**A Project Report On**

**Medicative Tracking System for Hospital Services**

Submitted in partial fulfillment of the requirements for award of the degree of

**BACHELOR OF TECHNOLOGY**

**In**

**COMPUTER SCIENCE AND ENGINEERING**

**By**

**GUMADAVELLY RAMYA(AP17110010081)**

**Under the guidance of**

**N.S. SHEKAR BABU**

**INCHARGE CED**

**Of**

**ECIL-ECIT**

**ELECTRONICS CORPORATION OF INDIA LIMITED**

**(A Government of India Enterprise)**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**SRM UNIVERSITY, AMARAVATHI, ANDHRA PRADESH**

**ACKNOWLEDGEMENT**

We wish to take this opportunity to express our deep gratitude to all those who helped, encouraged, motivated and have extended their cooperation in various ways during our project work. It is our pleasure to acknowledgement the help of all those individuals who was responsible for foreseeing the successful completion of our project.

We would like to thank **N.S. SHEKAR BABU**(INCHARGE CED) and express our gratitude with great admiration and respect to our project guide **Mr. N.SURESH** for their and help throughout the development of this project by providing us with required information without whose guidance, cooperation and encouragement, this project couldn’t have been materialized

Last but not least, we would like to thank the entire respondents for extending their help in all circumstance

**BY**

**G.RAMYA (AP17110010081)**

**DECLARATION**

We hereby declare that project entitled “**Medicative Tracking System for Hospital Services**”submitted in partial fulfillment of the requirements for the award of degree of Bachelor of Technology in Computer Science and Engineering. This dissertation is our original work and the project has not formed the basis for the award of any degree, associate ship, fellowship or any other similar titles and no part of it has been published or sent for the publication at the time of submission.

**Table of Contents**

**ABSTRACT**

**I. LIST OF FIGURES**

**1. INTRODUCTION**

**1.1 Motivation**

**1.2 Problem Definition**

**1.3 Scope of Project**

**1.4 User Classes and Characteristics**

**2.** LITERATURE SURVEY

**2.1 Introduction**

**2.2 Existing System**

**2.3 Proposed System**

**2.4 Modules**

**3. ANALYSIS**

**3.1 Introduction**

**3.2 Software Requirement Specification**

**3.3 User Interfaces**

**3.4 External Interface Requirements**

**3.4.1 Software Requirements**

**3.4.2 Hardware Requirements**

**3.5 Feasibility Study**

**3.5.1 Economic Feasibility**

**3.5.2 Operational Feasibility**

**3.5.3 Technical Feasibility**

**3.6 Java**

**3.7 Project Perspective**

**3.8 Android**

**3.8.1 About Native code**

**3.9 User Documentation**

**3.9.1 Creating an android project**

**3.9.2 To create an AVD with the AVD manager**

**3.9.3 To Create or Modify a Launch Configuration**

**3.10 SQLite**

**3.10.1 Design**

**3.10.2 Features**

**4. DESIGN**

**4.1 Introduction**

**4.2 Architecture Diagram**

**4.3 Unified Modeling Language (UML)**

**4.4 Building Blocks of UML**

**4.4.1 Things in the UML**

**4.4.2 Relational in the UML**

**4.4.3 Sequence Diagrams**

**4.5 UML diagrams**

**5. IMPLEMENTATION**

**5.1 Introduction**

**5.2 Design and Implementation Constraints**

**5.3 Screen Shots**

**6. TESTING AND VALIDATION**

**6.1 Software Testing**

**6.2 Testing Phases**

**6.3 Testing Activities**

**6.3.1 Unit Testing**

**6.3.2 Equivalence Testing**

**6.3.3 Integration Testing**

* 1. **Test Case Design**

**6.4.1 White-Box Testing**

**6.4.2 Black-Box Testing**

**7. CONCLUSIONS**

**8. REFERENCES**

**ABSTRACT**

**Medicative Tracking System for Hospital Services**

Introduction: Medicative Tracking System for Hospital Services

System help patients to register online for the doctor appointment and it also help doctor to view the patients registered for appointment at particular time. It also helps both patients and doctors to know the information of each other without visiting hospital. This application enables patients to give feedback to doctors and doctors can reply to their patients.

# 1. INTRODUCTION

**1.1. MOTIVATION**

It is difficult for both patients and doctors to know about them without visiting hospital and this requires lot of time. So, this application helps to save time.

**1.2. PROBLEM DEFINITION**

**It is very important to maintain efficient software to handle information of a Doctors, Patients and employee and save their time. This application provides a way to record this information and to access these in a simple way.**

**1.3. SCOPE OF PROJECT**

The different areas where we can use this application are:

* The system will be used in any Hospital ,Clinic , Pathology labs in any Hospital clinic .Pathology labs to get the information from the patients and then storing the data for future usage
* The current system in use is a offline based system ,it is too slow and can’t provide updated lists of patients with in a recommended timespan.

**1.4.USER CLASSES AND CHARACTERISTICS**

The system provides different types of services based on the type of users[Doctor/patient/Employee]. The member can be either a doctor or patient or employee of the Hospital who will be accessing the app online.

1. **LITERATURE SURVEY**

**2.1 INTRODUCTION**

In literature survey we look into the details about the existing system and we try to reduce the disadvantages of the existing system. We try to improve the performance and the efficiency of the new system.

**2.2 EXISTING SYSTEM**

**The Existing systems provide the basic functionalities needed to be handled in a hospital management environment.**

**There is no intelligence of the software in such cases in the existing system all the patient details ,doctor availability details, employee details and regarding the tests done to the patients prescribed by the doctor is maintained manually by the receptionist**

**2.3 PROPOSED SYSTEM**

In our proposed system we are going to provide solutions to all the above mentioned problems by automating the whole hospital management system by using and integrated software that handles the whole system.

The proposed provides one integrated view to patients for Billing , Discharge details, patient medical history, doctors availability time and employee working timings .

**2.4 MODULES**

**Module Description:**

This application consists of following modules

1. Registration Module
2. Employee Module
3. Doctor Module
4. Patient Module

Module I: Registration module consists of sub modules namely:

* + Maintains New Patient Registration, New doctor Registration, new Employee Registration.

Module II: Employee Module helps to view to the profile and timings of employee.

Module III: Doctor Modules helps to doctor to fix appointment time and view patients and reply for patients feedback.

Module IV: Patient module consists of sub modules to register for appointment, give feedback.

3. ANALYSIS

INTRODUCTION

In this phase the requirements are gathered and analyzed. Usersrequirements are gathered in this phase. This phase is the main focus of the administrators and registered accounts. Meetings with users and registered people are held in order to determine the requirements like: Who is going to use the system? How will they use the system?  What data should be input into the system?  What data should be output by the system?  These are general questions that get answered during a requirements gathering phase. After requirement gathering these requirements are analyzed for their validity and the possibility of incorporating the requirements in the system to be development is also studied.

Finally, a Requirement Specification document is created which serves the purpose of guideline for the next phase of the model.

* 1. **SOFTWARE REQUIREMENT SPECIFICIFICATION**

The software requirements specification specifies the functional requirements and nonfunctional requirements. Functional requirements refers to how the system is going to react according to the input provided and how it is going to behave in particular situations and non functional requirements refers to Usability, Reliability, Availability, Performance, Security, Supportability, Interface.

## 3.3 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.

# 3.4 EXTERNAL INTERFACE REQUIREMENTS:

**3.4.1 SOFTWARE REQUIREMENTS**

* Android SDK
* Eclipse Ganymede IDE
* Operating System can be either of these- Windows XP, Windows 7, Windows 8 with different versions, Windows 10, etc.

**3.4.2 HARDWARE REQUIREMENTS**

1. PROCESSOR : (min) P4 processor

2. RAM : 1GB

3. Hard Disk : 20 GB

### 3.5 FEASIBILITYSTUDY

An important outcome of the preliminary investigation is the determination that system requested is feasible. This is to identify the objectives of a new system. Before solving a problem one must know what the problem is. The study is carried out by a small group of people who are familiar with system analysis and design process. Fact finding techniques are used to gather the required information.

The three major areas consider while determining the feasibility of the project are

1. Economic Feasibility
2. Operational Feasibility
3. Technical Feasibility

**3.5.1 ECONOMIC FEASIBILITY**

Economic feasibility attempts to weigh the costs of developing and implementing a new system, against the benefits that would accrue from having the new system in place. This feasibility study gives the top management the economic justification for the new system.

A simple economic analysis which gives the actual comparison of costs and benefits are much more meaningful in this case. These could include increased customer satisfaction, improvement in product quality better decision making timeliness of information, expediting activities, improved accuracy of operations, better documentation and record keeping, faster retrieval of information, better employee morale.

**3.5.2 OPERATIONAL FEASIBILITY**

Proposed projects are beneficial only if they can be turned into information systems that will meet the organizations operating requirements. Simply stated, this test of feasibility asks if the system will work when it is developed and installed. Are there major barriers to Implementation? Here are questions that will help test the operational feasibility of a project:

Is there sufficient support for the project from management from users? If the current system is well liked and used to the extent that persons will not be able to see reasons for change, there may be resistance.

Are the current business methods acceptable to the user? If they are not, Users may welcome a change that will bring about a more operational and useful systems.

Have the user been involved in the planning and development of the project?

Early involvement reduces the chances of resistance to the system and in general and increases the likelihood of successful project. Since the proposed system was to help reduce the hardships encountered. In the existing manual system, the new system was considered to be operational feasible.

**3.5.3TECHNICAL FEASIBILITY**

Evaluating the technical feasibility is the trickiest part of a feasibility study. This is because, at this point in time, not too many-detailed design of the system, making it difficult to access issues like performance, costs on (on account of the kind of technology to be deployed) etc.

A number of issues have to be considered while doing a technical analysis.

i) Understand the different technologies involved in the proposed system:

Before commencing the project, we have to be very clear about what are the technologies that are to be required for the development of the new system.

ii) Find out whether the organization currently possesses the required technologies:

Is the required technology available with the organization?

If so is the capacity sufficient?

For instance -

“Will the current printer be able to handle the new reports and forms required for the new system?”

**3.6 JAVA**

Java is an object-oriented language, and is very similar to C++. Java is simplified to eliminate language features that cause common programming errors. Java source code files are compiled into a format called bytecode, which can then be executed by a Java interpreter.Features being

1. **PLATFORM INDEPENDENT**

The programs written on one platform can run on any platform provided the platform must have the JVM.

1. **PORTABLE**

The feature Write-once-run-anywhere makes the java language portable provided that the system must have interpreter for the JVM.

1. **SIMPLE**

Programs are easy to write and debug because java does not use the pointers explicitly. It also has the automatic memory allocation and deallocation system.

1. **MULTITHREADED**

Multithreading means a single program having different threads executing independently at the same time.

1. **ROBUST**

Java has the strong memory allocation and automatic garbage collection mechanism. It provides the powerful exception handling and type checking mechanism as compare to other programming languages.

1. **OBJECT ORIENTED**

To be an Object Oriented language, any language must follow at least the four characteristics.

* Inheritance
* Encapsulation
* Polymorphism
* Dynamic binding

1. **DISTRIBUTED**

The widely used protocols like HTTP and FTP are developed in java. Internet programmers can call functions on these protocols and can get access to the files from any remote machine on the internet rather than writing codes on their local system.

1. **SECURE**

All the programs in java run under an area known as the sand box. Security manager determines the accessibility options of a class like reading and writing a file to the local disk.

1. **HIGH PERFORMANCE**

In the beginning interpretation of bytecode resulted in slow performance but the advance version of JVM uses the adaptive and just in time compilation technique that improves the performance.

1. **INTEGRATED**

Java is an interpreted language as well. Programs run directly from the source code. 

## 3.7.PROJECT PERSPECTIVE

Android is a software stack for mobile devices that includes an operating system, middleware and key applications. The [Android SDK](http://developer.android.com/sdk/index.html) provides the tools and APIs necessary to begin developing applications on the Android platform using the Java programming language.

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. It would be more cost effective if we can use normal phone numbers for receiving data from customers via SMS. However it is a very tedious process if we don’t have backend automated system to analyze the received data. We are proposing the below approach to make this process cost effective as well as efficient.

**3.8. ANDROID**

Android is a software stack for mobile devices that includes an operating system, middleware and key applications. The android SDK provides the tools and APIs necessary to begin developing applications on the Android platform using the Java programming language.

The Android SDK includes a comprehensive set of development tools. These include a debugger, libraries, a handset emulator (based on QEMU), documentation, sample code, and tutorials. Currently supported development platforms include x86-architecture computers running Linux (any modern desktop Linux distribution), Mac OS X 10.4.8 or later, Windows XP or Vista. The officially supported integrated development environment (IDE) is Eclipse (3.2 or later) using the Android Development Tools (ADT) Plugin, 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.

**3.8.1 ABOUT NATIVE CODE**:

Libraries written in C and other languages can be compiled to ARM native code and installed, but the Native Development Kit is not yet officially supported by Google. Native classes can be called from Java code running under the Dalvik VM using the System.loadLibrary call, which is part of the standard Android Java classes.

## 3.9.USER DOCUMENTATION

In this 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.

**3.9.1. 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:

* Select File > New > Project.
* Select Android > Android Project, and click Next.
* 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.
* Under Properties, fill in all necessary fields:

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

1. 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.
2. Select Create Activity (optional, of course, but common) and enter a name for your main Activity class.
3. Enter a minimum SDK Version. This is an integer that indicates the minimum API Level required to properly run your application. Entering this here automatically sets the minimum SDK Version attribute in the [<uses-sdk>](http://developer.android.com/guide/topics/manifest/uses-sdk-element.html) 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.
4. Click Finish.

**3.9.2.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.

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).

**3.9.3. 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 ,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.

**3.10. SQLITE**

**SQLite** is an ACID-compliant [embedded](http://en.wikipedia.org/wiki/Embedded_database)[relational database management system](http://en.wikipedia.org/wiki/Relational_database_management_system) contained in a relatively small C programming library. The source code for SQLite is in the public domain.

**3.10.1. DESIGN**

Unlike client-server database management systems, the SQLite engine is not a standalone process with which the application program communicates. Instead, the SQLite [library](http://en.wikipedia.org/wiki/Library_%28computing%29) is linked in and thus becomes an integral part of the application program. The library can also be called dynamically. The application program uses SQLite's functionality through simple sms, which reduces [latency](http://en.wikipedia.org/wiki/Latency_%28engineering%29) in database access as function sms within a single process are more efficient than [inter-process communication](http://en.wikipedia.org/wiki/Inter-process_communication).

**3.10.2. FEATURES**

SQLite implements most of the [SQL-92](http://en.wikipedia.org/wiki/SQL-92) standard for [SQL](http://en.wikipedia.org/wiki/SQL) but it lacks some features. A standalone program called sqlite3 is provided which can be used to create a database, define tables within it, insert and change rows, run queries and manage a SQLite database file. SQLite is a popular choice for local/client SQL storage within a [web browser](http://en.wikipedia.org/wiki/Web_browser) and within a [rich internet application framework](http://en.wikipedia.org/wiki/List_of_rich_internet_application_frameworks). This may be because SQLite's dynamically typed storage matches the [web browser](http://en.wikipedia.org/wiki/Web_browser)'s core languages of [JavaScript](http://en.wikipedia.org/wiki/Javascript) and [XML](http://en.wikipedia.org/wiki/XML). SQLite uses an unusual [type system](http://en.wikipedia.org/wiki/Type_system) for an SQL-compatible DBMS. Instead of assigning a type to a column as in most SQL database systems, types are assigned to individual values; in language terms it is dynamically typed.

**4.DESIGN**

**4.1. INTRODUCTION**

Software design is the process by which an agent creates a specification of a software artifact, intended to accomplish goals, using a set of primitive components and subject to constraints. Software design may refer to either "all the activity involved in conceptualizing, framing, implementing, commissioning, and ultimately modifying complex systems" or "the activity following requirements specification and before programming, as in a stylized software engineering process." Software design usually involves problem solving and planning a software solution. This includes both a low-level component design and a high-level, architecture design.

**4.2. ARCHITECTURE DIAGRAM**

Architecture diagram is a[diagram](http://en.wikipedia.org/wiki/Diagram) of a [system](http://en.wikipedia.org/wiki/System), in which the principal parts or functions are represented by blocks connected by lines that show the relationships of the blocks.The block diagram is typically used for a higher level, less detailed description aimed more at understanding the overall concepts and less at understanding the details of implementation.

SMS

User

AndroidSDK

SQLite

Java program

Get

CSV file

**Fig 4.2 Architecture**

A SMS user for who the application looks like an user interface actually consists of a database called as SQLite that comes along with Android SDK and need no other installation. This is the database that is used to store and retrieve information. This is an application that is developed in java and hence all its features apply here as well such as platform independence, data hiding,

**4.3.UNIFIED MODELING LANGUAGE (UML)**

The unified modeling is a standard language for specifying, visualizing, constructing and documenting the system and its components is a graphical language which provides a vocabulary and set of semantics and rules. The UML focuses on the conceptual and physical representation of the system. It captures the decisions and understandings about systems that must be constructed. It is used to understand, design, configure and control information about the systems.

Depending on the development culture, some of these artifacts are treated more or less formally than others. Such artifacts are not only the deliverables of a project; they are also critical in controlling, measuring, and communicating about a system during its development and after its deployment.

The UML addresses the documentation of a system's architecture and all of its details. The UML also provides a language for expressing requirements and for tests. Finally, the UML provides a language for modeling the activities of project planning and release management.

**4.4. BUILINDING BLOCKS OF UML**

The vocabulary of the UML encompasses three kinds of building blocks:

1. Things
2. Relationships
3. Diagrams

Things are the abstractions that are first-class citizens in a model, relationships tie these things together and diagrams group interesting collections of things.

**4.4.1. THINGS IN THE UML**

There are four kinds of things in the UML:

1. Structural things
2. Behavioral things
3. Grouping things
4. Annotational things

**1.STUCTURALTHINGS:** are the nouns of UML models. The structural things used in the project design are:

First, a **class** is a description of a set of objects that share the same attributes, operations, relationships and semantics.

|  |
| --- |
| Window |
| Origin  Size |
| open()  close()  move()  display() |

##### **Fig: Classes**

Second, a **use case** is a description of set of sequence of actions that a system performs that yields an observable result of value to particular actor.



**Fig: Use Cases**

Third, a node is a physical element that exists at runtime and represents a computational resource, generally having at least some memory and often processing capability.

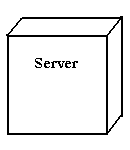


Fig: Nodes

**2.BEHAVIORAL THINGS:**are the dynamic parts of UML models. The behavioral thing used is:

**INTERACTION:**

An interaction is a behavior that comprises a set of messages exchanged among a set of objects within a particular context to accomplish a specific purpose. An interaction involves a number of other elements, including messages, action sequences (the behavior invoked by a message, and links (the connection between objects).



**Fig: Messages**

**4.4.2. RELATIONAL IN THE UML**

There are four kinds of relationships in the UML:

* Dependency
* Association
* Generalization
* Realization

A **dependency** is a semantic relationship between two things in which a change to one thing may affect the semantics of the other thing (the dependent thing).



**Fig: Dependencies**

An **association** is a structural relationship that describes a set links, a link being a connection among objects. Aggregation is a special kind of association, representing a structural relationship between a whole and its parts.



**Fig: Association**

A **generalization** is a specialization/ generalization relationship in which objects of the specialized element (the child) are substitutable for objects of the generalized element(the parent).



**Fig: Generalization**

A **realization** is a semantic relationship between classifiers, where in one classifier specifies a contract that another classifier guarantees to carry out.



Fig: Realization

**4.4.3.SEQUENCE DIAGRAMS**

UML sequence diagrams are used to represent the flow of messages, events and actions between the objects or components of a system. Time is represented in the vertical direction showing the sequence of interactions of the header elements, which are displayed horizontally at the top of the diagram.

Sequence Diagrams are used primarily to design, document and validate the architecture, interfaces and logic of the system by describing the sequence of actions that need to be performed to complete a task or scenario. UML sequence diagrams are useful design tools because they provide a dynamic view of the system behavior which can be difficult to extract from static diagrams or specifications.

**ACTOR**

Represents an external person or entity that interacts with the system

**Sequence diagram actor element**

**OBJECT**

Represents an object in the system or one of its components

**Sequence diagram object element**

**UNIT**

Represents a subsystem, component, unit, or other logical entity in the system (may or may not be implemented by objects)

**Sequence diagram unit element**

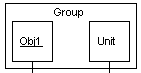
**SEPERATOR**

Represents an interface or boundary between subsystems, components or units (e.g., air interface, Internet, network)

**Sequence diagram separator element**

**GROUP**

Groups related header elements into subsystems or components

****

## 4.4.3.1.SEQUENCE DIAGRAM BODY ELEMENTS

**ACTION**

Represents an action taken by an actor, object or unit

**Sequence diagram action element**

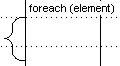
**ASYNCHRONUS MESSAGE**

An asynchronous message between header elements

**Sequence diagram asynchronous message element**

**BLOCK**

A block representing a loop or conditional for a particular header element

****

**CALL MESSAGE**

A call (procedure) message between header elements

**Sequence diagram call message element**

**CREATE MESSAGE**

A "create" message that creates a header element (represented by lifeline going from dashed to solid pattern)

**Sequence diagram create message element**

**DIAGRAM LINK**

Represents a portion of a diagram treated as a functional block. Similar to a procedure or function call that abstracts functionality or details not shown at this level and can be an optional link to another diagram for elaboration.

**Sequence diagram diagram link element**

Else Block Represents an "else" block portion of a diagram block

**Sequence diagram else block element**

**MESSAGE**

A simple message between header elements

**Sequence diagram message element**

**RETURN MESSAGE**

A return message between header elements

**Sequence diagram return message element**

**4.5.UML DIAGRAMS FOR ONLINE LIBRARY SYSTEM APP**

**USE CASE DIAGRAM**

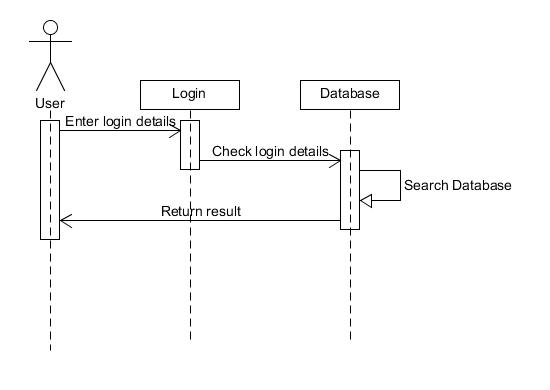
patient

doctor

**Use Case Diagram**

Employee

**SEQUENCE DIAGRAM:**



**Sequence Diagram**

**5.IMPLEMENTATION**

**5.1.INTRODUCTION**

Implementation is the most crucial stage in achieving a successful system and giving the user’s confidence that the new system is workable and effective. Implementation of the modified application to replace an existing one. This type of conversation is relatively easy to handle, provide there are no major changes in the system.

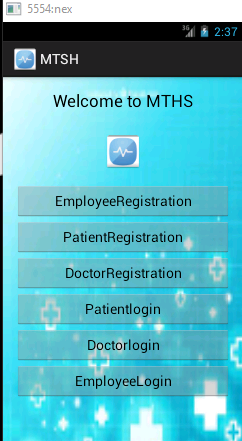
Each program is tested individually at the time of development using the data and has verified that this program linked together in the way specified in the programs specification, the computer system and its environment is tested to the satisfaction of the user. The system that has been developed is accepted and proved to be satisfactory for the user. And so the system is going to be implemented very soon. A simple operating procedure is included so that the user can understand the different functions clearly and quickly.

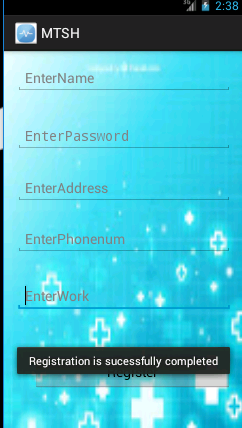
Initially as a first step the executable form of the application is to be created and loaded in the common server machine which is accessible to all the user and the server is to be connected to a network. The final stage is to document the entire system which provides components and the operating procedures of the system.

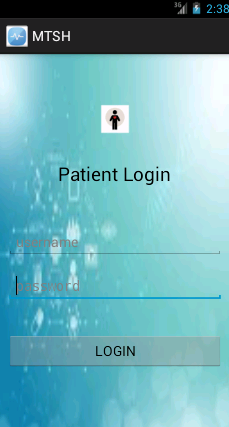
## 5.2. DESIGN AND IMPLEMENTATION CONSTRAINTS

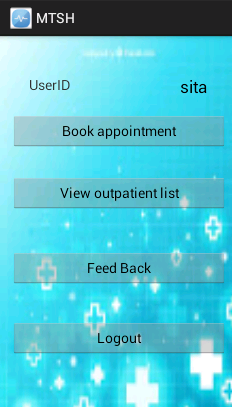
All modules are coded thoroughly based on requirements from software organization. The software is designed in such a way that the user can easily interact with the screen. Software is designed in such a way that it can be extended to the real time business*.*

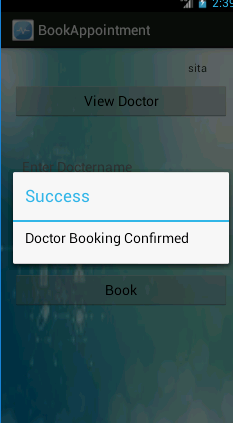
**5.3. SCREEN SHOTS**



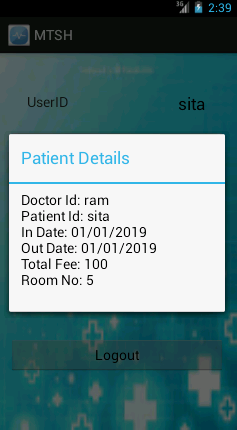


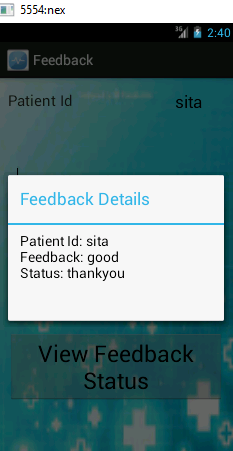


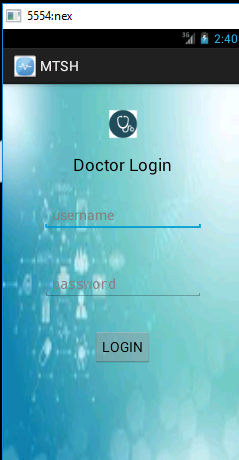


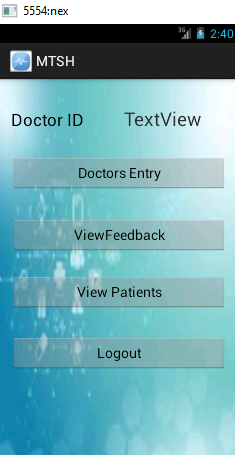


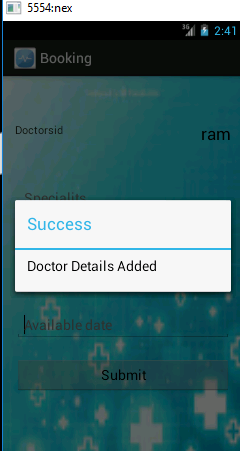


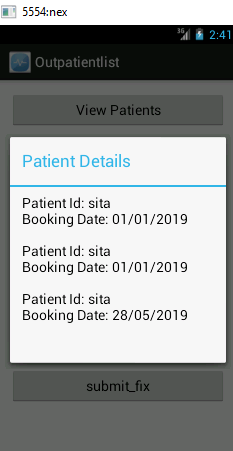


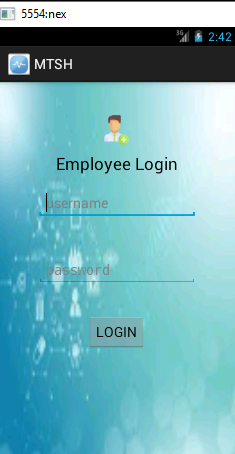


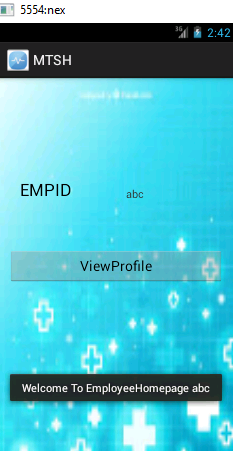


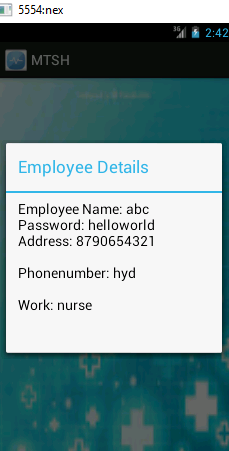












**6. TESTING**

## 6.1.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.

## 6.2.TESTING PHASES

Software testing includes the following:

1. Test activities are determined and test data selected

2. The test is conducted and test results are compared with the expected results.

## 6.3.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.

## 6.3.1.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?

## 6.3.2.EQUIALENCE 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.

1. **DISJOINTEDNESS**:

No input belongs to one of the equivalent classes.

1. **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.

## 6.3.2.1.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.

## 6.3.2.2.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.

**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.

1. **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.

## 6.3.2.3.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 technique which focuses 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.

## 6.3.3.INTEGRATION TESTING

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

## 6.4.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 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

## 6.4.1. WHITE-BOX TESTING

White –box testing, sometimes called glass-box testing is a test, case designed method that uses the control structure of the procedural design to derive test cases. Using white-box testing methods, the s/w engineer can derive test cases that guarantee that all independent paths within a module have been exercised at least once. Exercise all logical decisions on their true and false sides. Execute all loops at their boundaries and within their operational bounds. Exercise internal data structures to ensure their validity.

Basis path testing is a white-box testing technique. The basis path method enables the test case designer to derive a logical complexity measure of a procedural design and use this measure as a guide for defining a basis set are guaranteed to exercise every statement in the program at least one time during testing.

## 6.4.2. BLACK-BOX TESTING

Black-box testing ,also called behavioral testing, focuses on the functional requirements of the s/w. Black-box testing enables the software engineer to derive sets of input conditions that will fully exercise all functional requirements of a program. It is a complementary approach that is likely to uncover a different class of errors that white-box methods could not.

Black-box testing attempted to find errors in the following categories.

* + - Incorrect or missing functions.
    - Interface errors.
    - Errors in data structures or external data base access.
    - Behavior or performance errors.
    - Initialization and termination errors.

Black-box testing purposely disregards control structure; attention is focused on information domain. By applying black-box techniques, we derive a set of cases that satisfies the criteria test cases that reduce, by a count that is greater than one, the number of additional test cases that must be designed to achieve reasonable testing. Test cases that tell us something about the presence or absence of classes of errors, rather than an error associated only with the specified.

**7. CONCLUSION**

At the end of this application it is a sophisticated approach for users to have a best selection and gives better performance for Administrator so that he can easily add, update and view packages details. This initiative of making travel plan in online made more enjoyable and easier than the real world of making a travel plan. This is convenient to the customers to view there trip details. There is also a facility for the customer to check the status of there selected package.

## 8. REFERENCES

1. Herbert Schildt.2008 ,”Java Complete Reference”, Tata McGraw-Hill ,

7th Edition, pp. 177-180.

1. GradyBrooch,James Rambaugh.1998,“UnifiedModeling Language User Guide”, Addison Wesley Publishing, chapter 8-31.

**WEBSITES:**  **REFERRED URLS:**

www.android.com http://developer.android.com/index.html

www.google.com http://en.wikipedia.org/wiki/SQLite