they are facing it makes access to the information easier

#### Advantages of Proposed System

* Security of data.
* Patients are easily allocated to the doctors.
* Doctors Search is possible.
* Ensure data accuracy's.
* Minimize manual data entry.

**Applications**

The Project is divided into 4 modules: 1.Doctor Module

* 1. Patient Module
  2. Nurse Module

### Doctor Module

It is one of the major modules in this software. The doctor registers to the application and login to the application.Its main work is that he/she can view patients profile and details and update the Medicine required for the patient.

### Patient Module

It is the main module of the application. The patient should first register to the application and login to it. A profile is created for the patient he/she can search for the medicine for the health problem they are facing.

### Nurse Module

It is the major module who operates the application register and login into the application. It views the patients profile about the requirements of the patients and then update the details into it.

## User Interfaces

This Software include sample screen images such as user login, interacting with database, any GUI (web based pages using ASP.net) standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on.

## Hardware Interfaces

We require LAN connection for interacting with database and local computers for any help or any other requirement. We use TCP/IP protocol for communicating with local hosts. We also need system with dual core processor; 1GB RAM and database memory and 50GB Hard Disk.

## Software Interfaces

We use java Programming language for writing the code for the project. ASP.Net 3.5 for creating the web pages, using GUI for login screens and interacting with database.Android sdk for mobile application. Cloud and s3 is used for creating the local and global database (server). Operating system: Windows XP or higher version.

## Communications Interfaces

The communications functions required by this product are LAN connection with in the whole company so that the Admin, Doctor and Patient can interact with each other. We use TCP/IP protocol.

## Summary

In our user manual we are going to keep the information regarding our product which can be understandable by a new person who is going to use it. If a new person is using it online help will provided in that we are going to explain each and every step clearly by our product can be useful for any user.

CHAPTER 3

## REQUIREMENTS SPECIFICATIONS

* + - **Application framework** enabling reuse and replacement of components
    - **Dalvik virtual machine** optimized for mobile devices
    - **Integrated browser** based on the open source Web Kit engine
    - **Optimized graphics** powered by a custom 2D graphics library; 3D graphics based on the OpenGL ES 1.0 specification (hardware acceleration optional)
    - **SQLite** for structured data storage
    - **Media support** for common audio, video, and still image formats (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
    - **GSM Telephony** (hardware dependent)
    - **Bluetooth, EDGE, 3G, and WiFi** (hardware dependent)
    - **Camera, GPS, compass, and accelerometer** (hardware dependent)
    - **Rich development environment** including a device emulator, tools for debugging, memory and performance profiling, and a plugin for the Eclipse IDE

## Software requirements

* + - The platform that is used is Java
    - Android SDK 1.5 or later
    - Eclipse Ganymede IDE
    - Operating System can be Windows XP, LINUX and Mac etc.
    - Content providers.
    - Cloud simpleDB and S3.

## Hardware Requirements:

* Ram : 1GB Ram and above
* Hard Disk : 50GB and above
* Processor : Dual core and above

CHAPTER 4

## SYSTEM DESIGN

System design is transition from a user oriented document to programmers or data base personnel. The design is a solution, how to approach to the creation of a new system. This is composed of several steps. It provides the understanding and procedural details necessary for implementing the system recommended in the feasibility study. Designing goes through logical and physical stages of development, logical design reviews the present physical system, prepare input and output specification, details of implementation plan and prepare a logical design walkthrough.

The database tables are designed by analyzing functions involved in the system and format of the fields is also designed. The fields in the database tables should define their role in the system. The unnecessary fields should be avoided because it affects the storage areas of the system. Then in the input and output screen design, the design should be made user friendly. The menu should be precise and compact.

Designing for performance:

An Android application should be fast. Well, it's probably more accurate to say that it should be *efficient*. That is, it should execute as efficiently as possible in the mobile device environment, with its limited computing power and data storage, smaller screen, and constrained battery life.

As you develop your application, keep in mind that, while the application may perform well enough in your emulator, running on your dual-core development computer, it will not perform that well when run a mobile device — even the most powerful mobile device can't match the capabilities of a typical desktop system. For that reason, you should strive to write efficient code, to ensure the best possible performance on a variety of mobile devices.

Generally speaking, writing fast or efficient code means keeping memory allocations to a minimum, writing tight code, and avoiding certain language and programming idioms that can subtly cripple performance. In object-oriented terms, most of this work takes place at the *method* level, on the order of actual lines of code, loops, and so on.

There are two basic rules for resource-constrained systems:

* Don't do work that you don't need to do.
* Don't allocate memory if you can avoid it.

All the tips below follow from these two basic tenets.

Some would argue that much of the advice on this page amounts to "premature optimization." While it's true that micro-optimizations sometimes make it harder to develop efficient data structures and algorithms, on embedded devices like handsets you often simply have no choice. For instance, if you bring your assumptions about VM performance on desktop machines to Android, you're quite likely to write code that exhausts system memory. This will bring your application to a crawl — let alone what it will do to other programs running on the system!

That's why these guidelines are important. Android's success depends on the user experience that your applications provide, and that user experience depends in part on whether your code is responsive and snappy, or slow and aggravating. Since all our applications will run on the same devices, we're all in this together, in a way. Think of this document as like the rules of the road you had to learn when you got your driver's license: things run smoothly when everybody follows them, but when you don't, you get your car smashed up.Before we get down to brass tacks, a brief observation: Nearly all issues described below are valid whether or not the VM features a JIT compiler. If I have two methods that accomplish the same thing, and the interpreted execution of foo() is faster than bar(), then the compiled version of foo() will probably be as fast or faster than compiled bar(). It is unwise to rely on a compiler to "save" you and make your code fast enough.

### Software design

In designing the software following principles are followed:

* + - **Modularity and partitioning**: Software is designed such that, each system should consists of hierarchy of modules and serve to partition into separate function.
    - **Coupling:** Modules should have little dependence on other modules of a system.
    - **Cohesion:** Modules should carry out in a single processing function.
    - **Shared use:** Avoid duplication by allowing a single module is called by other that need the function it provides

### Input/output Design

#### Input design

Considering the requirements, procedures to collect the necessary input data in most efficiently designed. The input design has been done keeping in view that, the interaction of the user with the system being the most effective and simplified way.

Also the measures are taken for the following

* + - Controlling the amount of input
    - Avoid unauthorized access to the Universal Dossier
    - Eliminating extra steps
    - Keeping the process simple
    - At this stage the input forms and screens are designed.

#### 4.2.2 Output design

All the screens of the system are designed with a view to provide the user with easy operations in simpler and efficient way, minimum key strokes possible. Instructions and important information is emphasized on the screen. Almost every screen is provided with no error and important messages and option selection facilitates. Emphasis is given for speedy processing and speedy transaction between the screens. Each screen assigned to make it as much user friendly as possible by using interactive procedures. So to say user can operate the system without much help from the operating manual.

### Design and Implementation Constraints

All modules are coded thoroughly based on requirements from software organization. The software is designed in such a way that the user can easily interact with the screen. Software is designed in such a way that it can be extended to the real time business*.*

### System Architecture

The Android SDK includes a mobile device emulator a virtual mobile device that runs on your computer. The emulator lets you develop and test Android applications without using a physical device. SQLite is an Open Source Database which is embedded into Android. SQLite supports standard relational database features like SQL syntax, transactions and prepared statements. In addition it requires only little memory at runtime .

Application

Modules (Fu

Database

Android SDK

[ Emulator ]

### Figure 4.1 System Architecture

The GUI will be a android mobile application and also includes the business functionalities in Business Access Layer and the data access functionalities in data access layer. The frontend(GUI) which is using android layout which is based on xml Coding will be done using the java language in code behind the mobile application.

#### Business Access Layer

Includes all the necessary business logic required for any business functionalities

#### Data Access Layer

Includes all the necessary data access functionalities

**Database**

**DAL**

**BAL**

**GUI**

### Figure 4.2 Application Architecture

The Unified Modelling Language (UML) is a standard language for writing software blue prints. The UML is a language for

* Visualizing
* Specifying
* Constructing
* Documenting the artefacts of a software intensive system.

The UML is a language which provides vocabulary and the rules for combining words in that vocabulary for the purpose of communication. A modelling language is a language whose vocabulary and the rules focus on the conceptual and physical representation of a system. Modelling yields an understanding of a system.

Design is concerned with identifying software components, specifying relationships among components, maintaining a record of design decisions. Design consists of architectural design and detailed design.

1. Architectural Design involves identifying the software components, decoupling and decomposing them in to processing modules and conceptual data structures and specifying the interconnection between the components.
2. Detailed Design is concerned with the details of how to package the processing modules and how to implement the processing algorithms, data structures and interconnection between them



Registration

update status of patient

Nurse

send mails

#### Fig: 4.3 Use Case Diagram For Nurse



Registration

Check status

Doctor

Suggest Medicine

Check Patient

**Fig: 4.4 Use Case Diagram For Doctor**



Register

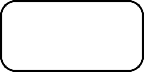
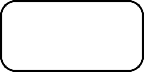
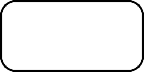
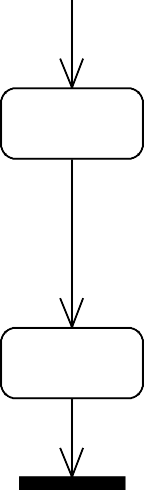
login

patient

Get medicine

#### Fig: 4.5 Use Case Diagram For Patient





Start

Register

Login

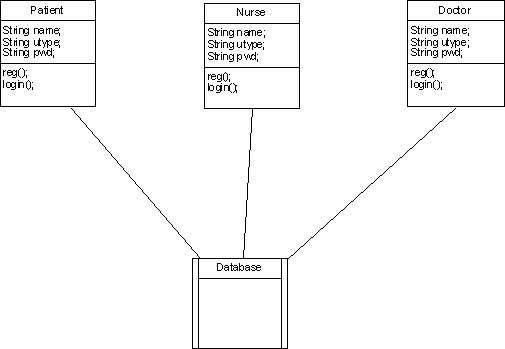
patient

Doctor

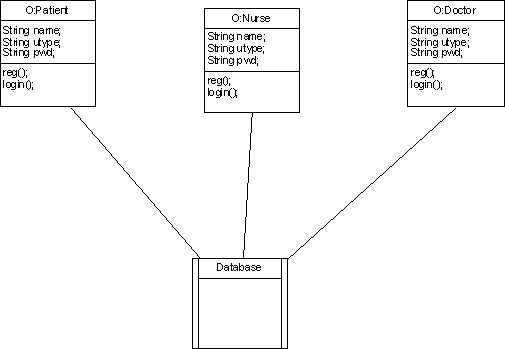
Nurse

end

#### Fig: 4.6 Activity Diagram



**Fig: 4.7 Class Diagram**

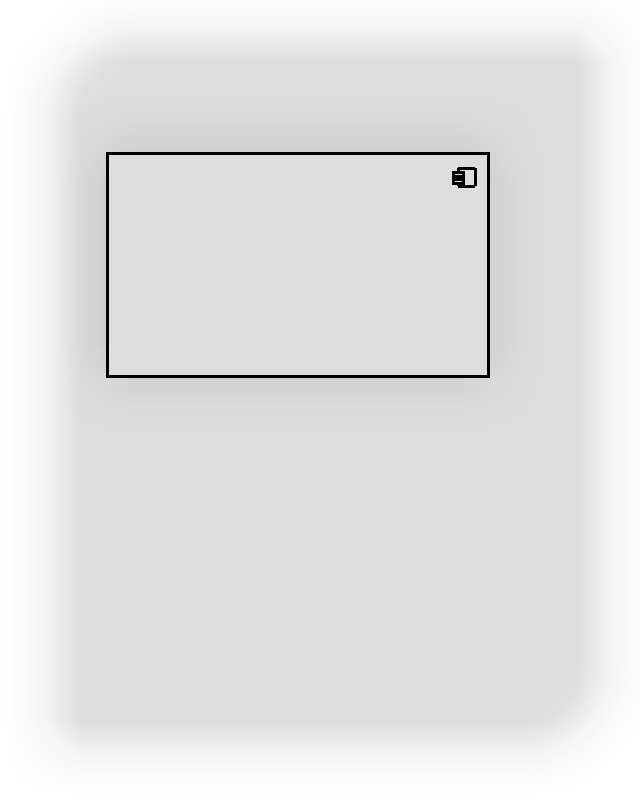


#### Fig: 4.8 Object Diagram



Actor

**Fig: 4.9 Architecture**



Servlet,Jsp(J2EE)

Application

Hospitality

Database

JSPEngine

ServletEngine

## Chapter 5 IMPLEMENTATION

**Environmental Setup**

Overview of Software Development Tools The life-cycle paradigm demands a systematic, sequential approach to software development that begins at the system level and progresses through requirements analysis, design, coding, testing and maintenance.

#### Installation Set up for JDK

The implementation phase of software development involves translation of design specifications source code and debugging, documentation and unit testing of the source code.

To run (or debug) your application, select **Run** > **Run** (or **Run** > **Debug**) from the Eclipse main menu. The ADT plugin will automatically create a default launch configuration for the project.

When you choose to run or debug your application, Eclipse will perform the following: Compile the project (if there have been changes since the last build).

Create a default launch configuration (if one does not already exist for the project).

Install and start the application on an emulator or device (based on the Deployment Target defined by the run configuration).

By default, Android application run configurations use an "automatic target" mode for selecting a device target. For information on how automatic target mode selects a deployment target, see Automatic and manual target modes below.

If debugging, the application will start in the "Waiting For Debugger" mode. Once the debugger is attached, Eclipse will open the Debug perspective.

To set or change the launch configuration used for your project, use the launch configuration manager. See Creating a Launch Configuration for information.

To create or modify a launch configuration, follow these steps as appropriate for your Eclipse version:

Open the run configuration manager.

In Eclipse 3.3 (Europa), select **Run** > **Open Run Dialog** (or **Open Debug Dialog**)

In Eclipse 3.4 (Ganymede), select **Run** > **Run Configurations** (or **Debug Configurations**) Expand the **Android Application** item and create a new configuration or open an existing one.

To create a new configuration:

Select **Android Application** and click the *New launch configuration* icon above the list (or, right-click **Android Application** and click **New**).

Enter a Name for your configuration.

In the Android tab, browse and select the project you'd like to run with the configuration.

To open an existing configuration, select the configuration name from the list nested below

**Android Application.**

Adjust your desired launch configuration settings.

In the Target tab, consider whether you'd like to use Manual or Automatic mode when selecting an AVD to run your application. See the following section on Automatic and manual target modes).

You can specify any emulator options to the Additional Emulator Command Line Options field. For example, you could add -scale 96dpi to scale the AVD's screen to an accurate size, based on the dpi of your computer monitor. For a full list of emulator options, see the Android Emulator document.

## MODULE DESCRIPTION

The analysis phase consists of two sub phases: planning and requirements definition. During planning the activities that are performed are - understand the customer’s problem, developing a recommended solution. Requirements definition is concerned with identifying the basic functions of a software component in a hardware/software/people system.

The various modules are**:**

* Patient : Patient details like Name, Age, Sex, Address, Contact number, etc.
* Nurse : Nurse details like Name, Age, Sex, Address, Contact number, etc.
* Doctor : Doctor details like Name, Age, Sex, Address, Contact number, etc.

#### Patient:

Every user (patient) who wants to use this product needs to register prior to get any Consultation, treatment or investigations done. Registration of patients involves accepting certain general and demographic information about the patient. The patient is allocated a User name and a Password. The Patient Username will remain same for his all subsequent visits to the Application (hospitality). The following information is required for the registration of patient.

Patient details like Name, Age, Sex, Address, Contact number, etc.

#### Nurse:

Every nurse who wants to use this product needs to register first. Registration of nurse involves

accepting certain general and demographic information about the nurse. The nurse is allocated a Username and a Password. The nurse Username will remain same for her all subsequent visits to the Application (hospitality). After registration a nurse can check the different things like the Blood Pressure, ECG, Blood Group etc and update these things in a database so that doctor can check related information to particular patient. The following information is required for the registration of Nurse:

Nurse details like Name, Age, Sex, Address, Contact number, etc.

**Doctor:**

Every Doctor who wants to use this product needs to register first. Registration of Doctor involves accepting certain general and demographic information about the nurse. The Doctor is

allocated a Username and a Password. The Patient Username will remain same for him all subsequent visits to the Application (hospitality).

The following information is required for the registration of Doctor: Doctor details like Name, Age, Sex, Address, Contact number, etc.

## SOFTWARE DESCRIPTION

### User Interfaces

This application include GUI standards or product family style guides that are to be followed, screen layout constraints, buttons and functions that will appear on every screen, error message display standards, and so on.

### User Documentation

In our user manual we are going to keep the information regarding our product, which can be understandable by a new person who is going to use it. If a new person is using it, online help will be provided in that. We are going to explain each and every step clearly about our product so that any user can easily understand it.

### User Classes and Characteristics

End user of the application is the mobile phone user. The user enter their profile, doctor can check the report of a patient which is sent by Nurse and then he can suggest medicines and ask for some more test if there is a necessity then doctor can check the patient directly.

### Document Conventions

To prepare this SRS we had used the lettering style of Times New Roman and Font size for sub headings is 14 with bold. The matter which is mentioned in this SRS is 12 with a lettering style Times New Roman. Then headings are of lettering style Times New Roman with Font size 12. Then the important points in our SRS are shown in italics.

#### Libraries

Android includes a set of C/C++ libraries used by various components of the Android system. These capabilities are exposed to developers through the Android application framework. Some of the core libraries are listed below:

**System C library** - a BSD-derived implementation of the standard C system library (libc), tuned for embedded Linux-based devices

**Media Libraries** - based on PacketVideo's OpenCORE; the libraries support playback and recording of many popular audio and video formats, as well as static image files, including MPEG4, H.264, MP3, AAC, AMR, JPG, and PNG

**Surface Manager** - manages access to the display subsystem and seamlessly composites 2D and 3D graphic layers from multiple applications

**LibWebCore** - a modern web browser engine which powers both the Android browser and an embeddable web view

**SGL** - the underlying 2D graphics engine

**3D libraries** - an implementation based on OpenGL ES 1.0 APIs; the libraries use either hardware 3D acceleration (where available) or the included, highly optimized 3D software rasterizer

**FreeType** - bitmap and vector font rendering

**SQLite** - a powerful and lightweight relational database engine available to all applications

#### Android Runtime

Android includes a set of core libraries that provides most of the functionality available in the core libraries of the Java programming language.

Every Android application runs in its own process, with its own instance of the Dalvik virtual machine. Dalvik has been written so that a device can run multiple VMs efficiently. The Dalvik VM executes files in the Dalvik Executable format which is optimized for minimal memory footprint. The VM is register-based, and runs classes compiled by a Java language compiler that have been transformed into the .dex format by the included "dx" tool.

The Dalvik VM relies on the Linux kernel for underlying functionality such as threading and low-level memory management.

An Android code editor that helps you write valid XML for your Android manifest and resource files. It will even export your project into a signed APK, which can be distributed to users.

To begin developing Android applications in the Eclipse IDE with ADT, you first need to download the Eclipse IDE and then download and install the ADT plugin. To do so, follow the steps given in Installing the ADT Plugin.

#### Developing in eclipse with ADT

The Android Development Tools (ADT) plugin for Eclipse adds powerful extensions to the Eclipse integrated development environment. It allows you to create and debug Android applications easier and faster. If you use Eclipse, the ADT plugin gives you an incredible boost in developing Android applications:

It gives you access to other Android development tools from inside the Eclipse IDE. For example, ADT lets you access the many capabilities of the DDMS tool: take screenshots, manage port-forwarding, set breakpoints, and view thread and process informationd irectly from Eclipse.

It provides a New Project Wizard, which helps you quickly create and set up all of the basic files you'll need for a new Android application. It automates and simplifies the process of building your Android application.

## SAMPLE CODE

#### Creating an Android project

The ADT plug-in provides a New Project Wizard that you can use to quickly create a new Android project (or a project from existing code). To create a new project:

* + - 1. Select **File** > **New** > **Project**.
      2. Select **Android** > **Android Project**, and click **Next**.
      3. Select the contents for the project:

Enter a *Project Name*. This will be the name of the folder where your project is created.

Under Contents, select **Create new project in workspace**. Select your project workspace location.

Under Target, select an Android target to be used as the project's Build Target. The Build Target specifies which Android platform you'd like your application built against.

Unless you know that you'll be using new APIs introduced in the latest SDK, you should select a target with the lowest platform version possible, such as Android 1.1.

Enter an *Application name*. This is the human-readable title for your application — the name that will appear on the Android device.

Enter a *Package name*. This is the package namespace (following the same rules as for packages in the Java programming language) where all your source code will reside.

Select *Create Activity* (optional, of course, but common) and enter a name for your main Activity class.

Enter a *Min SDK Version*. This is an integer that indicates the minimum API Level required to properly run your application. Entering this here automatically sets the minSdkVersion attribute in the <uses-sdk> of your Android Manifest file. If you're unsure of the appropriate API Level to use, copy the API Level listed for the Build Target you selected in the Target tab.

* + - 1. Click **Finish**.

Once you complete the New Project Wizard, ADT creates the following folders and files in your new project:

src/

Includes your stub Activity Java file. All other Java files for your application go here.

*<Android Version>*/ (e.g., Android 1.1/)

Includes the android.jar file that your application will build against. This is determined by the build target that you have chosen in the *New Project Wizard*.

gen/

This contains the Java files generated by ADT, such as your R.java file and interfaces created from AIDL files.

assets/

This is empty. You can use it to store raw asset files. See Resources and Assets.

res/

A folder for your application resources, such as drawable files, layout files, string values, etc. See Resources and Assets.

AndroidManifest.xml

The Android Manifest for your project. See The AndroidManifest.xml File.

This file contains project settings, such as the build target. This files is integral to the project, as such, it should be maintained in a Source Revision Control system. It should never be edited manually — to edit project properties, right-click the project folder and select "Properties".

Before you can run your application on the Android Emulator, you **must** create an Android Virtual Device (AVD). An AVD is a configuration that specifies the Android platform to be used on the emulator. You can read more in the Android Virtual Devices document, but if you just want to get started, follow the simple guide below to create an AVD.

If you will be running your applications only on actual device hardware, you do not need an AVD — see Developing On a Device for information on running your applicaiton.

With ADT 0.9.3 and above, the Android SDK and AVD Manager provides a simple graphical interface for creating and managing AVDs. (If you're using ADT version 0.9.1 or older, you must use the android tool to create your AVDs—read the AVD guide to Creating an AVD.)

To create an AVD with the AVD Manager:

Select **Window > Android SDK and AVD Manager**, or click the Android SDK and AVD Manager icon (a black device) in the Eclipse toolbar.

In the Virtual Devices panel, you'll see a list of existing AVDs. Click **New** to create a new AVD.

Fill in the details for the AVD.

Give it a name, a platform target, an SD card image (optional), and a skin (HVGA is default). Click **Create AVD.**

Your AVD is now ready and you can close the AVD Manager. In the next section, you'll see how the AVD is used when launching your application on an emulator.

For more information about AVDs, read the Android Virtual Devices documentation.

**Note:** Before you can run your application, be sure that you have created an AVD with a target that satisfies your application's Build Target. If an AVD cannot be found that meets the requirements of your Build Target, you will see a console error telling you so and the launch will be aborted.

The run configuration specifies the project to run, the Activity to start, the emulator options to use, and so on. When you first run a project as an *Android Application*, ADT will automatically create a run configuration. The default run configuration will launch the default project Activity and use automatic target mode for device selection (with no preferred AVD). If the default settings don't suit your project, you can customize the launch configuration or even create a new.

Object creation is never free. A generational GC with per-thread allocation pools for temporary objects can make allocation cheaper, but allocating memory is always more expensive than not allocating memory.

If you allocate objects in a user interface loop, you will force a periodic garbage collection, creating little "hiccups" in the user experience.

Thus, you should avoid creating object instances you don't need to. Some examples of things that can help:

Map myMap1 = new HashMap(); HashMap myMap2 = new HashMap();

#### Cache Field Lookup:

Accessing object fields is much slower than accessing local variables. Instead of writing: for (int i = 0; i < this.mCount; i++)

dumpItem(this.mItems[i]);

**You should write:**

int count = this.mCount; Item[] items = this.mItems; for (int i = 0; i < count; i++) dumpItems(items[i]);

(We're using an explicit "this" to make it clear that these are member variables.)

A similar guideline is never call a method in the second clause of a "for" statement. For example, the following code will execute the getCount() method once per iteration, which is a huge waste when you could have simply cached the value as an int:

for (int i = 0; i < this.getCount(); i++) dumpItems(this.getItem(i));

It's also usually a good idea to create a local variable if you're going to be accessing an instance field more than once. For example:

protected void drawHorizontalScrollBar(Canvas canvas, int width, int height) { if (isHorizontalScrollBarEnabled()) {

int size = **mScrollBar**.getSize(*false*); if (size <= 0) {

size = mScrollBarSize;

}

**mScrollBar**.setBounds(0, *height* - size, width, height); **mScrollBar**.setParams( computeHorizontalScrollRange(), computeHorizontalScrollOffset(), computeHorizontalScrollExtent(), *false*); **mScrollBar**.draw(canvas);

}

}

That's four separate lookups of the member field mScrollBar. By caching mScrollBar in a local stack variable, the four member field lookups become four stack variable references, which are much more efficient.

Incidentally, method arguments have the same performance characteristics as local variables.

Declare Constants Final:

Consider the following declaration at the top of a class: static int intVal = 42;

static String strVal = "Hello, world!";

The compiler generates a class initializer method, called <clinit>, that is executed when the class is first used. The method stores the value 42 into intVal, and extracts a reference from the classfile string constant table for strVal. When these values are referenced later on, they are accessed with field lookups.

We can improve matters with the "final" keyword:

static final int intVal = 42;

static final String strVal = "Hello, world!";

The class no longer requires a <clinit> method, because the constants go into classfile static field initializes, which are handled directly by the VM. Code accessing intVal will use the integer value 42 directly, and accesses to strVal will use a relatively inexpensive "string constant" instruction instead of a field lookup.

Declaring a method or class "final" does not confer any immediate performance benefits, but it does allow certain optimizations. For example, if the compiler knows that a "getter" method can't be overridden by a sub-class, it can inline the method call.

You can also declare local variables final. However, this has no definitive performance benefits. For local variables, only use "final" if it makes the code clearer (or you have to, e.g. for use in an anonymous inner class).

Designing For Responsiveness:

It's possible to write code that wins every performance test in the world, but still sends users in a fiery rage when they try to use the application. These are the applications that aren't *responsive* enough — the ones that feel sluggish, hang or freeze for significant periods, or take too long to process input.

In Android, the system guards against applications that are insufficiently responsive for a period of time by displaying a dialog to the user, called the Application Not Responding (ANR) dialog. The user can choose to let the application continue, but the user won't appreciate having to act on this dialog every time he or she uses your application. So it's important to

design responsiveness into your application, so that the system never has cause to display an ANR to the user.

Generally, the system displays an ANR if an application cannot respond to user input. For example, if an application blocks on some I/O operation (frequently a network access), then the main application thread won't be able to process incoming user input events. After a time, the system concludes that the application has hung, and displays the ANR to give the user the option to kill it.

Similarly, if your application spends too much time building an elaborate in-memory structure, or perhaps computing the next move in a game, the system will conclude that your application has hung. It's always important to make sure these computations are efficient using the techniques above, but even the most efficient code still takes time to run.

In both of these cases, the fix is usually to create a child thread, and do most of your work there. This keeps the main thread (which drives the user interface event loop) running, and prevents the system from concluding your code has frozen. Since such threading usually is accomplished at the class level, you can think of responsiveness as a *class* problem. (Compare this with basic performance, which was described above as a *method*-level concern.)This document discusses how the Android system determines whether an application is not responding and provides guidelines for ensuring that your application is responsive.

What Triggers ANR:

In Android, application responsiveness is monitored by the Activity Manager and Window Manager System services. Android will display the ANR dialog for a particular application when it detects one of the following conditions:

How to Avoid ANR:

Given the above definition for ANR, let's examine why this can occur in Android applications and how best to structure your application to avoid ANR.

Android applications normally run entirely on a single (i.e. main) thread. This means that anything your application is doing in the main thread that takes a long time to complete can trigger the ANR dialog because your application is not giving itself a chance to handle the input event or Intent broadcast.

Therefore any method that runs in the main thread should do as little work as possible. In particular, Activities should do as little as possible to set up in key life-cycle methods such as on Create () and on Resume (). Potentially long running operations such as network or database operations, or computationally expensive calculations such as resizing bitmaps should be done in a child thread (or in the case of databases operations, via an asynchronous request). However, this does not mean that your main thread has to block while waiting for the child thread to complete nor should you call Thread. Wait () or Thread. Sleep (). Instead of blocking while waiting for a child thread to complete, your main thread should provide a Handler for child threads to post back to upon completion. Designing your application in this way will allow your main thread to remain responsive to input and thus avoid ANR dialogs caused by the 5 second input event timeout. These same practices should be followed for any other threads that display UI, as they are also subject to the same timeouts.

The specific constraint on Intent Receiver execution time emphasizes what they were meant to do: small, discrete amounts of work in the background such as saving a setting or registering a Notification. So as with other methods called in the main thread, applications should avoid potentially long-running operations or calculations in Broadcast Receivers. But instead of doing intensive tasks via child threads (as the life of a Broadcast Receiver is short), your application should start a Service if a potentially long running action needs to be taken in response to an Intent broadcast. As a side note, you should also avoid starting an Activity from an Intent Receiver, as it will spawn a new screen that will steal focus from whatever application the user is currently has running. If your application has something to show the user in response to an Intent broadcast, it should do so using the Notification Manager.

Designing for Seamlessness:

Even if your application is fast and responsive, certain design decisions can still cause problems for users — because of unplanned interactions with other applications or dialogs, inadvertent loss of data, unintended blocking, and so on. To avoid these problems, it helps to understand the context in which your applications run and the system interactions that can affect your application. In short, you should strive to develop an application that interacts seamlessly with the system and with other applications.

Another example of a seamlessness problem is when an activity inadvertently loses state or user data because it doesn't correctly implement the on Pause () and other lifecycle methods. Or, if your application exposes data intended to be used by other applications, you should expose it via a Content Provider, rather than (for example) doing so through a world-readable raw file or database.

What those examples have in common is that they involve cooperating nicely with the system and other applications. The Android system is designed to treat applications as a sort of federation of loosely-coupled components, rather than chunks of black-box code. This allows you as the developer to view the entire system as just an even-larger federation of these components. This benefits you by allowing you to integrate cleanly and seamlessly with other applications, and so you should design your own code to return the favor.

This document discusses common seamlessness problems and how to avoid them. It covers these topics:

A classic example of a good use of this behavior is a mail application. If the user was composing an email when another Activity started up, the application should save the in- process email as a draft.

## Chapter 6 TESTING

### Software Testing

Software testing is a critical element of software quality and assurance and represents ultimate review of specifications, design and coding. Testing is an exposure of the system to trial input to see whether it produces correct output.

### Testing Phases

It involves two kinds of testing:

1. In integration testing the individual program units or programs are integrated and tested.
2. Acceptance Testing involves planning and execution of various types of tests in order to demonstrations that the implemented software satisfies the stated requirements.

### Testing Activities

* 1. INSPECTING COMPONENTS: This finds faults in the individual component through the manual inspection of its source code.
  2. UNIT TESTING: This find faults by isolating an individual component using test stubs and drivers and by exercising the components using a test case.
  3. INTEGRATION TESTING: This finds faults by integrating several components together. System testing, which focuses on the complete system, its functional and non-functional requirements and its target environment.

#### Unit Testing

Unit testing focuses on the building blocks of the software system, that is, objects and subsystems. They are three motivations behind focusing on components. First, unit testing reduces the complexity of the overall test activities, allowing us to focus on smaller units of the system. Unit testing makes it easier to pinpoint and correct faults given that few computers are involved in this test. Unit testing allows parallelism in the testing activities; that is each component can be tested independently of one another.

The specific candidates for unit testing are chosen from the object model and the system decomposition of the system. In principle, all the objects developed during the development process should be tested. Which is often not feasible because of time and budget?

Many unit testing techniques have been devised. Some of them are**:**

#### Equivalence Testing

It is a black box testing technique that minimizes the number of test cases. The possible inputs are partitioned into equivalence classes, and a test case is selected for each class. The assumption of equivalence testing is that the system usually behaves in similar ways for all members of a class. To test the behavior associated with an equivalence class, we only need to test one member of the class. Equivalence testing consists of two steps: identification of the equivalence classes and selection of the test inputs.

The following criteria are used for the equivalence testing:

* + - 1. COVERAGE: Every possible input belongs to one of the equivalent classes.
      2. DISJOINTEDNESS: No input belongs to one of the equivalent classes.
      3. REPRESENTATION: If the execution demonstrates an error when a particular member of an equivalence class is used as input, then the same error can be detected by using any other member of the class as input.

#### Boundary Testing:

Boundary testing is a special case of equivalence testing and focuses on the conditions at the boundary of the equivalence classes. Rather than selecting any element in the equivalence class, boundary testing requires that the element be selected from the “edges” of the equivalence class.

A disadvantage of equivalence class and boundary testing is that these techniques do not explore combination of test input data.

#### Path Testing

Path testing is a white box testing technique that identifies faults in the implementation of the component. The assumption behind path is that, by exercising all possible paths through the code at least once, most faults will trigger failures. The identification of paths requires knowledge of the source code and data structures.

The path testing technique was developed for imperative languages. Object oriented language introduce several difficulties when using path testing.

* + - 1. POLYMORPHISM: Polymorphism enables messages to be bound to different methods bases on the class of the target. Although this enables developers to reuse code across a large number of classes, it is also introduce more cases to test.
      2. SHORTER METHODS: Methods in object oriented language have the tendency to be shorter then procedures and functions in imperative languages. This decreases the likelihood of control flow faults, which can be uncovered using the path testing technique.

#### State Based Testing

Object oriented languages introduce the opportunity for new types of faults in object-oriented systems.

State based testing is a recent testing technique, which focuses on object-oriented systems. Most testing techniques which focus on selecting a number of test inputs for a given state of the system, exercising a component or a system, and comparing the observed outputs with java. State based testing focuses on comparing the resulting state of the system with the expected state. In the context of a class, state-based testing consists of deriving test cases from the UML state chart diagram for the class.

#### Integration Testing

It detects faults that have not been detected during unit testing, by focusing on small group of components.

### Test Case Design

The design of tests for software and other engineering products can be as challenging as the initial design of the product. Test case methods provide the developer with a systematic

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S. No | Action Required | Expected Output | Observed Output | Result |
| 1 | Enter login details | Login successful | Login successful | Pass |
| 2 | View patients’ details | Details page appears | Details obtained | Pass |
| 3 | Suggest medicine | Medicine updated | Medicine updated | Pass |
| 4 | Log out | Doctor should log out | Log out successful | Pass |

approach to testing. Moreover, these methods provide a mechanism that can help to ensure the completeness of tests and provide the highest like hood for uncovering errors in software.

Any Engineered product can be tested in either of the two ways:

* + 1. Knowing the specified function that a product has been designed to perform, tests can be conducted. These tests demonstrate whether each function is full operational and at the same time searches for errors in each function.
    2. Knowing the internal workings of a product, tests can be conducted to ensure that internal operations are performed according to specifications and all internal components hence been adequately exercised.

Test case design methods are divided into two types:

* + - 1. White-box testing
      2. Black-box testing

### Test cases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S. No | Action Required | Expected Output | Observed Output | Result |
| 1 | Enter login details | Login successful | Login successful | Pass |
| 2 | View medicine details | Details page appears | Details obtained | Pass |
| 3 | Enter patients’ | Details updated | Details updated | Pass |
| 4 | Log out | Patient should log out | Log out successful | Pass |

Table 5.2 Test cases for nurse module

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S. No | Action Required | Expected Output | Observed Output | Result |
| 1 | Enter login details | Login successful | Login successful | Pass |
| 2 | View patients’ details | Details page appears | Details obtained | Pass |
| 3 | Enter patients’ details | Details updated | Details updated | Pass |
| 4 | Log out | Nurse should log out | Log out successful | Pass |

Table 5.3 Test cases for patient module

## Chapter 7 RESULT



Figure 6.1 Home Screen

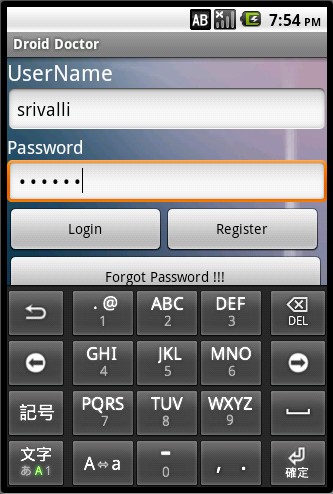


Figure 6.2 Login Screen

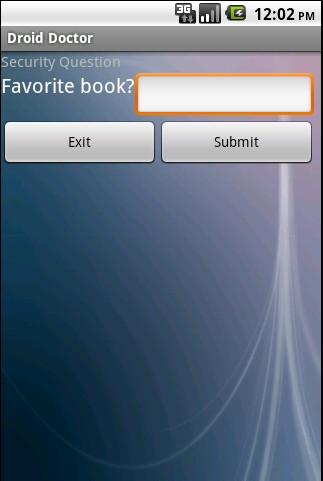


Figure 6.3 Security Question

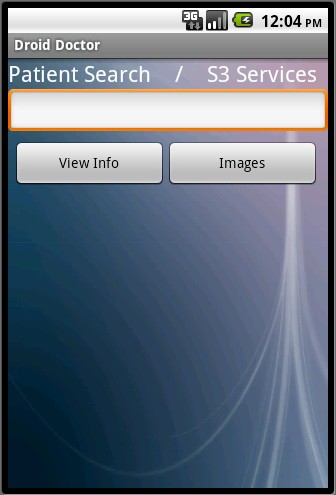


Figure 6.4 Patient Search



Figure 6.5 Patient Details

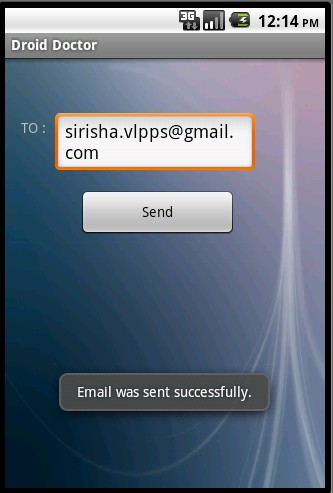


Figure 6.6 Sending of mail

Figure 6.7 Inbox

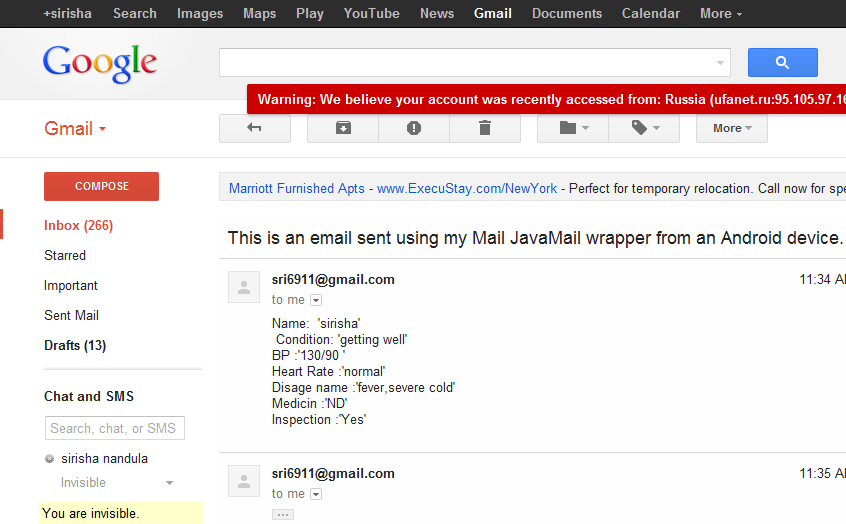




Figure 6.8 Registration Form

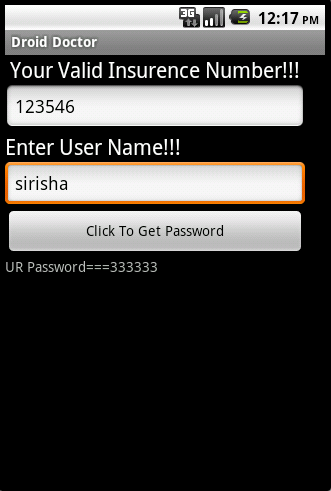


Figure 6.9 Password Retrieval

The applications prior to the development of the project have been observed to be having some drawbacks. They were found to have physical presence of the patient near the doctor in order to get prescription. Hence, an application was developed using various modules. The above screen shots have been taken from the application developed. They show how the application works by entering details into the required fields.

We can see that the patient can view the medicine suggested by the doctor. The patient’s details will be entered by the nurse, which include the BP rate, heart beat rate, the disease which the patient suffers from and the patient’s condition as on that time. The nurse updates all these details and thus, they will be visible to the doctor. The patient can thus get a medicine suggested by the doctor, when he/she logs into the application and views the medicine prescribed by the doctor.

The mails are also sent to the doctor about the condition of the patient. Also, we can see that the users have a facility to enter the login details; also they can utilize the mail service which can be used to retrieve the forgotten password. The mail services can be accessed by the respective users’ personal mail ids. The images of the patients can also be uploaded.

## Chapter 8 CONCLUSION AND FUTURE SCOPE

The applications prior to the development of the project have been observed to be having some drawbacks. They were found to have physical presence of the patient near the doctor in order to get prescription. Hence, an application was developed using various modules.

This application was found to have various useful modules that require operations to be performed by the users depending in the designation. The patient found to have various advantages over the existing system. The module can effectively access data through the Amazon cloud server. The server was found very useful to the application. The users with the android mobile phones can efficiently utilize the application through the various interfaces.

The scope of this project is limited to only the android mobile users. The use of this must be extended to other mediums too. The modules must have extensive operational facilities like a remainder service and alert messages to the patient to warn him/ her about the deadline for updating status. There must a way to access the details through a message sent to mobile phone instead of relying on the mails. These must be implemented in the project for future use.

## BIBLIOGRAPHY

1. Shadi Aljawarneh and Hong Cai “*International Journal of Cloud Applications and Computing*
2. *(IJCAC)*”Published by Information Resources Management Association
3. Herbert Schildt **“JAVA 2-The Complete Reference”** 5th edition
4. David Flangan “**Java is a Nutshell**” 5th edition
5. Jeff Friesen, Geoff Friesen “**Java by Example”**
6. Laura Lemay,Rogers Cadenhead **“Teach yourself Java in 21 days”**
7. [Http://Developer.Android.com/Index.Html](http://Developer.Android.com/Index.Html)
8. [Http://www.Android-Trainer.com/](http://www.Android-Trainer.com/)
9. [Http://Stackoverflow.com/](http://stackoverflow.com/)

<https://github.com/bhanutrinadh/Hospitality-mobile-health-care-using-cloud>