**1. INTRODUCTION**

**1.1 PROBLEM DEFINITION**

One of thefastestgrowing industries now a day 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 many manufactured phones. These phones are described as next Generation mobiles [As described by Google]. In real life, forgetting the things is very common human tendency. sometimes we may kept the things in our premises and we may forget about it.Same thing can be happened with our mobiles. We may forget/lose our phone in our near premises. We may think that, we can just make call to that phone and find the location.but if the phone is in silent mode what we can do? how can we find the mobile location?

In that situation this project is very helpful. In this project, just by sending a single message to our phone we are changing our phone mode to ringing or silent or vibrate whatever we want.

. This application supports 2.3 version of android frame work.

**1.2 EXISTING SYSTEM:**

# In the existing system there is no service like changing our phone upon receiving security code as a message previously. If any user unexpectedly forget his phone in his near premises and its in silent mode. As of now in android mobiles there is no such system which changes our phone mode to whatever we want i.e. silent or ring or vibrate. ISf user wants to change his mode, only manually he can change the mode of the phone which is not possible when phone is missing.

**1.3 PROPOSED SYSTEM:**

To overcome the drawback in the exsisting System we are going to develop this project.

Advantages:

* Giving the security codes for each mode(silent, Ring, Vibrate)
* Using that security code, changing our phone mode to silent, ring or vibrate.

***1.3.1 Modules***

Ring Mode on android is having the following modules

1. Setting the Security Code
2. Changing the phone mode

1.Setting the Security Code:

In this module, we need to set security code for each mode i.e. silent, ring and vibrate. By default values will be silent for silent mode, ring for Ring mode and vibrate for vibrate mode. User can also change this values.

2.Changing the phone mode:

Whatever the codes you have given,just send that code as amessage to your phone.then your phone will be changed to that particular mode what u have sent.

* 1. **SCOPE OF PROJECT**

One of the fastest growing industries 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]. Developing application for such mobile phones using the open source android SDK is quite interesting. This makes the application call history quite easy, efficient, flexible and economic.

**2.LITERATURE SURVEY**

**2.1. 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 byte code, 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 deallaocation 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 are 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 byte code 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.

**2.2 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.

## Features

**Application framework** enabling reuse and replacement of components

**Dalvik virtual machine** optimized for mobile devices

**Integrated browser** based on the open source [WebKit](http://webkit.org/) 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

**Android application**

Developers have full access to the same framework APIs used by the core applications. The application architecture is designed to simplify the reuse of components; any application can publish its capabilities and any other application may then make use of those capabilities (subject to security constraints enforced by the framework). This same mechanism allows components to be replaced by the user.

## 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](http://developer.android.com/sdk/eclipse-adt.html#installing).

## 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.

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) 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 application

**2.2.1 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. Load Library call, which is part of the standard Android Java classes.

**2.2.2 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 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 min 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.

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](http://developer.android.com/guide/topics/resources/index.html).

res/

A folder for your application resources, such as drawable files, layout files, string values, etc. See [Resources and Assets](http://developer.android.com/guide/topics/resources/index.html).

AndroidManifest.xml

The Android Manifest for your project. See [The AndroidManifest.xml File](http://developer.android.com/guide/topics/manifest/manifest-intro.html).

default.properties

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".

**2.2.3 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).

**2.2.4 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.

**2.3. 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.

***2.3.1 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 [function calls](http://en.wikipedia.org/wiki/Subroutine), which reduces [latency](http://en.wikipedia.org/wiki/Latency_%28engineering%29) in database access as function calls within a single process are more efficient than [inter-process communication](http://en.wikipedia.org/wiki/Inter-process_communication).

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

**3. SYSTEM ANALYSIS**

**3.1 SOFTWARE REQUIREMENTS**

* Java
* Android SDK 1.5 or later
* Eclipse Ganymede IDE
* Operating System can be Windows XP, LINUX and Mac etc.

**3.2 HARDWARE REQUIREMENTS**

1. PROCESSOR : (min) P2 Processor

2. RAM : (min) 128

3. HARD DISK : 100 MB

**3.3 FEASIBILITY STUDY**

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

***3.3.1 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.

* + 1. ***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.

***3.3.3******Technical 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?”

**4.SYSTEM DESIGN**

**4.1 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.

call history

User

android sdk

sqlite

java program

get call logs

Fig 4.1 Architecture of Call History

A Call History 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, portable etc.

**4.2 DATA FLOW DIAGRAMS**

A data-flow diagram (DFD) is a graphical representation of the "flow" of data through an [information system](http://en.wikipedia.org/wiki/Information_system). DFDs can also be used for the [visualization](http://en.wikipedia.org/wiki/Data_visualization) of [data processing](http://en.wikipedia.org/wiki/Data_processing) (structured design).

On a DFD, data items flow from an external data source or an internal data store to an internal data store or an external data sink, via an internal process. The idea behind the explosion of a process into more process is that understanding at one level of detail is exploded into greater detail at the next level. This is done further explosion is necessary and an adequate amount of detail is described for analyst to understand the process.

`A DFD is also known as a “bubble chart” has the purpose of clarifying system requirements and identifying major transformations that will become programs in system design.

So it is the starting point of the design to the lowest level of detail. A DFD consists of a series of bubbles joined by data flows in the system.

**4.2.1 DFD Symbols**

In the DFD, there are four symbols

1. A Square defines a source or destination of system data.
2. An arrow identifies data flow. It is the pipeline through which the information flows.
3. A circle or a bubble represents a process that transforms incoming data flow into outgoing data flows.
4. An open rectangle is a data store ,data at rest or a temporary repository of data

Process that transforms data flow

Source or destination of data

Data flow

Data store

*Fig 4.2 DFD Symbols*

**4.3 UNIFIED MODELING LANGUAGE (UML)**

UML Concepts

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

* Visualizing
* Specifying
* Constructing
* Documenting the artifacts 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 modeling language is a language whose vocabulary and the rules focus on the conceptual and physical representation of a system. Modeling yields an understanding of a system.

Building Blocks of the UML:

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

* Things
* Relationships
* Diagrams

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

Things in the UML:

There are four kinds of things in the UML:

* Structural things
* Behavioral things
* Grouping things
* Annotational things

**Structural things** 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() |

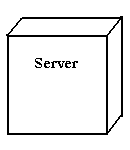
##### *Fig 4.3 Classes*

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 4.4: 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 4.5 Nodes*

**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 4.6 Messages*

**6.1.3 Relationships 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 4.7 Dependencies*

An associationis 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 4.8 Association*

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



*Fig 4.9 Generalization*

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



*Fig 4.10 Realization*

**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**

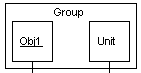
**Separator**

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

****

## Sequence Diagram Body Elements

**Action**

Represents an action taken by an actor, object or unit

**Sequence diagram action element**

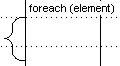
**Asynchronous Message**

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 being treated as a functional block. Similar to a procedure or function call that abstracts functionality or details not shown at this level. Can optionally be linked 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**



*Fig 4.11 Sequence diagram for setting security codes.*

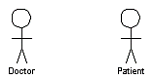


*Fig 4.12 Sequence diagram for changing mode.*

**Use Case Diagram**

A use case diagram is a graph of actors set of use cases enclosed by a system boundary, communication associations between the actors and users and generalization among use cases. The use case model defines the outside(actors) and inside(use case) of the system’s behavior.

Use case diagram is quite simple in nature and depicts two types of elements: one representing the business roles and the other representing the business processes.



*Fig 4.13 an actor in a use case diagram*

To identify an actor, search in the problem statement for business terms that portray roles in the system. For example, in the statement "patients visit the doctor in the clinic for medical tests," "doctor" and "patients" are the business roles and can be easily identified as actors in the system.

**Use case:**

A use case in a use case diagram is a visual representation of a distinct business functionality in a system. The key term here is "distinct business functionality." To choose a business process as a likely candidate for modeling as a use case, you need to ensure that the business process is discrete in nature.

As the first step in identifying use cases, you should list the discrete business functions in your problem statement. Each of these business functions can be classified as a potential use case. Remember that identifying use cases is a discovery rather than a creation. As business functionality becomes clearer, the underlying use cases become more easily evident. A use case is shown as an ellipse in a use case diagram (see Figure 3.2).

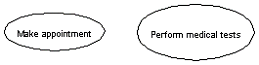


Fig 4.14:use cases in a use case diagram

Figure 4.13 shows two uses cases: "Make appointment" and "Perform medical tests" in the use case diagram of a clinic system. As another example, consider that a business process such as "manage patient records" can in turn have sub-processes like "manage patient's personal information" and "manage patient's medical information." Discovering such implicit use cases is possible only with a thorough understanding of all the business processes of the system through discussions with potential users of the system and relevant domain knowledge.

**USE CASE DIAGRAM**



*Fig 4.15 Use case diagram*

**Activity Diagram**

Activity diagrams represent the business and operational workflows of a system. An Activity diagram is a dynamic diagram that shows the activity and the event that causes the object to be in the particular state.

So, what is the importance of an Activity diagram, as opposed to a State diagram? A State diagram shows the different states an object is in during the lifecycle of its existence in the system, and the transitions in the states of the objects. These transitions depict the activities causing these transitions, shown by arrows.

An Activity diagram talks more about these transitions and activities causing the changes in the object states.



*Fig 4.16 Activity diagram*

#### Defining an Activity diagram

Let us take a look at the building blocks of an Activity diagram.

#### Elements of an Activity diagram

An Activity diagram consists of the following behavioral elements:

**Initial Activity:**

This shows the starting point or first activity of the flow. Denoted by a solid circle. This is similar to the notation used for Initial State.

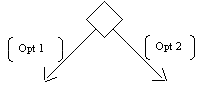
http://www.developer.com/img/articles/2003/08/11/UML07T1.gif

**Activity:** Represented by a rectangle with rounded (almost oval) edges.

.http://www.developer.com/img/articles/2003/08/11/UML07T2.gif

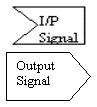
**Decisions:**

Similar to flowcharts, a logic where a decision is to be made is depicted by a diamond, with the options written on either sides of the arrows emerging from the diamond, within box brackets.



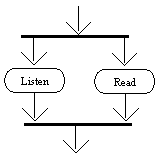
**Signal:**

When an activity sends or receives a message, that activity is called a signal. Signals are of two types: Input signal (Message receiving activity) shown by a concave polygon and Output signal (Message sending activity) shown by a convex polygon.



**Concurrent Activities:**

Some activities occur simultaneously or in parallel. Such activities are called concurrent activities. For example, listening to the lecturer and looking at the blackboard is a parallel activity. This is represented by a horizontal split (thick dark line) and the two concurrent activities next to each other, and the horizontal line again to show the end of the parallel activity.



**Final Activity:**

The end of the Activity diagram is shown by a bull's eye symbol, also called as a final activity.

http://www.developer.com/img/articles/2003/08/11/UML07T6.gif

**Class Diagram**

An object is any person, place, thing, concept, event, screen, or report applicable to your system. Objects both know things (they have attributes) and they do things (they have methods).

A class is a representation of an object and, in many ways, it is simply a template from which objects are created. Classes form the main building blocks of an object-oriented application.  Although thousands of students attend the university, you would only model one class, called *Student*, which would represent the entire collection of students.

### Responsibilities

Classes are typically modeled as rectangles with three sections: the top section for the name of the class, the middle section for the attributes of the class, and the bottom section for the methods of the class. Attributes are the information stored about an object, while methods are the things an object or class do. For example, students have student numbers, names, addresses, and phone numbers. Those are all examples of the attributes of a student. Students also enroll in courses, drop courses, and request transcripts. Those are all examples of the things a student does, which get implemented (coded) as methods. You should think of methods as the object-oriented equivalent of functions and procedures.



*Fig 4.17 Class Diagram*

**object diagram**

An object diagram in the Unified Modeling Language (UML), is a diagram that shows a complete or partial view of the structure of a modeled system at a specific time.

An Object diagram focus on some particular set of object instances and attributes, and the links between the instances. A correlated set of object diagrams provides insight into how an arbitrary view of a system is expected to evolve over time. Object diagrams are more concrete than class diagrams, and are often used to provide examples, or act as test cases for the class diagrams. Only those aspects of a model that are of current interest need be shown on an object diagram.



## *Fig 4.18 Object diagram*

**State chart diagram**

State chart diagram is used to describe the states of different objects in its life cycle. So the emphasis is given on the state changes upon some internal or external events. These states of objects are important to analyze and implement them accurately.State chart diagrams are very important for describing the states. States can be identified as the condition of objects when a particular event occurs.

Before drawing a State chart diagram we must have clarified the following points:

* Identify important objects to be analyzed.
* Identify the states.
* Identify the events.



*4.19 State chart diagram*

**Deployment Diagram**

Deployment diagrams are used to visualize the topology of the physical components of a system where the software components are deployed.So deployment diagrams are used to describe the static deployment view of a system. Deployment diagrams consist of nodes and their relationships.

The purpose of deployment diagrams can be described as:

The name Deployment itself describes the purpose of the diagram. Deployment diagrams are used for describing the hardware components where software components are deployed.

* Visualize hardware topology of a system.
* Describe the hardware components used to deploy software components.
* Describe runtime processing nodes.



*4.20 Deployment diagram*

**5. SYSTEM CODING & IMPLEMENTATION**

**First Splash.xml**

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<LinearLayout

xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"fill\_parent"*

android:orientation=*"vertical"*

android:background=*"@drawable/finalconfirm1"*

android:id=*"@+id/lll"* >

</LinearLayout>

**SplashUI.java**

package com.mode;

import android.app.Activity;

import android.content.Intent;

import android.os.Bundle;

import android.view.View;

import android.view.ViewGroup;

public class SplashUI extends Activity {

/\*\* Called when the activity is first created. \*/

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.firstsplash);

Thread runnerlog=new Thread()

{

public void run()

{

try

{

int logoTimer=0;

while(logoTimer<2000)

{

sleep(100);

logoTimer=logoTimer+100;

}

//startActivity(new Intent("com.mode.HowTo"));

Intent i=new Intent(SplashUI.this,HowTo.class);

startActivity(i);

}catch (Exception e) {

// TODO: handle exception

e.printStackTrace();

}finally

{

finish();

}

}

};

runnerlog.start();

}

@Override

protected void onDestroy() {

// TODO Auto-generated method stub

super.onDestroy();

unbindDrawables(findViewById(R.id.lll));

System.gc();

}

private void unbindDrawables(View view) {

if (view.getBackground() != null) {

view.getBackground().setCallback(null);

}

if (view instanceof ViewGroup) {

for (int i = 0; i < ((ViewGroup) view).getChildCount(); i++) {

unbindDrawables(((ViewGroup) view).getChildAt(i));

}

((ViewGroup) view).removeAllViews();

}

}

}

Howto.xml

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<ScrollView

xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:id=*"@+id/scrollView1"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"fill\_parent"*

android:background=*"@drawable/thk2"*

>

<LinearLayout

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"fill\_parent"*

android:orientation=*"vertical"*

android:layout\_marginLeft=*"6px"*

android:layout\_marginRight=*"6px"*

>

<TextView android:text=*"How to use this app"*

android:id=*"@+id/tView121"*

android:textColor=*"#ffffff"*

android:textStyle=*"bold"*

android:textSize=*"20px"*

android:layout\_gravity=*"center"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_marginTop=*"6px"*

android:layout\_marginBottom=*"15px"*

></TextView>

<TextView android:text=*"@string/s1"*

android:id=*"@+id/tView1"*

android:textStyle=*"bold"*

android:textSize=*"15px"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_marginLeft=*"10px"*

android:layout\_marginRight=*"10px"*></TextView>

<TextView android:text=*"@string/howtotext1"*

android:id=*"@+id/tView1"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_marginLeft=*"10px"*

android:layout\_marginRight=*"10px"*></TextView>

<View

android:layout\_marginTop=*"6px"*

android:layout\_marginBottom=*"6px"*

android:layout\_marginLeft=*"10px"*

android:layout\_marginRight=*"10px"*

android:background=*"#C2DFFF"*

android:layout\_height=*"1dip"*

android:layout\_width=*"fill\_parent"*

/>

<TextView android:text=*"@string/s3i"*

android:id=*"@+id/tView1"*

android:textStyle=*"bold"*

android:textSize=*"15px"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_marginLeft=*"10px"*

android:layout\_marginRight=*"10px"*></TextView>

<TextView android:text=*"@string/s3id"*

android:id=*"@+id/tView1"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_marginLeft=*"10px"*

android:layout\_marginRight=*"10px"*></TextView>

<Button android:layout\_width=*"fill\_parent"*

android:id=*"@+id/startb"* android:text=*"Start App"*

android:layout\_marginTop=*"10px"*

android:layout\_marginBottom=*"10px"*

android:layout\_height=*"50px"*></Button>

</LinearLayout>

</ScrollView>

HowTo.java

package com.mode;

import android.app.Activity;

import android.content.Intent;

import android.os.Bundle;

import android.view.View;

import android.view.View.OnClickListener;

import android.view.Window;

import android.widget.Button;

public class HowTo extends Activity {

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

//requestWindowFeature(Window.FEATURE\_CUSTOM\_TITLE);

setContentView(R.layout.howto);

Button b=(Button)findViewById(R.id.startb);

b.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

// TODO Auto-generated method stub

Intent it=new Intent(HowTo.this,main.class);

startActivity(it);

}

});

}

}

Main.xml

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<TableLayout android:id=*"@+id/TableLayout01"*

android:background=*"@drawable/bc"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"fill\_parent"*

xmlns:android=*"http://schemas.android.com/apk/res/android"*>

<TableRow android:id=*"@+id/TableRow01"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*>

<TextView android:text=*"SwitchTo Silent"*

android:id=*"@+id/uname"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*>

</TextView>

<EditText android:text=*""*

android:id=*"@+id/silent"*

android:layout\_width=*"180dip"*

android:layout\_height=*"wrap\_content"*

>

</EditText>

</TableRow>

<TableRow android:id=*"@+id/TableRow02"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*>

<TextView android:text=*"SwitchTo Ring"*

android:id=*"@+id/pass"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*>

</TextView>

<EditText android:text=*""*

android:id=*"@+id/ring"*

android:layout\_width=*"180dip"*

android:layout\_height=*"wrap\_content"*>

</EditText>

</TableRow>

<TableRow android:id=*"@+id/TableRow03"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*>

<TextView android:text=*"SwitchTo Vibrate"*

android:id=*"@+id/vibr"*

android:layout\_width=*"180dip"*

android:layout\_height=*"wrap\_content"*>

</TextView>

<EditText android:text=*""*

android:id=*"@+id/vibrate"*

android:layout\_width=*"180dip"*

android:layout\_height=*"wrap\_content"*>

</EditText>

</TableRow>

<TableRow android:id=*"@+id/TableRow06"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*>

<AbsoluteLayout

android:id=*"@+id/AbsoluteLayout01"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*>

<Button android:text=*"Save"*

android:layout\_x=*"120px"*

android:layout\_y=*"30px"*

android:id=*"@+id/Save"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*>

</Button>

</AbsoluteLayout>

</TableRow>

</TableLayout>

**Main.java**

package com.mode;

import java.io.IOException;

import java.util.ArrayList;

import android.app.Activity;

import android.content.Context;

import android.database.SQLException;

import android.database.sqlite.SQLiteDatabase;

import android.os.Bundle;

import android.view.View;

import android.view.View.OnClickListener;

import android.widget.Button;

import android.widget.EditText;

import android.widget.Toast;

public class main extends Activity {

Dbhandler myDbHelper;

SQLiteDatabase Mydatabase;

ArrayList<String> aa;

EditText silent,ring,vibrate;

Button save;

/\*\* Called when the activity is first created. \*/

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

silent=(EditText)findViewById(R.id.silent);

ring=(EditText)findViewById(R.id.ring);

vibrate=(EditText)findViewById(R.id.vibrate);

save=(Button)findViewById(R.id.Save);

this.myDbHelper=new Dbhandler(this);

FetchingData();

Mydatabase=myDbHelper.getReadableDatabase();

aa=this.myDbHelper.getModes(Mydatabase);

System.out.println("values of Db "+aa);

String s1= aa.toString();

s1=s1.substring(1, s1.length()-1);

//s1=s1.replace("", newChar)

String[] s=s1.split(",");

silent.setText(s[0]);

ring.setText(s[1]);

vibrate.setText(s[2]);

save.setOnClickListener(new OnClickListener(){

@Override

public void onClick(View arg0) {

// TODO Auto-generated method stub

String stxt=silent.getText().toString();

String rtxt=ring.getText().toString();

String vtxt=vibrate.getText().toString();

if(stxt.equals("") || rtxt.equals("") || vtxt.equals(""))

{

Toast.makeText(getApplicationContext(), "Please fill all fields", 60).show();

}

else

{

/\*Context context = getApplicationContext();

CharSequence error = "Please enter a track name" +stxt;

int duration = Toast.LENGTH\_LONG;

Toast toast = Toast.makeText(context, error, duration);

toast.show();\*/

myDbHelper=new Dbhandler(main.this);

FetchingData();

Mydatabase=myDbHelper.getReadableDatabase();

Mydatabase.execSQL("update mode set text='"+stxt+"' where Mode='silent'");

System.out.println("^^^^^^^^^^^^^^^ inserted value is "+stxt);

Mydatabase.execSQL("update mode set text='"+rtxt+"' where Mode='ring'");

Mydatabase.execSQL("update mode set text='"+vtxt+"' where Mode='vibrate'");

Toast.makeText(getApplicationContext(), stxt, 70).show();

}

}

});

}

private void FetchingData() {

// TODO Auto-generated method stub

try {

myDbHelper.onCreateDataBase();

} catch (IOException ioe) {

throw new Error("Unable to create database");

}

try {

myDbHelper.openDataBase();

Mydatabase = myDbHelper.getWritableDatabase();

System.out.println("executed");

}catch(SQLException sqle){

throw sqle;

}

// TODO Auto-generated method stub

}

}

Broadcast.java

package com.mode;

import java.io.IOException;

import android.content.BroadcastReceiver;

import android.content.Context;

import android.content.Intent;

import android.database.Cursor;

import android.database.SQLException;

import android.database.sqlite.SQLiteDatabase;

import android.media.AudioManager;

import android.os.Bundle;

import android.telephony.SmsMessage;

import android.util.Log;

import android.widget.Toast;

public class Broadcast extends BroadcastReceiver {

// private ITelephony telephonyService;

private static final String SMS\_RECEIVED = "android.provider.Telephony.SMS\_RECEIVED";

private static final String TAG = "SMSBroadcastReceiver";

Dbhandler myDbHelper;

private AudioManager maudio;

SQLiteDatabase Mydatabase;

SQLiteDatabase db;

Context context = null;

@Override

public void onReceive(Context context, Intent intent) {

// TODO Auto-generated method stub

Log.i(TAG, "Intent recieved: " + intent.getAction());

maudio=(AudioManager)context.getSystemService(context.AUDIO\_SERVICE);

if (intent.getAction().equals(SMS\_RECEIVED)) {

Bundle bundle = intent.getExtras();

if (bundle != null) {

Log.i(TAG, "Message recieved: ");

Object[] pdus = (Object[])bundle.get("pdus");

final SmsMessage[] messages = new SmsMessage[pdus.length];

for (int i = 0; i < pdus.length; i++) {

messages[i] = SmsMessage.createFromPdu((byte[])pdus[i]);

System.out.println("message is.."+messages[i]);

Log.i(TAG, "Message recieved: " + messages[0].getMessageBody());

String mode=messages[0].getMessageBody();

System.out.println("message body is.."+mode);

String m\_mode=getMode(mode,context);

System.out.println("mode is...."+m\_mode);

changemode(m\_mode);

}

if (messages.length > -1) {

Log.i(TAG, "Message recieved: " + messages[0].getMessageBody());

}

}

}

else

{

Log.i(TAG, "in else... ");

}

}

private void changemode(String m\_mode) {

// TODO Auto-generated method stub

if(m\_mode.equals("silent"))

{

System.out.println("The phone state is changing to silent mode");

// int n=maudio.getRingerMode();

// System.out.println("The phone state is changing to silent mode "+n);

maudio.setRingerMode(AudioManager.RINGER\_MODE\_SILENT);

Log.i(TAG, "Changed to Silent");

}

else if(m\_mode.equals("ring"))

{

System.out.println("The phone state is changing to ring mode");

maudio.setRingerMode(AudioManager.RINGER\_MODE\_NORMAL);

Log.i(TAG, "Changed to Ring ");

}

else if(m\_mode.equals("vibrate"))

{

System.out.println("The phone state is changing to vibrate mode");

maudio.setRingerMode(AudioManager.RINGER\_MODE\_VIBRATE);

Log.i(TAG, "Changed to Vibrate");

}

else if(m\_mode.equals("") || m\_mode==null)

{

Log.i(TAG, "Other message Recieved...");

// Toast.makeText(context,"Other message Recieved...",Toast.LENGTH\_LONG).show();

}

}

private String getMode(String mode,Context context) {

// TODO Auto-generated method stub

String m="";

this.myDbHelper=new Dbhandler(context);

FetchingData();

System.out.println("inside getmodde");

Mydatabase=myDbHelper.getReadableDatabase();

Cursor c=Mydatabase.rawQuery("SELECT Mode FROM mode where text='"+mode+"' ", null);

System.out.println("cursor is.."+c);

c.moveToFirst();

if(c!=null){

int i=c.getColumnIndex("Mode");

System.out.println("column index is.."+i);

m =c.getString(i).toString();

System.out.println("mode in getmode.."+m);

}

return m;

}

private void FetchingData() {

// TODO Auto-generated method stub

try {

myDbHelper.onCreateDataBase();

} catch (IOException ioe) {

throw new Error("Unable to create database");

}

try {

myDbHelper.openDataBase();

db = myDbHelper.getReadableDatabase();

System.out.println("executed");

}catch(SQLException sqle){

throw sqle;

}

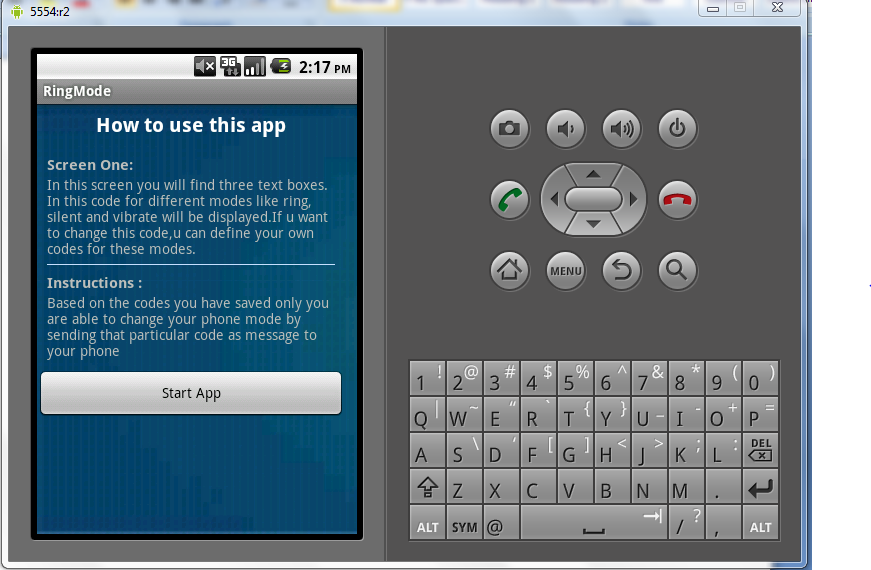
// TODO Auto-generated method stu

}

}

**5.2 Screen Shots:**

**Start Up Screen**



*Fig 5.1 startup screen 1*



*Fig 5.2 startup screen 2*

**5.3 Method of Implementation**

Implementation literally means to put into effect or to carry out. The system implementation phase of the software deals with the translation of the design specifications into the source code. The ultimate goal of the implementation is to write the source code and the internal documentation so that it can be verified easily. The code and documentation should be written in a manner that eases debugging, testing and modification. System flowcharts, sample run on packages, sample output etc. Is part of the implementation?

An effort was made to satisfy the following goals in order specified.

• Minimization of Response Time.

• Clarity and Simplicity of the Code.

• Minimization of Hard-Coding.

**6. TESTING AND VALIDATION**

The development of software systems involves of a series of production activities where opportunities for injection of human fallibilities are enormous. Errors may begin to occur at the very inception of the process where the objectives may be erroneously or imperfectly specified, as well as in later design and development stages. Because of human inability to perform and communicate with perfection, software development is accompanied by a quality assurance activity.

# TESTING TECHNIQUES

Testing is the process of executing a program with the intention of finding errors. The various test strategies used for testing the software are as follows.

**6.1 Unit Testing**

Unit testing focuses on verification effort on the smallest unit of the software design module. The main goal is to make sure that every source statement and logic path has been executed correctly at least once. The output of this stage is the source code.

**6.2 Integration Testing**

In Integration testing, we find errors that have occurred during the integration. After testing each module, which is then integrated into subsystems and then to form the entire system on which integration testing is performed. The goal of testing is to detect the design errors, while focusing on the testing the interconnection between modules.

## 6.3 Validation Testing

This testing concentrates on confirming that the software is error-free in all respects. All the specified validations are verified and the software is subjected to hard-core testing. It also aims at determining the degree of deviation that exists in the software designed from the specification; they are listed out and are corrected.

**6.4 System Testing**

In this testing, the system is tested for the errors after coupling all the modules together The system is tested against the specified requirements to see if all the requirements are met and the system performs as specified by the requirements.

**7. CONCLUSION**

**7.1 CONCLUSION**

This application is used for changing phone mode. This application is build in Google mobiles using Android SDK. It is a tool developed for android platform, which is used to search various books and their related information within the mobile. This is an advantage when compared to existing system because a single mobile piece is enough for deploying the application .As this is a mobile application one can easily search for required information. One can search for books whenever one wants to without waiting for some system. This makes this application efficient, convenient and easy to use along with providing maximum user satisfaction which is the key aspect for any developer.

## 

**8.REFERENCES**

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

7th Edition, pp.177-180 .

2. Grady Booch, 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

## 